Natural Proofs Versus Derandomization∗

Ryan Williams†

Abstract

We study connections between Natural Proofs, derandomization, and the problem of proving “weak” circuit lower bounds such as $\text{NEXP} \not\subset \text{TC}^0$, 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:

• Constructivity is unavoidable, even for $\text{NEXP}$ lower bounds. Informally, we prove for all “typical” non-uniform circuit classes $C$, $\text{NEXP} \not\subset C$ if and only if there is a polynomial-time algorithm distinguishing some function from all functions computable by $C$-circuits. Hence $\text{NEXP} \not\subset C$ is equivalent to exhibiting a constructive property useful against $C$.

• There are no $\text{P}$-natural properties useful against $C$ if and only if randomized exponential time can be “derandomized” using truth tables of circuits from $C$ as random seeds. Therefore the task of proving there are no $\text{P}$-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 $\text{ACC}$ lower bounds and new unconditional derandomizations.

1 Introduction

The Natural Proofs barrier of Razborov and Rudich [RR97] argues that

(a) almost all known proofs of non-uniform 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 preliminary version of this paper appeared in the ACM Symposium on Theory of Computing in 2013.
†Computer Science Department, Stanford University, rrw@cs.stanford.edu. Supported in part by a David Morgenthaler II Faculty Fellowship, a Sloan Fellowship, a Microsoft Research Faculty Fellowship, 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.
Natural Proofs are self-deceiving: 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 $\text{TC}^0$ [NR04, KL01, MV12], so any major progress in the future depends on proving un-Natural 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 $C$ we are proving lower bounds against. This criterion is necessary: $f \notin C$ if and only if there is a trivial property, true of only $f$, distinguishing $f$ from all functions computable in $C$. ) 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 EXP $\not\subseteq C$, one can obtain a polynomial-time (non-large) property useful against $C$.2 So, strong enough lower bounds entail constructive useful properties. However, a separation like EXP $\not\subseteq C$ is stronger than currently known, for all classes $C$ containing ACC. Could lower bounds be proved for larger classes like NEXP, without entering constructive/useful territory? In the other direction, could one exhibit a constructive (non-large) property against a small circuit class like $\text{TC}^0$, without proving a new lower bound against that class?

The answer to both questions is no. Call a (non-uniform) circuit class $C$ typical if $C \in \{\text{AC}^0, \text{ACC}, \text{TC}^0, \text{NC}^1, \text{NC}, \text{P/poly}\}$.3 For any typical $C$, a property $P$ of Boolean functions is said to be useful against $C$ if, for all $k$, there are infinitely many $n$ such that
\begin{itemize}
  \item $P(f)$ is true for at least one $f : \{0,1\}^n \to \{0,1\}$, and
  \item $P(g)$ is false for all $g : \{0,1\}^n \to \{0,1\}$ having $n^k$ size $C$-circuits.
\end{itemize}
In other words, $P$ distinguishes some function from all easy functions. We prove:

**Theorem 1.1** For all typical $C$, NEXP $\not\subseteq C$ if and only if there is a polynomial-time property of Boolean functions that is useful against $C$.

Theorem 1.1 helps 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

---

1See the webpage [Aar07] for a discussion with many views on these questions.
2Define $A(T)$ to accept its $2^n$-bit input $T$ if and only if $T$ is the truth table of a function that is complete for $\text{E} = \text{TIME}[2^{O(n)}]$. $A$ can be implemented to run in $\text{poly}(2^n)$ time and rejects all $T$ with $C$ circuits, assuming $\text{EXP} \not\subseteq C$.
3For 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.
4One may also wonder if non-constructive 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 $\text{P/poly}$ (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 [Rud97].
One can make a heuristic argument that the recent proof of \( \text{NEXP} \not\subset \text{ACC} \) ([Wil11]) evades Natural Proofs by being non-constructive. Intuitively, the proof uses an ACC Circuit SAT algorithm that only mildly improves over brute force, so it runs too slowly to obtain a polytime property useful against ACC. Theorem 1.1 shows that, in fact, constructivity is necessary. Moreover, the proof of Theorem 1.1 yields an explicit property useful against ACC.

The techniques used in Theorem 1.1 can be applied, along with several other ideas, to prove new super-polynomial lower bounds against ACC. First, we prove exponential-size lower bounds on the ACC circuit complexity of encoding witnesses for NEXP languages.

**Theorem 1.2** For all \( d, m \) there is an \( \varepsilon > 0 \) such that \( \text{NTIME}[2^{O(n)}] \) does not have \( 2^{n^\varepsilon} \)-size \( d \)-depth \( \text{AC}^0[m] \) witnesses.

Formal definitions can be found in Section 3; informally, Theorem 1.2 says that there are NEXP languages with verifiers that only accept witness strings of exponentially high ACC 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 lower bounds for some complexity classes. Recall that \( \text{NE} = \text{NTIME}[2^{O(n)}] \) and \( \text{io-coNE} = \text{io-coNTIME}[2^{O(n)}] \), the latter being the class of languages \( L \) such that there is an \( L' \in \text{coNTIME}[2^{O(n)}] \) where, for infinitely many \( n \), \( L \cap \{0,1\}^n = L' \cap \{0,1\}^n \). That is, \( L \) agrees with a language in \( \text{coNTIME}[2^{O(n)}] \) on infinitely many input lengths.

The class \( \text{NE}/1 \cap \text{coNE}/1 \) consists of languages \( L \in \text{NE} \cap \text{coNE} \) 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 \( n \).

**Theorem 1.3** \( \text{NE} \cap \text{io-coNE} \) and \( \text{NE}/1 \cap \text{coNE}/1 \) do not have ACC circuits of \( n^{\log n} \) size.\(^5\)

This lower bound is intriguing not just because \( \text{NE} \cap \text{io-coNE} \subseteq \text{NE} \) (and it is believed that the containment is proper), but also because the lower bound necessarily must be proved differently. The known proof of \( \text{NEXP} \not\subset \text{ACC} \) works for the class NEXP because there is a tight time hierarchy for nondeterminism [Z83]. However, the \( \text{NTIME} \cap \text{coNTIME} \) classes (and \( \text{NTIME} \cap \text{io-coNTIME} \) 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, our proof of Theorem 1.3 crucially uses the previous lower bound framework against NEXP, and builds on it, via Theorem 1.1 and a modification of the \( \text{NEXP} \not\subset \text{ACC} \) lower bound. Indeed, it follows from the arguments here (building on [Wil10, Wil11]) that the lower bound consequences of non-trivial circuit SAT algorithms can be strengthened, in the following sense:

**Theorem 1.4** Let \( C \) be typical. Suppose the satisfiability problem for \( n^{O(\log^c n)} \)-size \( C \) circuits can be solved in \( O(2^n/n^{10}) \) time, for all constants \( c \). Then \( \text{NE} \cap \text{io-coNE} \) and \( \text{NE}/1 \cap \text{coNE}/1 \) do not have \( n^{\log n} \)-size \( C \) circuits.

\(^5\)This is not the strongest size lower bound that can be proved, but it is among the cleanest. Please note that the conference version of this paper claimed a lower bound for the (hypothetically larger) class \( \text{NE} \cap \text{coNE} \); we are grateful to Russell Impagliazzo and Igor Carboni Oliveira for observing that our argument only proves a lower bound for \( \text{NE} \cap \text{io-coNE} \) (and \( \text{NE} \cap \text{coNE} \) with one bit of advice, under the appropriate definition).
Theorem 1.5 Suppose we can approximate the acceptance probability of any given $n^{O(\log^c n)}$-size circuit with $n$ inputs to within $1/6$, for all $c$, in $O(2^n/n^{10})$ time (even nondeterministically). Then $\text{NE} \cap \text{io-coNE}$ and $\text{NE}/1 \cap \text{coNE}/1$ do not have $n^{\log^2 n}$-size circuits.

Natural Proofs vs derandomization  Given Theorem 1.1, it is natural to wonder if 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 $\text{RE} = \text{RTIME}[2^{O(n)}]$ and $\text{ZPE} = \text{ZTIME}[2^{O(n)}]$; 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 $C$, we informally say that RE (respectively, ZPE) has $C$ seeds if, for every predicate defining a language in the respective complexity class, there are $C$ circuit families succinctly encoding exponential-length “seeds” that correctly decide the predicate. (Formal definitions are given in Section 4.) Having $C$ seeds means that the randomized class can be derandomized very strongly: by trying all poly-size $C$ circuits as random seeds, one can decide any predicate from the class in EXP.

We give a tight correspondence between the existence of such seeds, and the nonexistence of natural properties:

Theorem 1.6 The following are equivalent:

1. There are no $\text{P}$-natural properties useful (respectively, ae-useful$^6$) against $C$
2. ZPE has $C$ seeds for almost all (resp., infinitely many) input lengths

Informally, the theorem says that ruling out $\text{P}$-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 4.1).

It is worth discussing the meaning of these results in a little more detail. Let $C, D$ be appropriate circuit classes. Roughly speaking, the key lesson of Natural Proofs [RR97, NR04, KL01] is that, if there are $D$-natural properties useful against $C$, then there are no pseudorandom functions (PRFs) computable in $C$ that fool $D$ circuits; namely, there is a statistical test $T$ computable in $D$ such that, for every function $f \in C$ (armed with an $n$-bit initial random seed), the test $T$ with query access to $f$ can distinguish $f$ from a uniform random function. Now, if we have a PRF computable in $C$ that can fool $D$ circuits, this PRF can be used to obtain $C$ seeds for randomized $D$ circuits with one-sided error.$^7$ That is, the existence of PRFs implies the existence of $C$ seeds, so our consequence in Theorem 1.6 (of the existence of natural properties) that “no ZPE predicate has $C$ seeds” appears stronger than “there are no PRFs.” Moreover, this stronger consequence in Theorem 1.6 (and Theorem 4.1, proved later) yields an implication in the reverse direction: the lack of $D$-natural properties implies strong derandomizations of randomized exponential-size $D$.

Theorem 1.6 also shows that some plausible derandomization problems are as hard as resolving $P \neq \text{NP}$. Suppose assuming $P = \text{NP}$, we can give a “canonical” derandomization of ZPE in EXP, along the lines of item (2) in Theorem 1.6. (Clearly if $P = \text{NP}$, we have $\text{ZPE} \subseteq \text{NEXP} = \text{EXP}$, but we ask for a special derandomization here.) By Theorem 1.6, there are no $P$-natural properties useful against $P/poly$. However, there are $\text{coNP}$-natural properties of this kind, so $P \neq \text{NP}$ would follow(!).

$^6$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.

$^7$Consider any $D$-circuit $D$ that tries to use $f$ as a source of randomness. A $C$-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 $D'$ print 1.
**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

\[ P \subseteq ZPP = \text{RP} \cap \text{coRP} \subseteq \text{RP} \subseteq \text{BPP} \]

but it is not known if any inclusion is an equality. The ideas behind Theorem 1.6 can also be applied to prove new relations between these classes. We define \( \text{ZPTIME}[t(n)]/d(n) \) to be the class of languages solvable in zero-error time \( t(n) \) by machines of description length at most \( d(n) \).\(^8\) The “infinitely often” version \( \text{io-ZPTIME}[t(n)]/d(n) \) 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 1.7**  Either \( \text{RTIME}[2^{O(n)}] \subseteq \text{SIZE}[n^c] \) for some \( c \), or \( \text{BPP} \subseteq \text{io-ZPTIME}[2^{n^c}]/n^\epsilon \) for all \( \epsilon > 0 \).

We have a win-win: either randomized exptime is very easy with non-uniform circuits, or randomized computation with two-sided error has a zero error simulation (with description size \( n^\epsilon \)) that dramatically avoids brute-force. To appreciate the theorem statement, suppose the first case could be modified to conclude that \( \text{RP} \subseteq \text{io-ZPTIME}[2^{n^c}]/n^\epsilon \) for all \( \epsilon > 0 \). Then the famous (coRP) problem of Polynomial Identity Testing would have a new subexponential-time algorithm, good enough to prove strong \( \text{NEXP} \) circuit lower bounds.\(^9\) A quick corollary of Theorem 1.7 comes close to achieving this. To simplify notation, we use the \( \text{SUBEXP} \) modifier in a complexity class to abbreviate “\( 2^{n^c} \) time, for every \( \epsilon > 0 \).”

**Corollary 1.1**  For some \( c \), \( \text{RP} \subseteq \text{io-ZPSUBEXP}/n^c \).

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 \( n^{o(c)} \) time and still need only \( n^c \) advice to be simulated in \( 2^{n^{1/o(c)}} \) time. Corollary 1.1 should be compared with a theorem of Kabanets [KI04], 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:

**Corollary 1.2**  For some \( c \geq 1 \), \( \text{AM} \subseteq \text{io-}\Sigma_2 \text{SUBEXP}/n^c \).

The ideas used here can also be applied to prove a new equivalence between \( \text{NEXP} = \text{BPP} \) and nontrivial simulations of BPP:

**Theorem 1.8**  \( \text{NEXP} \neq \text{BPP} \) if and only if for all \( \epsilon > 0 \), \( \text{BPP} \subseteq \text{io-HeuristicZTIME}[2^{n^c}]/n^\epsilon \).

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:

---

\(^8\) **N.B.** Although our definition is standard (see for example [Bar02, FST05]), 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 that the algorithm is zero-error no matter what advice is given.

\(^9\) More precisely, the main result of Kabanets and Impagliazzo [KI04] concerning the derandomization of Polynomial Identity Testing (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 \( \text{NEXP} \not\subseteq P/poly \) or the Permanent does not have polynomial-size arithmetic circuits.
Theorem 1.9 If there exists a RP-natural property \( P \) useful against a class \( C \), then there exists a P-natural property \( P' \) against \( C \).

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 \( P \).

2 Preliminaries

For simplicity, all languages are over \( \{0,1\} \). We assume knowledge of the basics of complexity theory [AB09] such as advice-taking machines, and complexity classes like \( \text{EXP} = \text{TIME}[2^{n^{O(1)}}] \), \( \text{NEXP} = \text{NTIME}[2^{n^{O(1)}}] \), \( \text{AC}^0[m] \), \( \text{ACC} \), and so on. We use \( \text{SIZE}[s(n)] \) to denote the class of languages recognized by a (non-uniform) \( s(n) \)-size circuit family. We also use the (standard) “subexponential-time” notation \( \text{SUBEXP} = \bigcap_{k>0} \text{TIME}[2^{o(n^k)}] \). (So for example, \( \text{NSUBEXP} \) refers to the class of languages accepted in nondeterministic \( 2^{o(n^k)} \) time, for all \( \epsilon > 0 \).) When we refer to a “typical” circuit class (\( \text{AC}^0 \), \( \text{ACC} \), \( \text{TC}^0 \), \( \text{NC}^1 \), \( \text{NC}^2 \), \( \text{NC}^3 \), \( \text{NC}^4 \), or \( \text{P(poly)} \)), we will always assume the class is non-uniform, unless otherwise specified. Some familiarity with prior work connecting SAT algorithms and circuit lower bounds [Wil10, Wil11] would be helpful, but this paper is mostly self-contained.

We will use advice classes: for a deterministic or nondeterministic class \( C \) and a function \( a(n) \), \( C/a(n) \) is the class of languages \( L \) such that there is an \( L' \in C \) and an arbitrary function \( f : \mathbb{N} \to \{0,1\}^* \) with \( |f(n)| \leq a(n) \) 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 \( C \).

For randomized classes \( C \), the definition of advice is technically subtle. For a given randomized machine \( M \) and class \( C \in \{\text{RTIME}[t(n)], \text{ZPTIME}[t(n)], \text{BPTIME}[t(n)]\} \), we say that \( M \) is of type \( C \) 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 \( \top \) (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 \( \bot \) on the others.) Then for \( C \in \{\text{RTIME}[t(n)], \text{ZPTIME}[t(n)], \text{BPTIME}[t(n)]\} \), \( C/a(n) \) is the class of languages \( L \) recognized by a randomized machine of description length \( a(n) \) that is of type \( C \) on all inputs [Bar02]. Equivalently, \( L \in C/a(n) \) 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 \( x \in \{0,1\}^* \), \( M \) is a machine of type \( C \) 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 [FST05].

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

Some particular notation and conventions will be useful for this paper. For any circuit \( C(x_1, x_2, \ldots, x_n) \), \( i < j \), and \( a_1, \ldots, a_n \in \{0,1\} \), the notation \( C(a_1, \ldots, a_i, a_j, \ldots, a_n) \) 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, \( \cdot \) is used to denote free unassigned inputs to the circuit.
For a Boolean function \( f : \{0, 1\}^n \rightarrow \{0, 1\} \), the truth table of \( f \) is defined to be
\[
\text{tt}(f) := f(y_1)f(y_2) \cdots f(y_{2^n}),
\]
and the truth table of a circuit is simply the truth table of the function defining it. For binary strings with lengths that are not powers of two, we use the following encoding convention. Let \( T \) be a binary string, let \( k = \lceil \log_2 |T| \rceil \), and let \( y_1, \ldots, y_{2^k} \in \{0, 1\}^k \) be the list of \( k \)-bit strings in lexicographic order. The function encoded by \( T \), denoted as \( f_T \), is the function satisfying \( \text{tt}(f_T) = T0^{2^k-|T|} \).

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 \( \text{NC}^1 \), we have to specify a depth upper bound \( c \log n \) for some constant \( c \).) For the class \( \text{ACC} \), we must specify a modulus \( m \) for the \( \text{MOD} \) gates, as well as a depth bound, so when considering \( \text{ACC} \) 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 \( 2^{k+\ell} \), where \( T_1, \ldots, T_{2^k} \) each have length \( 2^\ell \). Then \( CC(T_i) \leq CC(T) \), \( CC_d(T_i) \leq CC_d(T) \), and \( CC_{d,m}(T_i) \leq CC_{d,m}(T) \).

**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 \( s \). \( \square \)

We will sometimes need 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 \( c \geq 1 \) such that the following holds. Let \( T \) be a binary string, and let \( S \) be any substring of \( T \). Then for all \( d \) and \( m \), \( CC(f_S) \leq CC(f_T) + (c \log |T|) \), \( CC_d(f_S) \leq CC_d(f_T) + (c \log |T|)^{1+o(1)} \), and \( CC_{d,m}(f_S) \leq CC_{d+e,m}(f_T) + (c \log |T|)^{1+o(1)} \).

**Proof.** Let \( c' \) be sufficiently large in the following. Let \( k \) be the minimum integer satisfying \( 2^k \geq |T| \), so 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 \( \ell \leq k \) be a minimum integer which satisfies \( 2^\ell \geq B - A \). Our goal is to construct a small circuit \( D \) with \( \ell \) inputs and truth table \( S0^{2^{\ell}-(B-A)} \).

Let \( x_1, \ldots, x_{2^\ell} \) be the \( \ell \)-bit strings in lexicographic order. The desired circuit \( D \) on input \( x_i \) can be implemented as follows: Compute \( i + A \). If \( i + A \leq B - A \) 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 \( c' \cdot n \log^* n \) size for addition of two \( n \)-bit numbers [CFL85]. Therefore in depth-\( c' \) and size at most \( c' \cdot k \log^* k \) we can, given input \( x_i \) of length \( \ell \), output \( i + A \). Determining if \( i + A \leq B - A \) can be done with \( (c' \cdot \ell) \)-size depth-\( c' \) circuits. Therefore \( D \) can either be implemented as a circuit of size at most \( s + c'(k \log^* k) + 1 \) and depth \( 2c'd \), or as an (unrestricted depth) circuit of size at most \( s + c'(k + \ell + 1) \). To complete the proof, let \( c \geq 3c' \). \( \square \)

We will use the following strong construction of pseudorandom generators from hard functions:
Theorem 2.1 (Umans [Uma03]) There is a universal constant $g$ and a function $G : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ such that, for all $s$ and $Y$ satisfying $CC(Y) \geq s^g$, and for all circuits $C$ of size $s$,

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

Furthermore, $G$ is computable in $\text{poly}(|Y|)$ time.

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

(Constructivity) $\mathcal{P}$ is decidable in $\Gamma$,

(Largeness) for all $n$, $\mathcal{P}$ contains a $1/2^{O(n)}$ fraction of all $n$-bit inputs,

(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.\(^\text{10}\)

Let $f = \{f_n : \{0, 1\}^n \rightarrow \{0, 1\}\}$ be a sequence of Boolean functions. A $\Gamma$-natural proof that $f \not\in \mathcal{C}$ establishes the existence of a $\Gamma$-natural property $\mathcal{P}$ useful against $\mathcal{C}$ such that $\mathcal{P}(f_n) = 1$ for all $n$. Razborov and Rudich proved that any $\mathbb{P}/\text{poly}$-natural property useful against $\mathbb{P}/\text{poly}$ could break all strong pseudorandom generator candidates in $\mathbb{P}/\text{poly}$. More generally, $\mathbb{P}/\text{poly}$-natural properties useful against typical $\mathcal{C} \subset \mathbb{P}/\text{poly}$ imply there are no strong pseudorandom functions in $\mathcal{C}$ (but such functions are believed to exist, even when $\mathcal{C} = \mathbb{TC}^0$ [NR04]).

2.1 Related Work

Equivalences between algorithms & 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 [NW94] famously proved an equivalence between “approximate” circuit lower bounds and the existence of pseudorandom generators. Impagliazzo and Wigderson [IW01] prove that $\text{BPP} \neq \text{EXP}$ implies deterministic subexponential-time heuristic algorithms for $\text{BPP}$ (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 [IKW02] proved another such equivalence, which we discuss below.) Two more recent examples are Jansen and Santhanam [JS12], who give an equivalence between nontrivial algorithms for polynomial identity testing and lower bounds for the algebraic version of $\text{NEXP}$, and Aydinlioglu and Van Melkebeek [AvM12], who give an equivalence between $\Sigma_2$-simulations of Arthur-Merlin games and circuit lower bounds for $\Sigma_2\text{EXP}$.

Almost-Natural Proofs PHILO

Note that some papers replace ‘infinitely many’ with ‘almost every’; in this paper, we call that version $ae$-usefulness.
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 \( \text{SIZE}[O(n)] \)-natural property that is \( 1/2^{n^{(\log n)^{o(1)}}} \)-large and useful against P/poly. Theorem 1.1 shows that if \( \text{SIZE}[O(n)] \) could be replaced with P, then NEXP \( \not\subset \) P/poly follows.

The work of IKW Impagliazzo-Kabanets-Wigderson [IKW02] proved a theorem similar to one direction of Theorem 1.1, showing that an NP-natural property (without largeness) useful against P/poly implies NEXP \( \not\subset \) P/poly. Allender [All01] proved that there is a (non-large) property computable in NP useful against P/poly if and only if there is such a property in uniform \( \text{AC}^0 \). Hence his equivalence implies, at least for \( C = \text{P/poly} \), that the “polynomial-time” guarantee of Theorem 1.1 can be relaxed to “\( \text{AC}^0 \).”

IKW [IKW02] also give an equivalence between NEXP lower bounds and an algorithmic problem: NEXP \( \not\subset \) P/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 Theorem 1.1 are in the underlying computational problems and the algorithmic guarantees: they study subexponential-time algorithms for approximating acceptance probabilities, while we study P-useful properties. 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 \( \text{ACC} \).

IKW posed the interesting open problem: does the existence of a P-natural property useful against P/poly imply EXP \( \not\subset \) P/poly? Our work shows that the absence of a P-natural property implies new lower bounds:

**Claim 2.1** If there is no P-natural property useful against P/poly, then NP \( \not= \) ZPP.

In brief, NP = ZPP implies the existence of a ZPP-natural property (since there are trivially coNP-natural properties), and Theorem 1.9 can be used to “derandomize” natural properties generically, yielding a P-natural one. Therefore, an affirmative solution to IKW’s problem would yield a proof that EXP \( \not= \) ZPP.

### 3 NEXP Lower Bounds and Useful Properties

In this section, we prove:

**Reminder of Theorem 1.1** For all typical \( C \), NEXP \( \not\subset C \) if and only if there is a polynomial-time property of Boolean functions that is useful against \( C \).

The proof of Theorem 1.1 takes several steps. 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 results of Impagliazzo, Kabanets, and Wigderson [IKW02] (who essentially proved one direction of the equivalence). Second, we prove an equivalence between the non-existence 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 \( P^* \): if for every circuit size \( s \) there is some P-constructive useful property \( P_s \), this particular property \( P^* \) 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.
**Definition 3.1** Let $L \in \text{NTIME}[t(n)]$ where $t(n) \geq n$ is constructible, and let $C$ be a circuit class. An algorithm $V(x, y)$ is a predicate for $L$ if $V$ runs in time $O(|y|) + t(|x|)$ and for all strings $x$,

$$x \in L \iff \text{there is a } y \text{ of length } O(t(n)) \text{ (a witness for } x) \text{ such that } V(x, y) \text{ accepts.}$$

We denote $L(V)$ to be the language accepted by $V$. $V$ has $C$ witnesses of size $s(n)$ if for all strings $x$, if $x \in L$ then there is a $C$-circuit $C_x$ of size at most $s(n)$ such that $V(x, C_x(\cdot))$ accepts. $L$ has $C$ witnesses of polynomial size if for all predicates $V$ for $L$, there is a polynomial $p(n)$ such that $V$ has $C$ witnesses of size $O(p(n))$.\(^{11}\) \text{NTIME}[t(n)] \text{ has } C \text{ witnesses if for every infinite language } L \in \text{NTIME}[t(n)], L \text{ has } C \text{ witnesses of polynomial size.}\(^{12}\)

The above definition 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 3.2** Let $L \in \text{NTIME}[t(n)]$, let $C$ be a circuit class, and let $V$ be a predicate for $L$. $L$ has oblivious $C$ witnesses of size $s(n)$ if for every predicate $V$ for $L$, there is a $C$ circuit family $\{C_n\}$ of size $s(n)$ such that for all $x \in \{0, 1\}^*$, if $x \in L$ then $V(x, tt(C_{|x|}(x, \cdot)))$ accepts.\(^{12}\) Finally, $\text{NTIME}[t(n)]$ has oblivious $C$ witnesses if every infinite $L \in \text{NTIME}[t(n)]$ has oblivious $C$ witnesses.

We first prove an equivalence between the existence of small circuits for $\text{NEXP}$ and small circuits for $\text{NEXP}$ witnesses, in both the oblivious and normal senses. Let $C$ be a typical circuit class.

**Theorem 3.1** The following are equivalent:

1. $\text{NEXP} \subset C$
2. $\text{NEXP}$ has $C$ witnesses of polynomial size
3. $\text{NEXP}$ has oblivious $C$ witnesses of polynomial size

**Proof.** (1) $\Rightarrow$ (2) Impagliazzo, Kabanets, and Wigderson [IKW02] proved this direction for $C = \mathbb{P}/\text{poly}$. The other cases of $C$ were proved in prior work [Wil10, Wil11].

(2) $\Rightarrow$ (3) Assume $\text{NEXP}$ has $C$ witnesses of polynomial size. Let $V(x, y)$ be an $\text{NEXP}$ predicate that (without loss of generality) accepts witnesses $y$ of length exactly $2^{|x|k}$. We will construct a $C$-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 are 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. Define a new 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 $2^{n+n^k}$:

$$V'(x, q, y): \text{Accept if and only if } y = z_1 \cdots z_{2^{|x|}}, \text{ where for all } i, z_i \in \{0, 1\}^{2^{|x|k}}, \text{ exactly } 2^n - q \text{ of the } z_i \text{ strings equals the all-zeroes string, and for all other } q \text{ strings } z_j, V(x_j, z_j) \text{ accepts.}$$

\(^{11}\)For circuit classes $C$ where the depth $d$ and/or modulus $m$ may be bounded, we also quantify this $d$ and $m$ simultaneously with the size parameter $p(n)$. 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 $x$.\(^{12}\)
Theorem 3.2

For all \( |x| \) runs in time exponential in \(|x|\); by assumption, \( V' \) has \( C \) witnesses of polynomial size. Observe that the computation of \( V' \) does not depend on the input \( x \).

To obtain oblivious \( C \) witnesses for \( V \), let \( q_n \) be the number of \( x \) of length \( n \) such that \( x \in L(V) \). Then for every \( y^n \) such that \( V'(x', q_n, y^n) \) accepts, the string \( y^n \) must encode a valid witnesses \( z_i \) for every \( x_i \in L(V) \), and all-zero strings for every \( x_j \notin L(V) \). By assumption, there is a circuit \( C(x', q_n) \) such that \( C(x', q_n)(i) \) outputs the \( i \)th bit of \( y^n \). 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 \( j \)th bit of a valid witness for \( x \) (or 0, if \( x \notin L(V) \)).

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

\[
V_k(x, y): \text{For all circuits } C \text{ of size } |x|^k + k, \\
\text{if } tt(C) \text{ encodes an accepting computation history of } M(x), \text{ then } \\
\text{accept if and only if the first bit of } y \text{ is } 1.
\]

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 \((n^k + k)\)-size \( C \) circuit family. 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

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

Because \( V_k \) is an \( \text{NEXP} \) predicate, we can apply the assumption again to \( V_k \) itself, meaning there is a \( C \)-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 \( \text{NE} \) computations and constructive useful properties. (This equivalence will be useful for proving new consequences later.) Here, the typical circuit class \( C \) does not have to be polynomial-size bounded; the size function \( s(n) \) can be any reasonable function in the range \([n^2, 2^n/(2n)]\) (for example).

Theorem 3.2 For all \( s(n) \), the following are equivalent:

1. There is a \( c \in (0, 1] \) such that \( \text{NTIME}[2^{O(n)}] \) does not have \( s(cn) \) size witness circuits from \( C \).

2. There is a \( c \in (0, 1] \) and a \( \text{P} \)-computable property that is useful against \( C \)-circuits of size at most \( s(cn) \).\(^{13}\)

Proof. (2) \( \Rightarrow \) (1) Let \( A \) be a poly\((n)\)-time algorithm which takes \( n \) bits of input \( T \) and is useful against size-\( s(cn) \) \( C \). Define a machine \( M(x, T) \) which rejects if \(|T| \neq 2^{2|x|}\), otherwise, it partitions \( T \) into strings \( T_1, \ldots, T_{2^{2|x|}} \) of \( 2^{2|x|} \) bits each, and accepts iff \( A(T_x) \) accepts, where \( T_x \) is the \( x \)th block of \( T \) (treating \( x \) as an integer from 1 to \( 2^{2|x|} \)).

\(^{13}\)For circuit classes \( C \) with depth bound \( d \), this \( d \) will be universally quantified after \( c \). So for example, there is a \( c \) such that for all constant \( d \), \( \text{NTIME}[2^{O(n)}] \) does not have \( s(cn) \) size depth-\( d \) \( AC^0[6] \) witnesses, if and only if there is a \( c \) such that for all \( d \), there is a \( \text{P} \)-computable property useful against depth-\( d \) \( AC^0[6] \) circuits of size \( s(cn) \).
Define $L = \{ x \mid (\exists T : |T| = 2^{|x|})[M(x,T) \text{ accepts}] \}$. Clearly $L \in \text{NTIME}[2^{O(n)}]$. Suppose for contradiction that $\text{NTIME}[2^{O(n)}]$ has $s(dn)$ size witnesses, for all $d \in (0,1]$. Then for almost every $\ell$, if $x \in L$ and $|x| = \ell$ then there is a circuit $C$ with $2\ell$ inputs and size at most $s(d\ell)$, such that $M(x,tt(C))$ accepts. By definition of $M$, this means the $\ell$th block of $tt(C)$ (call it $T_x$) is accepted by $A$. By Proposition 1, the $C$-circuit complexity of $T_x$ is at most $s(d\ell)$ (with the same depth/modulus), as it is a substring of $tt(C)$. However, by assumption on $A$, there are infinitely many $\ell$ such that $T_x$ must have circuit complexity greater than $s(cf\ell)$, a contradiction when $d < c$.

(1) $\Rightarrow$ (2) Suppose $\text{NTIME}[2^{O(n)}]$ does not have $s(n)$-size witness circuits. Let $V$ be such a predicate (running in $\text{NTIME}[2^{cn}]$ for some $c \geq 1$) that does not have $s(n)$-size witnesses, but $L(V)$ is infinite. Then there is an infinite subsequence of inputs $\{x_i'\}$ such that $x_i' \in L(V)$, but for every $x_i'$ and every $y$ such that $M(x_i',y)$ accepts, $y$ requires $s(|x_i'|)$ size circuits to encode.

We give a polynomial-time algorithm $A$ that is useful. The trick (similar to the (2) to (3) direction of Theorem 3.1) is to “merge” all the witnesses for a given length into one string. When $A$ is given a string $y$, if $y$ does not have the form $y_1 \cdots y_{2^\ell}01^k$ for some $k = 0, \ldots, 2^\ell$ and $\ell$ where every $|y_i| = 2^\ell$, then $A$ rejects. Otherwise, $A$ extracts $k$ from $y$ by counting the trailing 1’s on the end of the string, and accepts if and only if $k$ equals the number of inputs such that $V(x_i,y_i)$ accepts (here, $x_i$ is the $i$th $\ell$-bit string in lex order).

Suppose the $k$ extracted from $A$ is exactly the number of inputs of length $\ell$ accepted by $V$. Then, for every $x$ of length $\ell$ accepted by $V$, the corresponding substring $y_i$ in $y$ is a witness for $x$. If there is an $i,j$ such that $x_j = x_i'$ (the $k$th $\ell$-bit string equals the $i$th string in the infinite sequence), then every $y'$ that is a witness for $x_j$ requires at least $s(\ell)$-size circuits. Hence the substring $y_j$ requires $s(\ell)$-size circuits, so by Lemma 1, $y$ requires circuits of size $s(\ell) - \ell^{1+o(1)} \geq s(\ell)/2$ (with depth possibly smaller, by a universal additive constant).

Notice that, for each $\ell$, and every possible $k = 0, \ldots, 2^\ell$, there is precisely one input length, namely $n = 2^{(c+1)\ell} + k + 1$, for which this value of $k$ will be considered on $\ell$-bit strings. Therefore, on those input lengths $n$ for which $k$ equals the number of inputs of length $\ell$ accepted by $V$, $A$ is useful against size-$s(\ell)/2$; in terms of the input length $n$, this quantity is at least $s(d\log n)$ for some constant $d > 0$. There are infinitely many such $n$, so this completes the proof.

Using complete languages for $\text{NEXP}$, one can obtain an explicit property in $\text{P}$ that is useful against $C$ 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 $P$-computable property $P_k$ that is useful against all $C$-circuits of $s_k(n)$ size. Then there is a single property $P^*$ such that, for all $k$, there is a $c > 0$ such that $P^*$ is useful against all $C$-circuits of $s_k(cn)$ size.\(^{14}\)

**Proof.** Let $b(n)$ denote the $n$th string of $\{0,1\}^*$ 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 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 $z \in \text{SuccinctHalting}$.

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

12
Observe that HISTORY is implementable in polynomial time. The theorem follows from the 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 P-time property 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 are 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 useful against $\mathcal{C}$-circuits of size $s(n)$. By Theorem 3.2, NTIME$[O(n)]$ does not have $s(dn)$ size witnesses from $\mathcal{C}$ for some constant $d$. Let $V$ be a 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 iff $V(x, y)$ accepts. It follows that there are infinitely many instances of SUCCINCTHALTING of the form $(M, x, b(2^{|x|}))$ that do not have $\mathcal{C}$ witnesses of size $s(cn)$ for some constant $c$. Therefore, there are infinitely many $z_i = (M_i, x_i, n_i)$ 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 $y''$ 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 it all together, we obtain Theorem 1.1:

**Proof of Theorem 1.1.** Let $\mathcal{C}$ be a typical class (of polynomial-size circuits). By Theorem 3.1, we have NEXP $\nsubseteq \mathcal{C}$ 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 is $c > 0$ and a P-computable property that is useful against all $\mathcal{C}$-circuits of size at most $(cn)^k$.

Applying Theorem 3.3, we conclude that NEXP $\nsubseteq \mathcal{C}$ if and only if there is a P-computable property such that, for all $k$, it is useful against all $\mathcal{C}$-circuits of size at most $n^k$. □

## 3.1 New ACC Lower Bounds

In this section, we prove new lower bounds against ACC. Our approach uses a new nondeterministic simulation of randomized computation (assuming small circuits for ACC). The simulation itself uses several ingredients. First, we prove an exponential-size lower bound on the sizes of ACC circuits encoding witnesses for NTIME$[O(n)]$. (Recall that, for NEXP, the best known ACC size lower bounds are only “third-exponential” [Wil11].) 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, then use the hard function to construct a pseudorandom generator. (Here, we will need to make an assumption like P $\subseteq$ ACC, as it is not known how to convert hardness into pseudorandomness in the ACC setting [SV10].)

**Reminder of Theorem 1.2** For all $d$, $m$ there is an $\epsilon = 1/m^{O(d)}$ such that NTIME$[O(n)]$ does not have $2^{\epsilon^d}$-size $d$-depth $\text{AC}^0[m]$ witnesses.\(^{15}\)

\(^{15}\)The $m^{O(d)}$ factor arises from the ACC-SAT algorithm in [Wil11], which in turn comes from Beigel and Tarui’s simulation of ACC in SYM-AND [BT94].
The proof is quite related in structure to the $\text{NEXP} \not\subseteq \text{ACC}$ proof, so we will merely sketch how it is different.

**Proof.** (Sketch) Assume $\text{NTIME}[2^{O(n)}]$ has $2^{n^\varepsilon}$-size ACC witnesses, for all $\varepsilon > 0$. We will show that the earlier framework [Wil11] can be adapted to still establish a contradiction. First, observe the assumption implies that $\text{TIME}[2^{O(n)}]$ has $2^{n^\varepsilon}$-size ACC circuits. (The proof is similar to the proof of Theorem 1.1: for any given exponential-time algorithm $A$, one can set up an $\text{NEXP}$ 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 [Wil11]) there is a nondeterministic $2^{n-n^\delta}$ time algorithm $A$ (where $\delta$ depends on the depth and modulus of ACC circuits for Circuit Evaluation) that, given any circuit $C$ of size $n^{O(1)}$ and $n$ inputs, $A$ generates an equivalent ACC circuit $C'$ of $2^{n^\varepsilon}$ 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 $\text{fail}$.)

The rest of the proof is analogous to prior $\text{NEXP}$ lower bounds [Wil11]; we sketch the details for completeness. Our goal is to simulate every $L \in \text{NTIME}[2^n]$ in nondeterministic time $2^{n-n^3}$, which will contradict the nondeterministic time hierarchy of Žák [Zá63]. Given an instance $x$ of $L$, we first reduce $L$ to the $\text{NEXP}$-complete $\text{SUCCINCT 3SAT}$ 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 $2^{n^\varepsilon}$ size ACC circuit $D'$. Then we guess a $2^{n^\varepsilon}$ size ACC circuit $E$ with truth table equal to a satisfying assignment for $F$. (If $x \in L$, then such a circuit exists, by assumption.) By combining copies of $D'$ and copies of $E$, we can obtain a single ACC 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 $2^{n-n^3}$ time simulation for every $L$, a contradiction. \[\square\]

Applying Theorem 3.2 and its corollary to the lower bound of Theorem 1.2, we can conclude:

**Corollary 3.1** For all $d, m$, there is an $\varepsilon = 1/m^{\Theta(d)}$ and a $\text{P}$-computable property that is useful against all depth-$d \text{AC}^0[m]$ 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 circuits. Let $\text{CAPP}$ be the problem: given a circuit $C$, output $p \in [0, 1]$ satisfying

$$|\text{Pr}_x[C(x) = 1] - p| < 1/6.$$  

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 $\text{CAPP}$, assuming $\text{P}$ is in quasi-polynomial size ACC.

**Theorem 3.4** Suppose $\text{P}$ has ACC circuits of size $n^{\log n}$. Then there is a constant $c$ such that for infinitely many sizes $s$, $\text{CAPP}$ for size $s$ circuits is computable in nondeterministic $2^{(\log s)^c}$ time.

Theorem 3.4 is a surprisingly strong consequence: given that $\text{NEXP} \not\subseteq \text{ACC}$, one would expect only a $2^{O(n^\varepsilon)}$-time algorithm for $\text{CAPP}$, with $n^\varepsilon$ bits of advice. (Indeed, from the results of IKW [IKW02] one can derive such an algorithm, assuming $\text{P} \subseteq \text{ACC}$.)

Before proving Theorem 3.4, we first extend Theorem 1.2 a little bit. Recall a unary language is a subset of $\{1^n \mid n \in \mathbb{N}\} \subseteq \{0, 1\}^*$. The proof of Theorem 1.2 also has the following consequence:

**Corollary 3.2** If $\text{P}$ has ACC circuits of $n^{\log n}$ size, then for all $d, m$ there is an $\varepsilon$ such that there are unary languages in $\text{NTIME}[2^n]$ without $2^{n^\varepsilon}$-size $d$-depth $\text{AC}^0[m]$ witnesses.
Let \( d \) can approximate the acceptance probability of a size-

\[ \frac{1}{2} \] circuit complexity. Choose a predicate \( \phi \) that lacks such witnesses, and let \( \bar{s} \) be the constant in the pseudorandom generator \( g \). To see this, consider the C

\[ P(1^k) \] nondeterministically generates strings \( Y \) of \( 2^{(g \log s)^{1/2}} \) length that do not have \( s^g = 2^{(n_1)^{c/2}} \) size circuits: as there is an infinite set \( \{ s_i \} \) such that all witnesses to \( 1^{n_1} \) have circuit complexity at least \( 2^{(n_1)^{c/2}} \), there is an infinite set \( \{ s_i \} \) such that \( P(1^{s_i}) \) computes \( \phi_n = (g \log s_i)^{2/\varepsilon} \) and generates \( Y \) which does not have \( s_i \) 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 \( G(Y, \cdot) : \{0, 1\}^{g \log |Y|} \rightarrow \{0, 1\}^s \) which fools circuits of size \( s \). By trying all \( Y \) of length \( \leq 2^{O((\log s)^{1/2})} \) inputs to \( G_Y \), we can approximate the acceptance probability of a size-\( s \) circuit in \( 2^{O((\log s)^{2/\varepsilon})} \) time. As \( \varepsilon \) depended only on \( d^* \) and \( m^* \), which are both constants, we can set \( c = 3/\varepsilon \) to complete the proof.

Now we turn to proving lower bounds for the classes \( NE \cap io-coNE \) and \( NE/1 \cap coNE/1 \). We will need an implication between circuits and Merlin-Arthur simulations that extends Babai-Fortnow-Nisan-Wigderson [BFNW93]:

**Theorem 3.5 (Lemma 8, [MNW99])** Let \( g(n) > 2^n \) and \( s(n) \geq n \) be increasing and time constructible. There is a constant \( c > 1 \) such that \( \text{TIME}[2^{O(n)}] \subseteq \text{SIZE}[s(n)] \Rightarrow \text{TIME}[g(n)] \subseteq \text{MATIME}[s(3 \log g(n))^c] \).

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 \( \text{EXP} \subseteq P/\text{poly} \Rightarrow \text{EXP} = \text{MA} \) ([BFNW93]) combined with a padding argument.

**Reminder of Theorem 1.3** \( NE \cap io-coNE \) and \( NE/1 \cap coNE/1 \) do not have ACC circuits of \( n^{\log n} \) size.
Proof. Suppose NE ∩ io-coNE has $n^{\log n}$-size ACC circuits. We wish to derive a contradiction. Of course the assumption implies that $\text{TIME}[2^{O(n^3)}]$ has $n^{\log n}$-size circuits as well. Applying Theorem 3.5 with $g(n) = 2^{n^2 \log n}$ and $s(n) = n^{\log n}$, we have

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

By Theorem 3.4 and assuming that $\mathbb{P}$ has ACC circuits of size $n^{\log n}$, there is a constant $c$ and a pseudorandom generator 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 $P$, 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 on infinitely many circuit sizes, as well as co-Merlin-Arthur games. Take a $n^{O(\log^3 n)}$-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 $n^{O(n)}$, and Arthur’s string of length $n^{O(\log n)}$, and outputs a bit). Our simulation first guesses Merlin’s string $m$, then runs the PRG which guesses a $Y$ and verifies that $Y$ is a hard function; if the verification fails, we reject. Then 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 $\mathbb{P}$-size circuit agrees with the complement of $L$ on the same list of input lengths $n_{\mathbb{P}}$. Hence there is a constant $d$ such that

$$\text{TIME}[2^{n^2 \log n}] \subseteq \text{MATIME}[n^{O(\log^3 n)}] \subseteq \text{io-NTIME}[n^{\log^d n}].$$

(1)

As $\text{TIME}[2^{n^2 \log n}]$ is closed under complement, an analogous argument (applied to any machine accepting the complement of a given $\text{TIME}[2^{n^2 \log n}]$ language) implies

$$\text{TIME}[2^{n^2 \log n}] \subseteq \text{coMATIME}[n^{O(\log^3 n)}] \subseteq \text{io-coNTIME}[n^{\log^d n}].$$

(2)

Let us look at these simulations more closely. Given a language $L$ in time $2^{n^2 \log n}$ time, by (1) we have a language $L' \in \text{NTIME}[n^{\log^d n}]$ which agrees with $L$ on infinitely many input lengths $n_1, n_2, \ldots$. Since $\text{TIME}[2^{n^2 \log n}]$ is closed under complement, for the language $\overline{L}$ (the complement of $L$) there is also a language $L'' \in \text{NTIME}[n^{\log^d n}]$ with agrees with $L'$ on the same list of input lengths $n_1, n_2, \ldots$. Since $L'$ agrees with the complement of $L''$ on these input lengths on infinitely many input lengths, we have that $L' \in \text{io-coNTIME}[n^{\log^d n}]$, and therefore

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

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

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

But this is a contradiction: for almost every $n$, by simply enumerating all $n^{\log n}$-size circuits and their $2^n$-bit truth tables, we can compute 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.
To prove a lower bound \( \text{NE}/1 \cap \text{coNE}/1 \), we follow precisely the same argument up to (1), and make the following modifications. By using a bit of advice \( y_n \in \{0, 1\} \) to encode whether or not the PRG will be successful for a given input length \( n \), we can simulate an arbitrary \( L \in \text{TIME}[2^{2n\log n}] \) infinitely often in \( \text{NE}/1 \cap \text{coNE}/1 \). In particular, we define a nondeterministic \( N \) and co-nondeterministic \( N' \) which take an advice bit, as follows: if the advice bit is 0, both simulations reject; otherwise, \( N \) attempts to run the Merlin-Arthur simulation of \( L \) (and \( N' \) attempts to Merlin-Arthur simulate \( L \)), respectively as described above. When the advice bits are assigned appropriately on all input lengths, \( N \) and \( N' \) accept a language \( L' \in \text{NE}/1 \cap \text{coNE}/1 \) such that for all \( n \), either \( L' \cap \{0, 1\}^n = \emptyset \) (for input lengths where the advice is set to 0) or \( L' \cap \{0, 1\}^n = L \cap \{0, 1\}^n \) (for infinitely many \( n \)). Therefore

\[
\text{TIME}[2^{n2\log n}] \subseteq \text{io-}(\text{NE}/1 \cap \text{coNE}/1),
\]

and the remainder of the argument concludes as above.

To extend the above lower bound proof to \( \text{NE} \cap \text{coNE} \), it would suffice to prove a stronger time hierarchy theorem for nondeterminism:

**Theorem 3.6** Suppose there is a unary language in \( \text{NTIME}[2^n] \) that is not contained in \( \text{io-coNTIME}[2^{n-n^\varepsilon}] \), for every \( \varepsilon > 0 \). Then \( \text{NE} \cap \text{coNE} \) is not in \( \text{ACC} \).

The proof is straightforward: such a time hierarchy (along with the assumption that \( P \) is in \( \text{ACC} \)) would extend Theorem 3.4 to a nondeterministic \( 2^{(\log s)^c} \) time algorithm for CAPP that works almost everywhere. Such an algorithm could then be applied to replace all “io” complexity classes in the proof of Theorem 1.3 with the standard almost-everywhere versions.

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

**Proof.** (Sketch) Suppose satisfiability for \( C \) circuits of \( n^{O((\log n)^c)} \) size is in \( O(2^{n/n^{10}}) \) time (for all \( c \)), and that \( \text{NE} \cap \text{io-coNE} \) has \( n^{\log n} \) size circuits. By the proof of Theorem 3.4, assuming \( P \) has \( n^{\log n} \) size \( C \) circuits, for all \( \varepsilon > 0 \), we obtain a nondeterministic algorithm \( N \) running in \( 2^{O((\log s)^c)} \) 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 \( \text{NTIME}[2^n] \) that does not have witness circuits of \( n^{\log n} \) size, for every \( c \); this can be used to obtain a nondeterministic algorithm \( N \) as in Theorem 3.4, by setting \( s = n^{O((\log n)^c)} \), solving for \( n = 2^{O((\log s)^{1/(c+1)})} \), then running the nondeterministic algorithm \( N \) in \( 2^{O(n)} \leq 2^{O((\log n)^c)} \) time, where \( \varepsilon \leq 1/(c+1) \).)

By the same argument as in the proof of Theorem 1.3, we obtain

\[
\text{TIME}[2^{n2\log n}] \subseteq \text{MATIME} \cap \text{coMATIME}[n^{O((\log n)^{c+1})}].
\]

By applying algorithm \( N \) to circuits of size \( s = n^{O((\log n)^c)} \) and setting \( \varepsilon \ll 1/4 \), we obtain

\[
(\text{MATIME} \cap \text{coMATIME})[n^{O((\log n)^c)}] \subseteq \text{io-}(\text{NTIME} \cap \text{io-coNTIME})[2^{O(n)}].
\]

But the latter class is in \( \text{io-SIZE}[n^{\log n}] \) by assumption; we obtain a contradiction as in Theorem 1.3.
Similarly as in Theorem 1.3, assuming NE/1 ∩ coNE/1 has \( n^{\log n} \) size circuits, we can conclude

\[
(M \cap \text{coM})[n^{O(\log^3 n)}] \subseteq \text{io-}(\text{N}[2^{O(n)}]/1 \cap \text{coN}[2^{O(n)}]/1) \subset \text{SIZE}[n^{\log n}],
\]

yielding another contradiction.

**Reminder of Theorem 1.5** Suppose we can approximate the acceptance probability of any given \( n^{O(\log n)} \)-size circuit with \( n \) inputs to within \( 1/6 \), for all \( c \), in \( O(2^{n/n^{10}}) \) time (even nondeterministically). Then NE ∩ io-coNE and NE/1 ∩ coNE/1 do not have \( n^{\log n} \)-size circuits.

**Proof.** (Sketch) For all the lower bound arguments given in this section, an algorithm which can approximate the acceptance probability of a given \( n^{O(\log n)} \)-size circuit can be applied in place of a faster SAT algorithm ([Wil10, Wil11, SW13]). That is, from the hypothesis of the theorem we can derive exponential-size witness circuit lower bounds for NEXP (as in Theorem 1.2) and infinitely-often correct pseudorandom generators against general circuits (as in Theorem 3.4). Therefore the proofs of Theorem 1.3 and consequently Theorem 1.4 also carry over under the hypothesis of the theorem. \( \square \)

## 4 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 ZPE = 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). RE = 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 4.1** Let \( L \in \text{ZPE} \). A ZPE predicate for \( L \) is a procedure \( M(x, y) \) that runs in time \( \text{poly}(|y|) \cdot 2^{O(|x|)} \), such that on every \( x \):

- The output of \( M(x, y) \) is in the set \( \{1, 0, ?\} \).
- \( x \in L \implies \Pr_y[M(x, y) \text{ outputs } 1] \geq 2/3 \), and for all \( y \), \( M(x, y) \in \{1, ?\} \).
- \( x \notin L \implies \Pr_y[M(x, y) \text{ outputs } 0] \geq 2/3 \), and for all \( y \), \( M(x, y) \in \{0, ?\} \).

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

That is, \( C \) 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 \( C \) 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 \( C \) seeds, we only require \( x \in L \) to have small circuits \( C_x \) encoding witnesses.

Succinct seeds for zero-error computation are tightly related to uniform natural properties:

**Reminder of Theorem 1.6** Let \( C \) be a polynomial-size typical circuit class. The following are equivalent:

1. There are no \( P \)-natural properties useful (respectively, ae-useful\(^{17}\)) against \( C \)

\(^{16}\)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 be distinguish random functions from easy ones on almost every input length.
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 accepting 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 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 1.6. \((-1) \Rightarrow -2\) Suppose there is a P-natural property \(\ae\)-useful (resp., useful) against \(C\). For some \(c, d\), this is an \(n^c\)-time algorithm \(A\) such that, for almost all \(n\) and all \(k\), \(A\) accepts at least a \(1/2^{dn} n = 1/n^d\) fraction of \(n\)-bit inputs. Moreover, for almost all \(n\) (resp., for infinitely many \(n\)) and all \(k\), \(A\) rejects all \(n\)-bit inputs representing truth tables of \((\log n)^k\)-size \(C\)-circuits.

Let \(b(n)\) denote the \(n\)th string in lexicographical order. Let \(\varepsilon > 0\) be sufficiently small. Define an algorithm \(V\):

\[
V(x, z): \text{If } x \neq b(|z|) \text{ then output } \text{false}. \text{ Partition } z \text{ into } t = |z|^{1-\varepsilon/d} \text{ strings } z_1, \ldots, z_t \text{ each of length } |z|^{\varepsilon/d}. \text{ If } A(z_i) \text{ accepts for some } i, \text{ then output } 1; \text{ else, output } \text{false}.
\]

We claim \(V\) is a ZPE predicate for \(L = \{0, 1\}^*\). Consider a \(z\) chosen at random. All the \(z_i\) are independent random variables, and \(A\) accepts at least \(1/|z_i|\) strings of that length. So the probability that all \(z_i\) are among the \((1-1/|z|^{\varepsilon})\) fraction of strings of length \(|z|^{\varepsilon/d}\) rejected by \(A\), is at most \((1-1/|z|^{\varepsilon})|z|^{1-\varepsilon/d} \leq e^{-|z|^{1-\varepsilon/d}}\). For small enough \(\varepsilon\), this quantity is less than \(2/3\), so \(V\) accepts a random \(z\) with at least \(2/3\) probability.

By construction, \(V\) accepts \((x, z)\) precisely when \(x = b(|z|)\) and some \(z_i\) is accepted by \(A\). Hence for almost all input lengths \(|x|\) (resp., infinitely many), when \(V\) accepts \(x\) via witness \(w\), some \(z_i\) has \(C\)-circuit complexity at least \((\log(|z|^{\varepsilon/d}))^k \geq \Omega((\log |z|))\). Therefore by Lemma 2.1, \(z\) itself has \(C\)-circuit complexity at least \(\Omega((\log |z|)^k - (\log |z|)^{1+o(1)})\). As this holds for every \(k\), the predicate \(V\) does not have \(C\) seeds almost everywhere (respectively, almost everywhere).

\((-2) \Rightarrow -1\) Suppose there is a ZPE predicate \(V\) that does not have \(C\) 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 an input \(x\) of length \(n_i\) such that, for every \(r\) satisfying \(V(x, r) \neq \text{false}\) (over random \(r\) of length \(2^{m_i}\)), \(r\) has \(C\)-circuit complexity at least \((cn_i)^k\). (Note \(c\) depends on \(V\) only.)

Define a new predicate \(V'(x, r)\) that takes random \(r\) of length \(2^{\ell + cn_i}\) where \(\ell\) is the smallest integer such that \(2\ell \leq 2^\ell\), partitions \(r\) into \(2^{\ell}\) strings \(\{r_i\}\) of length \(2^{m_i}\), each, and accepts if and only if if \(V(x, r_i) \neq \text{false}\) for some \(i\). Any \(r\) with this property does not have circuits of size \(n_i^k\), due to Proposition 1 and the fact that such an \(r\) contains a substring \(r_i\) with circuit complexity at least \(n_i^k\). By standard probabilistic arguments, it is likely that \(r\) encodes a hitting set, i.e., \(Pr_r[(\exists x \in \{0, 1\}^{n_i})(\forall i) V(x, r_i) = \text{false}] < 1/3\). Therefore, a randomly chosen \(r\) of length \(2^{(\ell+1)n_i}\) is accepted by \(V\), with probability at least \(2/3\).

We now define an algorithm \(A\) that gives a P-natural property. \(A\) takes \(R\) of length \(N\) as input, computes the largest \(n_i\) such that \(N \geq 2^{\ell + cn_i}\), sets \(r\) to be the first \(2^{\ell + cn_i}\) bits of \(R\), and partitions \(r\) into \(r_i\)’s as above. Then \(A\) checks for all \(x\) of length \(n_i\) that some \(V(x, r_i)\) does not output \(\text{false}\). This algorithm \(A\) runs in \(poly(N)\) time and accepts at least \(1/2\) of its inputs. Furthermore, \(A\) must reject all strings \(R\) with \(C\)-circuit complexity at most \(O(n_i^k)\) (i.e., \(O((\log N)^k))\), almost everywhere (resp., infinitely often), because if \(R\) had such circuits, then all \(r_i\) would as well (by Proposition 1). This occurs for every \(k\), so \(A\) is a P-natural property useful against polynomial-size \(C\) circuits. \(\square\)
To prove a related result for RE predicates, we first need a little more notation. Let $V$ be an $\text{RTIME}[2^{kn}]$ predicate accepting a language $L$. For a given input length $n$, a set $S_n \subseteq \{0,1\}^{2kn}$ 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 $V(x,y)$ 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 also consider another relaxation of naturalness: We say that a property $\mathcal{P}$ is io-$\mathcal{P}$-natural against polynomial-size $\mathcal{C}$ provided that, for every $k$ and infinitely many $n$, $\mathcal{P}$ accepts at least a $1/\text{poly}(n)$ fraction of $n$-bit inputs, and $\mathcal{P}$ rejects all $n$-bit inputs representing functions computable with $((\log n)^k + k)$-size $\mathcal{C}$-circuits.\footnote{As usual, if $\mathcal{C}$ is also characterized by a depth $d$ or modulus constraint $m$, those $d$ and $m$ are quantified alongside $k$.} In the usual notion of natural proofs, largeness holds almost everywhere; here, that is not required. We can relate succinctly encoded hitting sets with natural properties as follows:

**Theorem 4.1** Suppose for all $c$, $\text{RTIME}[2^{O(n)}]$ does not have $O(n^2)$-size hitting sets encoded by $n^c$-size circuits. Then for all $c$, there is an io-$\mathcal{P}$-natural property useful against $n^c$-size circuits.

**Proof.** The hypothesis says that for every $c$, there is an $\text{RTIME}[2^{O(n)}]$ 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 a $O(n^2)$-size circuit family.

We can get an io-$\mathcal{P}$-natural property computable in $\mathcal{P}$ with $O(\log n)$ bits of advice, as follows. Given an input string $Y$ of length $N = 2^{kn+2\log n}$, the advice string $a$ encodes the number of inputs of length $n$ in $L(V_c)$.

Our polynomial-time algorithm partitions $Y$ into $y_1, \ldots, y_{2\log n}$ of equal length, and counts the number of $x$ of length $n \leq (\log N)/k$ such that $V_c(x,y_i)$ accepts for some $i$. If this number equals the advice $a$, then reject else accept. For infinitely many $n$, this procedure (with the appropriate advice) accepts a random string with high probability, and rejects strings encoded by $n^c$-size circuit families, by assumption.

Now, given an io-$\mathcal{P}/(\log n)$-natural property $A$ against $n^c$-size circuits, we can convert it into an io-$\mathcal{P}$-natural property. For each natural number $n$ we associate the interval $I_n = [n^2, (n+1)^2 - 1]$; note that the collection of $I_n$ partitions $\mathbb{N}$. Given input $X$ of length $m$, our new property $A'$ determines $I_n$ such that $m \in I_n$, and computes $a = m - n^2$. Since $a \in \{0, \ldots, 2n\}$, $a$ can be treated as an advice string of length $(\log n)$ for $n$-bit inputs; $A'$ takes the first $n$ bits of $X$, and runs $A(x,a)$.

For infinitely many input lengths $n_i$, $A$ (equipped with the appropriate advice $a_i$) is simultaneously large and useful against $n^c$-size circuits. The above shows that each such $n_i$ has an associated length $m_i$ such that the advice $a_i$ can be correctly extracted from $m_i$, and $A(x,a_i)$ is executed. Hence on these $m_i$, the property $A'$ is both large and useful. Notice that, since the input has increased by a square $(m_i = \Theta(n_i^2))$, the strings of length $m_i$ define functions on only twice as many inputs as $n_i$. Therefore, when $A(x,a_i)$ accepts (hence $x$ has circuit complexity at least $(\log n_i)^5$), by Lemma 2.1 we can infer that the original input $X$ defines a 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 property $A'$ is useful against circuits of size $(n/4)^c$. As this condition holds for every $c$, the theorem follows.

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

### 4.1 Unconditional Mild Derandomizations

We are now prepared to give some unconditionally-true derandomization results. The first one is:
Reminder of Theorem 1.7 Either $\text{RTIME}[2^{O(n)}] \subset \text{SIZE}[n^c]$ for some $c$, or $\text{BPP} \subset \text{io-ZPTIME}[2^{n^c}]/n^\varepsilon$ for all $\varepsilon > 0$.

To give intuition for the proof, we compare with the “easy witness” method of Kabanets [Kab01], which shows that $\text{RP}$ can be pseudo-simulated in $\text{io-ZPTIME}[2^{n^c}]$ (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 $\text{RP}$ predicate, try all $n^\varepsilon$-size circuits and check if any encode a good seed for the predicate. If this always happens (against all efficient adversaries), then we can simulate $\text{RP}$ 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 $n^\varepsilon$-size circuits; this hard function can be used to derandomize $\text{BPP}$.

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 4.1) that contain seeds for all inputs of a given length, encoded by $n^\varepsilon$-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 $O(n^c)$ bits of advice to simulate $\text{RP}$ deterministically. Otherwise, we apply Theorem 4.1 to obtain an io-P-natural property which can be used (by randomly guessing a hard function) to simulate $\text{BPP}$ 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 enumerate over (short) inputs in our natural property.

Proof of Theorem 1.7. First, suppose there is a $c \geq 2$ so that for every $\text{RTIME}[2^{O(n)}]$ 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 $2^{\log n}$ 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 $\text{RTIME}[2^{O(n)}] \subset \text{TIME}[2^{O(n^c)}/n^\varepsilon]$, 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 $\text{RTIME}[2^{O(n)}]$ 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 4.1 says that for all $c$, we can extract an io-P-natural property $A_c$ useful against $n^c$ size circuits, for all $c$. In particular, the proof of Theorem 4.1 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 $\text{BPTIME}[O(n^k)]$ (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 $O(n^c)$, specifying an input length $m = 2^\Theta(n^c)$. 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 $m = 2^\Theta(n^c)$ bit string with circuit complexity at least $(n^c)^c \geq n^{9k}$.

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 $\varepsilon$ and $k$). Each call to $G_Y$ takes poly$(|Y|) \leq 2^{O(n^c)}$ time. Trying all $|Y|^d \leq 2^{O(n^c)}$ seeds to $G_Y$, we can approximate the acceptance prob-
ability of a $n^{3k}$-size circuit simulating any $\text{BPTIME}[O(n^k)]$ language on $n$-bit inputs, thereby determining acceptance/rejection of any $n$-bit input.

Now we have either (1) $\text{RTIME}[2^{O(n)}] \subseteq \text{TIME}[2^{O(n)}]/n^c$ for some $c$, or (2) $\text{BPP} \subseteq \text{io-ZPTIME}[2^{n^c}]/n^c$ for all $\varepsilon > 0$. To complete the proof, we recall that Babai-Fortnow-Nisan-Wigderson [BFNW93] proved that if $\text{BPP} \not\subseteq \text{io-SUBEXP}$ then $\text{EXP} \subset \text{P/poly}$. Therefore, if case (2) does not hold, the first case can be improved: using a complete language for $\text{E}$, we infer from $\text{EXP} \subset \text{P/poly}$ that $\text{TIME}[2^{O(n)}] \subset \text{SIZE}[n^c]$ for some $c$, so $\text{RTIME}[2^{O(n)}] \subset \text{SIZE}[n^c]$ for some constant $c$. □

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

**Proof.** By Theorem 1.7, there are two cases: (1) $\text{RTIME}[2^{O(n)}] \subset \text{SIZE}[n^c]$ for some $c$, or (2) $\text{BPP} \subset \text{io-ZPTIME}[2^{n^c}]/n^c$ for all $\varepsilon$. In case (1), $\text{RP} \subseteq \text{RTIME}[2^{O(n)}] \subseteq \text{TIME}[n^c]/n^c$. In case (2), $\text{RP} \subseteq \text{BPP} \subseteq \text{io-ZPTIME}[2^{n^c}]/n^c$.

The simulation can be ported over to Arthur-Merlin games. Recall that a language $L$ is in $\text{AM}$ if and only if there is a $k$ and deterministic algorithm $V(x, y, z)$ running in time $|x|^k$ with the 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 $c \geq 1$, $\text{AM} \subseteq \text{io-}\Sigma_2\text{SUBEXP}/n^c$.

The problem of finding nontrivial relationships between $\text{AM}$ and $\Sigma_2\text{P}$ has been open for some time [GSTS03, AvM12].

**Proof.** (Sketch) The proof is similar to Theorem 1.7, with the following modifications. Instead of hitting sets for $\text{RP}$ computations, we consider hitting sets for $\text{AM}$ computations: a $\text{poly}(n)$-size set $S$ of $n^k$-bit strings that can replace the role of $y$ (Arthur) in the $\text{AM}$ 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 $\text{AM}$ 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 $L \in \text{AM}$ and verifiers $V_e$ for $L$, there is an $n^c$-size SAT-oracle circuit family encoding hitting sets for $V_e$, on almost all input lengths $n$.** In this case, we can put $\text{AM}$ in the class $\text{P}^{\text{NP}/\tilde{O}(n^c)}$: 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 $\text{P}^{\text{NP}}$, producing the hitting set, then use the hitting set and the $\text{NP}$ oracle to simulate the $\text{AM}$ computation.

2. **For all $c$, there is some verifier $V$ of some $\text{AM}$ 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 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 $n = m^c$ (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 which of the $2^n$ strings of length $n$ are accepted, and which are rejected (considering 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 \( \Sigma_2 \) computation running in time \( 2^{O(n)} \leq 2^{O(m^c)} \), which (when given the appropriate advice of length \( O(m^c) \)) correctly determines that at least some string \( Y \) has SAT-oracle circuit complexity at least \( (m^c)^{10k/\epsilon} \geq n^{10k} \), on infinitely many input lengths.

Now suppose we want to simulate an AM computation on inputs of length \( m \) running in time \( m^k \). Then we can simulate the AM computation in \( \text{io-S}_{2\Sigma} \text{TIME}[2^{O(n^c)}]/O(n^c) \), as follows: we guess a string \( Y \) and apply known results in derandomization [KvM02] that use \( Y \) to simulate AM computations in NSUBEXP. Then we apply the aforementioned \( \Sigma_2 \) 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. \( \square \)

It is 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-Santhanam [BFS09]) to separate \( \Sigma_2 \text{EXP} \) from AM.

Another application of Theorem 1.7 is an unexpected equivalence between the infamous separation problem \( \text{NEXP} \neq \text{BPP} \) and zero-error simulations of BPP. We need one more definition: Heuristic \( \mathcal{C} \) is the class of languages \( L \) such that there is a \( L' \in \mathcal{C} \) 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 \( \text{io-Heuristic} \mathcal{C} \) is defined analogously.

**Reminder of Theorem 1.8** \( \text{NEXP} \neq \text{BPP} \) if and only if for all \( \epsilon > 0 \), \( \text{BPP} \subseteq \text{io-HeuristicZPTIME}[2^{n^c}]/n^c \).

This extends an amazing result of Impagliazzo and Wigderson [IW01] that \( \text{EXP} \neq \text{BPP} \) if and only if for all \( \epsilon > 0 \), \( \text{BPP} \subseteq \text{io-HeuristicTIME}[2^{n^c}] \). It is interesting that \( \text{NEXP} \) versus \( \text{BPP} \), a problem concerning the power of nondeterminism, is equivalent to a statement about derandomization of \( \text{BPP} \) without nondeterminism. Theorem 1.8 should also be contrasted with the \( \text{NEXP} \) versus \( \text{P/poly} \) equivalence of IKW [IKW02]: \( \text{NEXP} \not\subseteq \text{P/poly} \) if and only if \( \text{MA} \subseteq \text{io-NTIME}[2^{n^c}]/n^c \), for all \( \epsilon > 0 \).

**Proof of Theorem 1.8.** First, assume \( \text{BPP} \) is not in \( \text{io-HeuristicZPTIME}[2^{n^c}]/n^c \) for some \( \epsilon > 0 \). Then \( \text{BPP} \not\subseteq \text{io-ZPTIME}[2^{n^c}]/n^c \), so by Theorem 1.7 we have that \( \text{RTIME}[2^{O(n)}] \) has size-\( n^c \) seeds, which implies \( \text{REXP} = \text{EXP} \). The result also implies that \( \text{BPP} \) is not in \( \text{io-HeuristicTIME}[2^{n^c}] \), so by Impagliazzo and Wigderson [IW01] we have \( \text{EXP} = \text{BPP} \). Therefore \( \text{REXP} = \text{BPP} \). But this implies \( \text{NP} \subseteq \text{BPP} \), so by Ko’s theorem [Ko82] we have \( \text{NP} = \text{RP} \). Finally, by padding, \( \text{NEXP} = \text{REXP} = \text{BPP} \).

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

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

The last inclusion in (3) can be proved as follows: let \( L \in \bigcap_{\epsilon > 0} \text{io-HeuristicNTIME}[2^{n^c}]/n^c \) be arbitrary, and let \( L' \in \bigcap_{\epsilon > 0} \text{NTIME}[2^{n^c}]/n^c \) be such that \((L \cap \{0, 1\}^n) \Delta (L' \cap \{0, 1\}^n) \leq 2^n/n \) on infinitely many \( n \).

\[19\textbf{N.B.} \text{ This is a weaker definition than usually stated, but it will suffice for our purposes.}\]
This means that, for any \( \varepsilon \), \( L' \) can be solved using a collection of nondeterministic machines \( \{M_n\} \) running in \( 2^{n^\varepsilon} \) 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 \( 2^{O(n^\varepsilon)} \) time algorithm for SUCCINTHALTING on the input \( (M_n, x, b(2^{n^\varepsilon})) \), 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 \( 2^{O(n^\varepsilon)} \) time.

Finally, we prove that the above inclusion (3) is false, by direct diagonalization. That is, we can find an \( L \in \text{EXP} \) such that \( L \not\in \text{i.o-HeuristicTIME}[2^{O(n^\varepsilon)}]/n^{1/2} \). Let \( \{M_i\} \) be a list of all \( 2^{n^\varepsilon} \) time machines. We will give a \( 2^{2n^{\varepsilon+1}} \)-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 the 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 \( x_i \) iff 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 \( B \cdot n \cdot 2^{O(n^\epsilon)} \leq O(2^{2n^{\varepsilon+1}}) \) time. For every block and every \( i \), we have \( |S_i| \leq |S_{i-1}|/2 \). Since \( |S_0| = 2^{n^{1/2}}-n \), this implies that \( |S_B| = 0 \). So for every block, each 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, \( L(M) \in \text{EXP} \) but \( L(M) \not\in \text{i.o-HeuristicTIME}[2^{O(n^\varepsilon)}]/n^{1/2} \).

\[ \square \]

### 4.2 Unconditional Derandomization of Natural Properties

Finally, 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:

**Reminder of Theorem 1.9** *If there exists a RP-natural property \( P \) useful against a 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 proofs in this paper: we split the input string \( T \) into small substrings, and 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 \( T_1, \ldots, T_{n^{1/k}} \) of length at most \( n^{1/k} \) each, and 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 $T = T_1 \cdots T_{n^{1/k}}$ (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

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

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. \hspace{1cm} \square

### 5 Conclusion

Ketan Mulmuley has recently suggested that “$\mathbb{P} \neq \mathbb{NP}$ because $\mathbb{P}$ is big, not because $\mathbb{P}$ is small” [Mul11]. That is to say, the power of efficient computation is the true reason we can prove lower bounds. The equivalence in Theorem 1.1 between NEXP lower bounds and $\mathbb{P}$-time 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?** 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 no ACC circuit of polynomial size agrees with these functions on $1/2 + 1/poly(n)$ of the inputs? Such predicates could be used to yield unconditional derandomized simulations of ACC circuits (using nondeterminism). The primary technical impediment seems to be that we do not think ACC can compute the Majority function, which appears to be necessary for hardness amplification (see [SV10]). But this should make it easier to prove lower bounds against ACC, not harder!

- **Equivalences for non-uniform natural properties?** In this paper, we have mainly studied natural properties decidable by uniform algorithms; however, the more general notion of $\mathbb{P}/\text{poly}$-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 “NEXP $\not\subset \mathbb{P}/\text{poly}$” and “integer factorization is not in subexponential time.” The
first asserts nontrivial efficient algorithms for recognizing some hard Boolean functions (as seen in this pa-
per); the second denies efficient algorithms for recognizing a non-negligible fraction of hard Boolean func-
tions [KC00, ABK+06]. An equivalence involving $\text{NP} \not\subset \text{P/poly}$ could yield more powerful algorithms for
recognizing hardness.

6 Acknowledgments

I thank Steven Rudich and Rahul Santhanam for useful discussions, and Amir Abboud for many comments
and corrections on the manuscript. I also thank Russell Impagliazzo and Igor Carboni Oliveira for several
useful insights, and Emanuele Viola for a pointer to his paper with Eric Miles.

References

[Aar07] Scott Aaronson. Shtetl-Optimized, page http://www.scottaaronson.com/blog/?p=240, May
2007.

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

[ABK+06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronnebu-
ger. Power from random strings. SIAM J. Comput., 35(6):1467–1493, 2006.

[All01] Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov com-
plicity. In FSTTCS, Springer LNCS 2245, pages 1–15, 2001.

[AvM12] Baris Aydinlioglu and Dieter van Melkebeek. Nondeterministic circuit lower bounds from
mildly de-randomizing Arthur-Merlin games. In IEEE Conf. Computational Complexity, pages
269–279, 2012.

[Bar02] Boaz Barak. A probabilistic-time hierarchy theorem for “Slightly Non-uniform” algorithms.
Lecture Notes in Computer Science, 2483:194–208, 2002.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time
simulations unless EXPTIME has publishable proofs. Computational Complexity, 3(4):307–
318, 1993.

[BFS09] Harry Buhrman, Lance Fortnow, and Rahul Santhanam. Unconditional lower bounds against
advice. In ICALP (Vol. 1), pages 195–209, 2009.

[BT94] Richard Beigel and Jun Tarui. On ACC. Computational Complexity, pages 350–366, 1994.

[CFL85] Ashok K. Chandra, Steven Fortune, and Richard J. Lipton. Unbounded fan-in circuits and
associative functions. JCSS, 30(2):222–234, 1985.

[Cho11] Timothy Y. Chow. Almost-natural proofs. JCSS, 77:728–737, 2011.

[FST05] Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Hierarchies for semantic classes. In
STOC, pages 348–355, 2005.
[GSTS03] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness versus randomness tradeoffs for arthur-merlin games. *Computational Complexity*, 12(3-4):85–130, 2003.

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

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: derandomization under a uniform assumption. *JCSS*, 63(4):672–688, 2001.

[JS12] Maurice J. Jansen and Rahul Santhanam. Stronger lower bounds and randomness-hardness trade-offs using associated algebraic complexity classes. In *STACS*, pages 519–530, 2012.

[Kab01] Valentine Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *JCSS*, 63(2):236–252, 2001.

[KC00] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *STOC*, pages 73–79, 2000.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KL01] Matthias Krause and Stefan Lucks. Pseudorandom functions in TC0 and cryptographic limitations to proving lower bounds. *Computational Complexity*, 10:297–313, 2001.

[Ko82] Ker-I Ko. Some observations on the probabilistic algorithms and np-hard problems. *IPL*, 14(1):39–43, 1982.

[KvM02] Adam Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

[MNW99] Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *COCOON, Springer LNCS 1627*, pages 210–220, 1999.

[Mul11] Ketan Mulmuley. Private communication, 2011.

[MV12] Eric Miles and Emanuele Viola. Substitution-permutation networks, pseudorandom functions, and natural proofs. In *CRYPTO*, pages 68–85. Springer LNCS, 2012.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *JACM*, 51(2):231–262, 2004.

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

[RR97] Alexander Razborov and Steven Rudich. Natural proofs. *JCSS*, 55(1):24–35, 1997.

[Rud97] Steven Rudich. Super-bits, demi-bits, and $N\tilde{P}$/qpoly-natural proofs. In *RANDOM*, pages 85–93. Springer LNCS, 1997.

[SV10] Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010.
[SW13] Rahul Santhanam and Ryan Williams. On medium-uniformity and circuit lower bounds. In *IEEE Conf. Computational Complexity*, pages 15–23, 2013.

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

[Wil11] Ryan Williams. Non-uniform ACC circuit lower bounds. In *IEEE Conf. Computational Complexity*, pages 115–125, 2011.

[Wil10] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal on Computing*, 42(3):1218–1244, 2013. See also STOC’10.

[Ž83] Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, October 1983.