Easy Witnesses and Hard Circuit Lower Bounds

Shawn Ong

December 7, 2018

1 Introduction and Definitions

One major research area of modern complexity theory is derandomization, that is, determining whether probabilistic algorithms can be simulated by deterministic ones without significantly increasing computational bounds on running time or space efficiency. In particular, though it is suspected that $\mathsf{BPP} = \mathsf{P}$, it is still not even known whether $\mathsf{BPP} \neq \mathsf{NEXP}$. The primary result of [4] is a proof that $\mathsf{NEXP} \subset \mathsf{P/poly}$ if and only if $\mathsf{NEXP} = \mathsf{MA}$; unless stated otherwise, any information referenced below comes from [4].

Another major result of the paper is a set of various downward closure results. Many complexity theory results are easily seen to "translate upwards" – for example, if $\mathsf{NP} = \mathsf{P}$, then it can be shown that $\mathsf{NEXP} = \mathsf{EXP}$. Such proofs often utilize a padding argument; suppose, for example, that $L \in \mathsf{NEXP}$, that is, there is a nondeterministic algorithm deciding L which runs in $2^{n^{O(1)}}$ time. We can construct a padded version of L by extending each string of length n to one of length 2^n, by appending $0^{2^n - n}$. There exists a nondeterministic poly-time algorithm to decide this language; namely, check whether the input is of this form, and if so, run the algorithm to decide L on the appropriate prefix of the input. Then, under the assumption that $\mathsf{NP} = \mathsf{P}$, there exists a deterministic poly-time algorithm for the padded language, which can be converted into a deterministic exponential-time algorithm for L, by converting the input to the padded form and running the deterministic algorithm for the padded language.

The paper demonstrates that, for certain classes, proving a relationship between two classes can also translate downwards. In particular, it provides evidence that proving $\mathsf{EXP} \neq \mathsf{BPP}$ is hard, as this is true iff $\mathsf{EE} \neq \mathsf{BPE}$ (the second claim implies the first by a padding argument; the paper demonstrates that the reverse direction holds as well).

We provide definitions for some of the machinery which will be relevant in the remainder of the paper. We will assume familiarity with the standard complexity classes P, NP, BPP, ZPP, RP, E, EXP, NE, NEXP, as well as $\mathsf{SPACE}(s)$, $\mathsf{DTIME}(s)$, and $\mathsf{NTIME}(s)$ where s is any function $\mathbb{N} \to \mathbb{R}$. Given a Turing machine M, or equivalently, any decision procedure, we define $L(M)$ to be the corresponding language. For any language L, define the characteristic function $\chi_L : \{0,1\}^* \to \{0,1\}$ such that $\chi_L(x) = 1$ iff $x \in L$. In addition, we will define some additional complexity classes:

Definition 1.1. *Let the class* MA *consist of all languages* $L \subseteq \{0,1\}^*$ *such that there exists a polynomial-time decidable predicate* $R(x, y, z)$ *(i.e. a function from binary strings* $x, y, z \mapsto$

$\{0,1\}$*) and a constant* c *such that, for every* $x \in \{0,1\}^n$*, we have:*

$$x \in L \implies \exists y \in \{0,1\}^{n^c} \text{ such that } \Pr_{z \in \{0,1\}^{n^c}}[R(x,y,z)=1] \geq \frac{2}{3}$$
$$x \notin L \implies \forall y \in \{0,1\}^{n^c} \text{ such that } \Pr_{z \in \{0,1\}^{n^c}}[R(x,y,z)=1] \leq \frac{1}{3}$$

Intuitively, this is a "nondeterministic version of BPP" [4]; in addition to the input x, we are allowed to nondeterministically receive advice y which gives a high probability (over z) of acceptance for $x \in L$; if $x \notin L$, then there is a low probability of acceptance for any possible advice.

Definition 1.2. *Let the class* AM *consist of all languages* $L \subseteq \{0,1\}^*$ *such that there exists a polynomial-time decidable predicate* $R(x,y,z)$ *(i.e. a function from binary strings* $x,y,z \mapsto \{0,1\}$*) and a constant* c *such that, for every* $x \in \{0,1\}^n$*, we have:*

$$x \in L \implies \Pr_{z \in \{0,1\}^{n^c}}[\exists y \in \{0,1\}^{n^c} \text{ such that } R(x,y,z)=1] \geq \frac{2}{3}$$
$$x \notin L \implies \Pr_{z \in \{0,1\}^{n^c}}[\exists y \in \{0,1\}^{n^c} \text{ such that } R(x,y,z)=1] \leq \frac{1}{3}$$

In contrast to MA, AM can be viewed as a "probabilistic version of NP" [4]. Given input $x \in L$, there exists with high probability a witness of R holding; on the other hand, for $x \notin L$, the probability of such a witness existing is low.

Definition 1.3. *For any function* $s : \mathbb{N} \to \mathbb{N}$*, define the nonuniform complexity class* $\mathsf{SIZE}(s)$ *to consist of all families* $\{f_n\}_{n \geq 0}$*, of* n*-variable boolean functions such that, for sufficiently large* n*,* f_n *can be computed by a Boolean circuit of size* $s(n)$*. We will similarly define* $\mathsf{SIZE}^A(s)$ *to contain the families of* n*-variable Boolean functions computable by oracle circuits of size* $s(n)$ *with* A*-gates, where* A *is an arbitrary oracle.*

Definition 1.4. *Let* C *be any complexity class over* Σ *and let* $t : \mathbb{N} \to \mathbb{N}$*. We define* C/t *to consist of all languages* L *for which there exists a language* $M \in \mathsf{C}$ *and family of strings* $\{y_n\}_{n \geq 0}$*, where* $y_n \in \Sigma^{O(t(n))}$*, such that for all* $x \in \Sigma^n$*, we have:*

$$x \in L \iff (x, y_n) \in M$$

Intuitively, C/t consists of all languages which can be decided by a Turing machine of complexity C, using advice of size up to t.

Definition 1.5. *For any complexity class* C *over alphabet* Σ*, define:*

$$\mathsf{io\text{-}C} := \{L \subseteq \Sigma^* \mid \exists M \in \mathsf{C}, L \cap \Sigma^n = M \cap \sigma^n \text{ infinitely often}\}$$

One key idea is the "easy witness" method, originally due to Kabanets [7], in which a user searches for an object satisfying specific properties (such as a witness in a search problem) among objects which have concise descriptions (say, logarithmic in the input size). Since there are few such objects, this search is more efficient than a standard brute-force. If the search fails, then the result is a "hardness test," that is, an efficient algorithm to generate strings that do not have small descriptions (as such a string would have been identified by the brute force search). The hardness test allows for nondeterministic generation of the truth table of a hard Boolean function, to which known hardness-randomness tradeoffs can be applied, allowing the table to be used as a source of pseudorandomness.

In [4], the easy witness method is used to reduce the search space for a witness of an NEXP problem, by considering the NEXP-complete problem Succinct-SAT. Succinct-SAT is the problem of deciding whether a propositional formula is satisfiable, given a Boolean circuit encoding the formula. Intuitively, the easy witness reduction in this problem shows that any satisfiable propositional formula described by a small Boolean circuit has a satisfying assignment (the easy witness) that can be described by a small Boolean circuit (where "small" is defined by a bound on circuit complexity).

The following definitions will prove useful to formally express the notion of easy witnesses:
Definition 1.6. *Let $R(x,y), x \in \{0,1\}^*, y \in \{0,1\}^*$ be a poly-time decidable relation and let $l : \mathbb{N} \to \mathbb{N}$. Define $f_R(x) : \{0,1\}^* \to \{0,1\}$ by:*

$$f_R(x) = 1 \text{ iff } \exists y \in \{0,1\}^{l(|x|)} \text{ such that } R(x,y) \text{ holds.}$$

This turns the search problem of finding a witness y for an input x so that $R(x,y)$ holds into a decision problem. For the purposes of [4], take $l(n) = 2^n$, so that $f_R(x)$ is the characteristic function of a language in NE.
Definition 1.7. *Let $T_{A,s}(n)$ be the set of truth tables of n-variable Boolean functions described by A-oracle circuits of size $s(n)$. Define: $\hat{f}_{R,A,s}(x) : \{0,1\}^* \to \{0,1\}$*

$$\hat{f}_{R,A,s}(x) = 1 \text{ iff } \exists y \in T_{A,s}(|x|) \text{ such that } R(x,y) \text{ holds.}$$

This defines a function indicating the inputs x for which easy witnesses y exist for $R(x,y)$. From these definitions, we have $f_R = \hat{f}_{R,A,s}$ iff every x which allows a satisfying assignment for R has such an assignment with an easy witness.

Additionally, for any fixed oracle $A \in \mathsf{EXP}$ and for any function $s : \mathbb{N} \to \mathbb{N}$, the set $T_{A,s}(n)$ can be enumerated in deterministic $2^{\mathsf{poly}(s(n))}$ time. A circuit of size $s(n)$ can query an oracle A on at most $2^{s(n)}$ and A can be simulated by a deterministic Turing Machine in 2^{n^d} time for some $d \in \mathbb{N}$. So the truth table of the Boolean function corresponding to the circuit can be found in $2^n \mathsf{poly}(s(n)) 2^{\mathsf{poly}(s(n))}$, and there are only $2^{O(s \log s)}$ possible A-oracle circuits of size s.

In particular, $\hat{f}_{R,A,s}$ can be deterministically computed in $2^{\mathsf{poly}(s(n))}$ time, which is strictly better than the brute force $2^{O(n)}2^{2^n}$ runtime to compute f_R. Therefore, if $f_R = \hat{f}_{R,A,s}$, a nontrivial deterministic algorithm for f_R exists. On the other hand, we will see that if $f_R \neq \hat{f}_{R,A,s}$, this yields a nondeterministic $\mathsf{poly}(2^n)$-time algorithm generating truth tables of n-variable Boolean functions of high A-oracle circuit complexity.

We will also reproduce the relevant definitions for pseudorandom generators. A generator is simply a function $G : \{0,1\}^* \to \{0,1\}^*$ mapping $\{0,1\}^{l(n)} \to \{0,1\}^n$, where $l : \mathbb{N} \to \mathbb{N}$.
Definition 1.8. *For oracle A, a generator $G : \{0,1\}^{l(n)} \to \{0,1\}^n$ is $\mathsf{SIZE}^A(n)$-pseudorandom if, given any n-input Boolean circuit C of size n with A-oracle gates:*

$$\left| \Pr_{x \in \{0,1\}^{l(n)}}[C(G(x)) = 1] - \Pr_{y \in \{0,1\}^n}[C(y) = 1] \right| \leq \frac{1}{n}$$

Informally, such a function is "good enough" to fool any such circuit within reasonable probability; this allows for simulating randomness using a small seed (of size $l(n)$), which may be iterated over by brute force without significantly increasing runtime, for sufficiently small $l(n)$. Additionally, call a generator *quick* if it can be deterministically computed in time $2^{O(l(n))}$.

2 Main Results

The main proof uses a number of separation results, the first of which is as follows:
Theorem 2.1. *For any constant* $c \in \mathbb{N}$, $\mathsf{EXP} \not\subset \mathsf{io\text{-}SIZE}(n^c)$.

The argument here is by diagonalization. For sufficiently large n, there is an n-variable Boolean function of circuit complexity $2n^c$ with no equivalent circuit of size n^c. Then brute force search over circuits of size $2n^c$ to find the first such circuit, in lexicographic order, can be done in deterministic exponential time. The language formed by taking all strings corresponding to these functions over all n is then in EXP but not $\mathsf{io\text{-}SIZE}(n^c)$.
Theorem 2.2. *For any fixed* $c \in \mathbb{N}$, $\mathsf{EXP} \not\subseteq \mathsf{io\text{-}[DTIME}(2^{n^c})/n^c]$.

Let $n \in \mathbb{N}$ be sufficiently large (so that $2^n \leq n^c$) and let S_n be all truth tables generated by a n-variable Boolean function decided by a deterministic Turing machine of description of size n running in 2^{n^c} time, using advice of length at most n^c. In particular, $|S_n| \leq 2^{2n^c}$. Construct a truth table $t = t_1, \ldots, t_{2^n}$ of an n-variable Boolean function inductively, such that t_k disagrees with the majority of elements of S_n which agree with $t_1, \ldots, t_{k-1}$; take all remaining bits of t to be 0. Let L be the language such that for $x \in \{0,1\}^n$, $x \in L$ iff $t_x = 1$. $L \in \mathsf{EXP}$ by construction, but $L \notin \mathsf{io\text{-}[DTIME}(2^{n^c})/n^c]$, as no Boolean function can agree with L if S_n^k is empty.

The proof also relies on lemmas based on the existence of universal Turing machines, that is, Turing machines that can simulate any Turing machine on any given input.
Lemma 2.3. *If* $\mathsf{NEXP} \subset \mathsf{P/poly}$, *then* $\mathsf{NTIME}(2^n)/n \subset \mathsf{SIZE}(\mathsf{poly}(n))$.

For any $L \in \mathsf{NTIME}(2^n)/n$ there exists $M \in \mathsf{NTIME}(2^n)$ and a sequence of advice strings $\{y_n\}_{n\geq 0}$, $y_n \in \{0,1\}^n$ such that for all $x \in \{0,1\}^n$, $x \in L$ iff $(x, y_n) \in M$. Construct a nondeterministic Turing machine U which, on input (i, x) of size n, with $i \in \mathbb{N}$ and $x \in \{0,1\}^*$, simulates the i^{th} nondeterministic TM M_i on x, accepting iff M_i accepts within 2^{2n} steps. Since $L(U) \in \mathsf{NEXP}$, it can be decided by Boolean circuits of size n^k, where $k \in \mathbb{N}$ is fixed; then any language $M \in \mathsf{NTIME}(2^n)$ can be decided by Boolean circuits of size $(c+n)^k = O(n^k)$, where c is the length of the description of a nondeterministic 2^n-time Turing Machine for M. Then any $L \in \mathsf{NTIME}(2^n)/n$ can be decided by Boolean circuits of size $O(n^k)$, as desired.
Lemma 2.4. *If* $\mathsf{NEXP} = \mathsf{EXP}$, *then there is a constant* $d_0 \in \mathbb{N}$ *such that* $\mathsf{NTIME}(2^n)/n \subseteq \mathsf{DTIME}(2^{n^{d_0}})/n$.

Like above, for any $L \in \mathsf{NTIME}(2^n)/n$, there exists a nondeterministic 2^n-time deterministic Turing machine M and n-bit advice strings $\{a_n\}_{n\geq 0}$ such that $x \in L$ iff M accepts (x, a_n).
Corollary 2.5. *If* $\mathsf{NEXP} \subset \mathsf{P/poly}$ *then* $\mathsf{EXP} \not\subseteq \mathsf{io\text{-}[NTIME}(2^n)/n]$. *Similarly, if* $\mathsf{NEXP} = \mathsf{EXP}$ *then* $\mathsf{NEXP} \not\subseteq \mathsf{io\text{-}[NTIME}(2^n)/n]$.

If $\mathsf{NEXP} \subset \mathsf{P/poly}$, then by Lemma 2.3, there exists $d_0 \in \mathbb{N}$ such that $\mathsf{NTIME}(2^n)/n \subset \mathsf{SIZE}(n^{d_0})$; this contradicts Theorem 2.1, as $\mathsf{EXP} \subseteq \mathsf{NTIME}(2^n)/n$. The second result follows similarly by applying Lemma 2.4 and Theorem 2.2 analogously.

Note that [4] also proves a similar result, that if $\mathsf{NEE} = \mathsf{EE}$, then $\mathsf{NEE} \not\subseteq \mathsf{io\text{-}[NTIME}(2^{2^n})/n]$, in much the same manner.

The proof also makes use of results relating to derandomization.
Theorem 2.6. *There exists polynomial-time computable* $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *such that, for any oracle* A *and any* $\epsilon > 0$, *there exist* $\delta < \epsilon$ *and* $d \in \mathbb{N}$ *such that:*

$$F : \{0,1\}^{2^{n^d}} \times \{0,1\}^{n^\epsilon} \to \{0,1\}^n.$$

If r is the truth table of a n^δ-variable Boolean function of A-circuit complexity at least $n^{\delta d}$, then $G_r(s) := F(r,s)$ is a $\mathsf{SIZE}^A(n)$-pseudorandom generator from $\{0,1\}^{n^\epsilon} \to \{0,1\}^n$. [9]

In other words, hard Boolean functions (those with high circuit complexity) can be used to construct quick pseudorandom generators. Yao [13] and Nisan and Wigderson [10] showed that quick such $\mathsf{SIZE}(n)$-pseudorandom generators allow for any BPP algorithm to be simulated in deterministic $2^{n^{k\epsilon}}$ time for constant $k \in \mathbb{N}$. Goldreich and Zuckerman [3] similarly show that such a pseudorandom generator allows a nondeterministic $2^{n^{k\epsilon}}$-time decision procedure for any MA language.

The following result is due to [5] and [9].
Theorem 2.7. *There exists polynomial-time computable $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for any oracle A and any $\epsilon > 0$, there exist $c, d \in \mathbb{N}$ such that:*

$$F : \{0,1\}^{n^c} \times \{0,1\}^{d \log n} \to \{0,1\}^n.$$

If r is the truth table of a $c \log n$-variable Boolean function of A-circuit complexity at least n^{ec}, then $G_r(s) := F(r,s)$ is a $\mathsf{SIZE}^A(n)$-pseudorandom generator. from $\{0,1\}^{d \log n} \to \{0,1\}^n$.

This provides another hardness-randomness tradeoff, namely, if there exists a deterministic $\mathsf{poly}(2^n)$-time algorithm to generate truth tables of n-variable Boolean functions of circuit complexity at least $2^{\Omega(n)}$, then $\mathsf{BPP} = \mathsf{P}$; if such an algorithm is zero-error probabilistic, then $\mathsf{BPP} = \mathsf{ZPP}$.
Lemma 2.8. *Let $R(x,y)$ be a polynomial-time decidable relation on $\{0,1\}^n \times \{0,1\}^{2^n}$, $A \in \mathsf{EXP}$, and $s : \mathbb{N} \to \mathbb{N}$. If $f_R \neq \hat{f}_{R,A,s}$, then there exists a nondeterministic $\mathsf{poly}(2^n)$-time algorithm B and family of n-bit strings $\{x_n\}_{n \geq 0}$ such that for infinitely many values of $n \in \mathbb{N}$, B on advice x_{n+1} nondeterministically generates the truth table of an n-variable Boolean function which has A-oracle circuit complexity greater than $s(n)$.*

This result formalizes the notion of a lack of easy witnesses inducing a hardness test. Intuitively, define $x_{n+1} = 1z_n$ for $f_R(z_n) = 1$ and $\hat{f}_{R,A,s}(z_n) = 0$ (this must happen infinitely often) or $x_{n+1} = 0^{n+1}$ if no such z_n exists. The algorithm B then proceeds as follows: given advice $1z \in \{0,1\}^{n+1}$, nondeterministically find $y \in \{0,1\}^{2^n}$ such that $R(x,y)$ holds, output y and halt in the accepting state. Otherwise, given advice 0^{n+1}, output 0^{2^n} and halt in the accepting state. All nonzero truth tables will be those of functions with A-oracle circuit complexity greater than $s(n)$.

This result, combined with the prior results on pseudorandom generators and circuit complexity, implies that if $f_R \neq \hat{f}_{R,A,s}$ for $A \in \mathsf{EXP}$ and $s \in n^{\Omega(1)}$, then some particular probabilistic algorithms may be derandomized.
Theorem 2.9. *If $\mathsf{NEXP} \neq \mathsf{EXP}$, then for every $\epsilon > 0$, $\mathsf{AM} \subseteq \mathsf{io}\text{-}[\mathsf{NTIME}(2^{n^\varepsilon})/n^\varepsilon]$.*

If every poly-time decidable relation $R(x,y)$ on $\{0,1\}^n \times \{0,1\}^{2^n}$ has a $d \in \mathbb{N}$ such that $f_R = \hat{f}_{R,SAT,n^d}$, then $\mathsf{NEXP} \subseteq \mathsf{EXP}$ (this follows from a padding argument). As such, if $\mathsf{NEXP} \neq \mathsf{EXP}$, there is some such R and $s : \mathbb{N} \to \mathbb{N}$ such that $f_R \neq \hat{f}_{R,A,s}$. Then Lemma 2.8 gives the existence of an algorithm which nondeterministically generates truth tables of Boolean functions f_n such that for every $d \in \mathbb{N}$, f_n has SAT-oracle complexity greater than n^d for infinitely many n.

But [9] showed that the existence of such functions f_n implies that, for every $\epsilon > 0$, $\mathsf{AM} \subseteq \mathsf{io}\text{-}[\mathsf{NTIME}(2^{n^\epsilon})/a(n^\epsilon)]$.

The primary result of [4] is as follows:
Theorem 2.10. $\mathsf{NEXP} \subset \mathsf{P/poly} \iff \mathsf{NEXP} = \mathsf{MA}$.

Babai, Fortnow, and Lund [1] proved the following, weaker result, which will be useful for the main proof:
Theorem 2.11. $\mathsf{EXP} \subset \mathsf{P/poly} \implies \mathsf{EXP} = \mathsf{MA}$.

The first direction of the proof is due to Dieter van Melkebeek, in private correspondence:
Theorem 2.12. *If* $\mathsf{NEXP} = \mathsf{MA}$*, then* $\mathsf{NEXP} \subset \mathsf{P/poly}$.

If the assumption holds, then $\mathsf{NEXP} = \mathsf{EXP}$, so $\mathsf{EXP} \not\subset \mathsf{P/poly}$. But this means that there exists a hard Boolean function; from Theorem 2.6, we have $\mathsf{MA} \subseteq \mathsf{io\text{-}NTIME}(2^n)$, contradicting Corollary 2.5.
Theorem 2.13. *If* $\mathsf{NEXP} \subset \mathsf{P/poly}$*, then* $\mathsf{NEXP} = \mathsf{EXP}$.

Assume towards contradiction that $\mathsf{NEXP} \subset \mathsf{P/poly}$ but $\mathsf{NEXP} \not\subseteq \mathsf{EXP}$. By Theorem 2.11, the first assumption implies that $\mathsf{NEXP} = \mathsf{EXP}$; in particular, $\mathsf{EXP} = \mathsf{AM} = \mathsf{MA}$ (both are contained in NEXP). Then from Theorem 2.9, for every $\epsilon > 0$, $\mathsf{EXP} = \mathsf{AM} \subseteq \mathsf{io\text{-}[NTIME}(2^{n^\epsilon})/n^\epsilon]$, contradicting Corollary 2.5.

From Theorem 2.11, if $\mathsf{NEXP} \subset \mathsf{P/poly}$ and $\mathsf{NEXP} = \mathsf{EXP}$, then $\mathsf{NEXP} = \mathsf{MA}$. This, along with the previous two theorems immediately yields a proof for Theorem 2.10.

3 Applications

We highlight a few of the applications of Theorem 2.10.

3.1 Natural Properties

Previous proofs for circuit lower bounds for nonmonotone Boolean functions followed the same two-part structure [11]: First, a "natural" property of Boolean functions is defined such that any family of Boolean functions satisfying the property has high circuit complexity. Then an explicit family of Boolean functions satisfying the natural property is demonstrated.

Formally, for a class of languages L, a family $\mathcal{F} = \{\mathcal{F}_n\}_{n>0}$ of n-variable Boolean functions is *L-natural* if it has the conditions:

1. **Constructiveness**: the language T consisting of truth tables of Boolean functions in $\mathcal{F}$ is in L.
2. **Largeness**: There exists $c \in \mathbb{N}$ such that for every $N = 2^n$, we have $|T_N| \geq 2^N/N^c$ where T_N is the elements of T which have length N.

A property $\mathcal{F}$ is *useful* against $\mathsf{P/poly}$ if for every family of Boolean functions $f = \{f_n\}_{n\geq 0}$, if $f_n \in \mathcal{F}_n$ for infinitely many n, then $f \notin \mathsf{P/poly}$.
Theorem 3.1. *If there exists an* NP*-natural property that is useful against* $\mathsf{P/poly}$ *then* $\mathsf{NEXP} \not\subset \mathsf{P/poly}$.

If such a property exists, we may nondeterministically guess Boolean functions of superpolynomial circuit complexity in time polynomial in the size of their truth tables. In particular, these can be used to derandomize MA, as described by Theorem 2.6. This gives $\mathsf{NEXP} \neq \mathsf{MA}$, to which we may apply Theorem 2.10.

3.2 Circuit Approximation

Recall the *Circuit Acceptance Probability Problem (CAPP)*, that is, the problem of determining the fraction of inputs accepted by a Boolean circuit. This problem is solvable in probabilistic polynomial-time, and is complete for $\mathsf{promise}-\mathsf{BPP}$ [8]. We say that CAPP can be *nontrivally approximated* if for every $\epsilon > 0$, there is a nondeterministic 2^{n^ϵ}-time algorithm using advice of size n^ϵ approximating the acceptance probability of any Boolean circuit of size n, within additive error $\frac{1}{6}$, for infinitely many $n \in \mathbb{N}$. Such an M algorithm is *black-box* if it only has oracle access to the function computed by the circuit. Finally, such an algorithm is *non-adaptive* if the queries asked by M depends only on n and are computed before obtaining the result of any other query.

Theorem 3.2. *The following are equivalent:*

1. $\mathsf{NEXP} \not\subset \mathsf{P/poly}$.
2. *CAPP can be nontrivially approximated.*
3. *CAPP can be nontrivially approximated by a black-box non-adaptive algorithm.*

The implication (3) $\implies$ (2) is trivial. Suppose that (2) holds. If CAPP can be nontrivially approximated, then for any $\epsilon > 0$, we have $\mathsf{MA} \subseteq \mathsf{io}\text{-}[\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon]$. Then corollary 2.5 gives $\mathsf{NEXP} \neq \mathsf{MA}$; applying Theorem 2.10 yields $\mathsf{NEXP} \not\subset \mathsf{P/poly}$. Finally, if $\mathsf{NEXP} \not\subset \mathsf{P/poly}$, then there is some $\mathsf{poly}(2^n)$-time Turing Machine using advice of size n which nondeterministically generates truth tables of functions having n-variable circuit complexity of arbitrarily high complexity (at least n^d, where $d \in \mathbb{N}$), for infinitely many n. Then Theorem 2.6 gives nontrivial approximation.

3.3 Downward Closures and Gap Theorems

Theorem 2.10 also leads to a number of *downward closure results*, i.e. results showing that a collapse of higher complexity classes forces a collapse of lower complexity class. Such results are rare; in contrast, padding arguments can be used to derive a number of results in the opposite direction. Though these downward closures were proven by Fortnow in [2], the techniques used in [4] also induce gap theorems which were not available using these methods.

The downward closure results will rely upon the following characterization of probability distributions computable by polynomial-time Turing machines. A family of probability distributions $\mu = \{\mu_n\}_{n \geq 0}$ is P-*sampleable* if there is a polynomial $p(n)$ and poly-time Turing machine M such that, if $r \in \{0,1\}^{p(n)}$ is chosen uniformly at random, the output if $M(n,r)$ is an n-bit string distributed according to μ_n.

Lemma 3.3. *Suppose that for every $L \in \mathsf{BPP}$, every $\epsilon > 0$, and every P-sampleable distribution family μ, there exists a deterministic 2^{n^ϵ}-time algorithm A such that $\Pr_{x \sim \mu_n}[A(x) \neq \chi_L(x)] < 1/n$ for infinitely many $n \in \mathbb{N}$. Then for every $\epsilon > 0$, $\mathsf{BPE} \subseteq \mathsf{io}\text{-}[\mathsf{DTIME}(2^{2^{\epsilon n}})/n]$.*

Let $\epsilon > 0$; for any $L \in \mathsf{BPE}$, associate the padded language $L_{pad} \in \mathsf{BPP}$, where for each $x \in L$ with $|x| = n$, L_{pad} contains padded versions of x of every length $[2^n..2^{n+1}]$. For any $n \in \mathbb{N}$ and $0 \leq i < 2^n$, there are 2^n strings which could be padded elements of L; but these have length all at least $m = 2^n + i$. The uniform distribution over such strings assigns each such string probability at least $\frac{1}{m}$; then this distribution is P-sampleable, as we can define $M(m,r)$ to output $r0^{m-n}$.

Then by assumption, there is a 2^{n^ϵ}-time algorithm A such that $\Pr_{x\sim\mu_n}[A(x) \neq \chi_L(x)] < 1/n$ for infinitely many $n \in \mathbb{N}$. Then, for infinitely many $n \in \mathbb{N}$, there exists $0 \leq i < 2^n$ such that A must be correct on every string of the form $x0^{2^n-n+i}$, because each string has probability at least $\frac{1}{m}$. We can take the n-bit encodings of i as advice for a deterministic algorithm using linear advice running in $O(2^{2^{\epsilon n}})$ time, deciding L infinitely often.

It may not be obvious how this result may be useful; the following theorem due to Impagliazzo and Wigderson [6] can be used in conjunction with Lemma 3.3.

Theorem 3.4. *Suppose that* $\mathsf{EXP} \neq \mathsf{BPP}$. *Then for every* $L \in \mathsf{BPP}$ *and every* $\epsilon > 0$, *there is a deterministic* 2^{n^ϵ}*-time algorithm* A *such that, for every* P*-sampleable distribution family* μ, *there are infinitely many* $n \in \mathbb{N}$ *such that* $\Pr_{x\sim\mu_n}[A(x) \neq \chi_L(x)] < 1/n$.

These results now allow us to prove the following:

Theorem 3.5. *If* $\mathsf{EXP} \neq \mathsf{BPP}$, *then for every* $\epsilon > 0$, $\mathsf{BPE} \subseteq \mathsf{io}\text{-}[\mathsf{DTIME}(2^{2^{\epsilon n}})/n]$.

If $\mathsf{EXP} \neq \mathsf{BPP}$, then Theorem 3.4 yields the premise of Lemma 3.3, which proves the claim.

We now prove the first downward closure result. Note that a padding argument suffices for the reverse direction.

Theorem 3.6. $\mathsf{EE} = \mathsf{BPE} \implies \mathsf{EXP} = \mathsf{BPP}$.

If $\mathsf{BPE} = \mathsf{EE}$ but $\mathsf{BPP} \neq \mathsf{EXP}$, then Theorem 3.5 gives $\mathsf{BPE} \subseteq \mathsf{io}\text{-}[\mathsf{DTIME}(2^{2^n})/n]$. It follows that $\mathsf{EE} \subseteq \mathsf{io}\text{-}[\mathsf{DTIME}(2^{2^n})/n]$, which can be shown to be false by a diagonalization argument.

Note also that Theorem 3.5 gives the following gap theorem:

Theorem 3.7. *Either* $\mathsf{BPP} = \mathsf{EE}$ *or, for every* $\epsilon > 0$, $\mathsf{BPE} \subseteq \mathsf{io}\text{-}[\mathsf{DTIME}(2^{2^{\epsilon n}})/n]$

As mentioned in the proof for Theorem 3.6, it is impossible for both to hold. If $\mathsf{EE} \neq \mathsf{BPE}$, then $\mathsf{EXP} \neq \mathsf{BPP}$ by padding, and Theorem 3.5 completes the proof.

Similar downward closure results and gap theorem are proved for randomized language classes in [4], where $\mathsf{EXP} = \mathsf{ZPP} \iff \mathsf{EE} = \mathsf{ZPE}$ and $\mathsf{EXP} = \mathsf{RP} \iff \mathsf{EE} = \mathsf{RE}$; the authors also provide a similar gap theorem for ZPE, namely:

Theorem 3.8. $\mathsf{ZPP} \neq \mathsf{EE}$ *iff for every* $\epsilon > 0$, $\mathsf{ZPE} \subseteq \mathsf{io}\text{-}\mathsf{DTIME}(2^{2^{\epsilon n}})$.

Additionally, [4] also proves a gap theorem for MA.

Theorem 3.9. $\mathsf{MA} \neq \mathsf{NEXP}$ *iff for every* $\epsilon > 0, \mathsf{MA} \subseteq \mathsf{io}\text{-}[\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon]$.

If $\mathsf{MA} \neq \mathsf{NEXP}$ then Theorem 2.10 gives $\mathsf{NEXP} \not\subset \mathsf{P/poly}$. Then there is a $\mathsf{poly}(2^n)$-time Turing machine which uses advice n to nondeterministically generate a truth table of n-variable Boolean function f_n of superpolynomial circuit complexity. This follows by construction; the algorithm can nondeterministically guess strings with certificates of exponential length, and verify that f_n accepts in polynomial time; then output the characteristic function of f_n. But applying 2.6 yields that for any $\epsilon > 0$, $\mathsf{MA} \subseteq \mathsf{io}\text{-}[\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon]$. In the other direction, if $\mathsf{MA} = \mathsf{NEXP}$ and $MA \subseteq \mathsf{io}\text{-}[\mathsf{NTIME}(2^n)/n]$, Corollary 2.5 yields a contradiction.

3.4 NEXP- Search Problems

Though the main result is not directly used, the same reasoning can be applied to show that if $\mathsf{NEXP} = \mathsf{AM}$, then every NEXP search problem can be solved in deterministic $2^{\mathsf{poly}(n)}$ time.

Theorem 3.10. *If* $\mathsf{NEXP} = \mathsf{AM}$, *then for every language* $L \in \mathsf{NEXP}$, *there exists* $d \in \mathbb{N}$ *such that, for* n *sufficiently large, every* n*-bit string* $x \in L$ *has at least one witness* $y \in \{0,1\}^{2^{\mathsf{poly}(n)}}$ *which can be described by a SAT-oracle circuit of size at most* n^d.

Assume towards contradiction that $\mathsf{NEXP} = \mathsf{AM}$ but the conclusion does not hold. If, for every poly-time decidable relation $R(x, y)$ on $\{0,1\}^n \times \{0,1\}^{2^n}$, there exists a $d \in \mathbb{N}$ such that $f_R = \hat{f}_{R,\mathrm{SAT},n^d}$, then the conclusion holds (by padding). So suppose some such R exist where, for every $d \in \mathbb{N}$, we have $f_R \neq \hat{f}_{R,\mathrm{SAT},n^d}$. Then Lemma 2.8 shows that there is an algorithm to nondeterministically generate truth tables of n-variable Boolean functions of SAT-oracle circuit complexity greater than n^d. The same result as used in the proof of Theorem 2.9 then gives $\mathsf{AM} \subseteq \mathsf{io}\text{-}[\mathsf{NTIME}(2^{n^\epsilon})/n^\epsilon]$ for any $\epsilon > 0$. Under the assumption that $\mathsf{NEXP} = \mathsf{EXP} = \mathsf{AM}$, this contradicts Corollary 2.5.

Under the assumption that $\mathsf{NEXP} = \mathsf{AM}$, Theorem 3.10 demonstrates that witnesses for any $L \in \mathsf{NEXP}$ can be found by enumerating all SAT-oracle circuits of fixed polynomial-size and checking to see if any encodes a witness; this procedure can be done deterministically in exponential time.

4 Conclusion

The primary result that $\mathsf{NEXP} \subset \mathsf{P/poly} \iff \mathsf{NEXP} = \mathsf{MA}$ is significant in that it demonstrates that *any* derandomization of MA is equivalent to proving a nontrivial circuit bound on NEXP. Work in the field of derandomization with regards to complexity classes has continued since. More recently, Ryan Williams [12] shows that $\mathsf{NTIME}(2^n)$ does not have non-uniform ACC circuits of polynomial size (ACC consists of circuits using unbounded fan-in AND, OR, NOT, and MOD_m gates with constant depth), using fast satisfiability problems and the nondeterministic time hierarchy theorem. While this unconditional derandomization result helps resolve a major open problem in computational complexity, it remains to be seen whether MA can be derandomized, and if so, whether there is a standard procedure for doing so.

References

[1] L.Babai, L. Fortnow, and C. Lund. Non-deterministic Exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[2] L. Fortnow. Comparing notions of full derandomization. In *Proceedings of the Sixteenth Annual IEEE Conference on Computational Complexity*, p. 28–34, 2001.

[3] O. Goldreich and D. Zuckerman. Another proof that BPP⊆PH (and more). *Electronic Colloquium on Computational Complexity*, TR97-045, 1997.

[4] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[5] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, p. 220–229, 1997.

[6] R. Impagliazzo and A. Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *Proceedings of the Thirty-Ninth Annual IEEE Symposium on Foundations of Computer Science*, p. 734–743, 1998.

[7] V. Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *Journal of Computer and System Sciences*, 63(2):236–252, 2001. (preliminary version in CCC'00).

[8] V. Kabanets, C. Rackoff, and S. Cook. Efficiently approximable real-valued functions. *Electronic Colloquium on Computational Complexity*, TR00–034, 2000.

[9] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial hierarchy collapses. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, p. 659–667, 1999.

[10] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[11] A.A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55:24–35, 1997.

[12] R. Williams. Nonuniform ACC Circuit Lower Bounds. *Journal of the ACM*, 61:(1):2–32, 2014.

[13] A.C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science*, p. 80–91, 1982.