Electronic Colloquium on Computational Complexity, Report No. 146 (2009)

Derandomizing Arthur-Merlin Games and Approximate Counting Implies Exponential-Size Lower Bounds

Dan Gutfreund[*]
dannygu@il.ibm.com
IBM Research, Haifa, Israel

Akinori Kawachi[†]
kawachi@is.titech.ac.jp
Dept. Math. & Comp. Sci., Tokyo Inst. Tech.

Abstract

We show that if Arthur-Merlin protocols can be derandomized, then there is a Boolean function computable in deterministic exponential-time with access to an NP oracle, that cannot be computed by Boolean circuits of *exponential* size. More formally, if prAM $\subseteq$ P$^{\mathrm{NP}}$ then there is a Boolean function in E$^{\mathrm{NP}}$ that requires circuits of size $2^{\Omega(n)}$. prAM is the class of promise problems that have Arthur-Merlin protocols, P$^{\mathrm{NP}}$ is the class of functions that can be computed in deterministic polynomial-time with an NP oracle and E$^{\mathrm{NP}}$ is its exponential analogue. The lower bound in the conclusion of our theorem suffices to construct very strong pseudorandom generators.

We also show that the same conclusion holds if the problem of approximate counting the number of accepting paths of a nondeterministic Turing machine up to multiplicative factors can be done in nondeterministic polynomial-time. In other words, showing nondeterministic fully polynomial-time approximation schemes for ♯P-complete problems require proving exponential-size circuit lower bounds.

A few works have already shown that if we can find efficient deterministic solutions to some specific tasks (or classes) that are known to be solvable efficiently by randomized algorithms (or proofs), then we obtain lower bounds against certain circuit models. These lower bounds were only with respect to polynomial-size circuits *even if full derandomization is assumed.* Thus they only implied fairly weak pseudorandom generators (if at all).

A key ingredient in our proof is a connection between computational learning theory and exponential-size lower bounds. We show that the existence of deterministic learning algorithms with certain properties implies exponential-size lower bounds, where the complexity of the hard function is related to the complexity of the learning algorithm.

1 Introduction

1.1 Background

The fascinating connection between the existence of explicit functions that cannot be computed by small Boolean circuits and efficiently computable pseudorandom generators (PRGs) that suffice for derandomization, is one of the greatest achievements of complexity theory. The following two are equivalent [14]:

1. There exist a Boolean function in the class E = TIME($2^{O(n)}$) that requires Boolean circuits of size $2^{\Omega(n)}$ to be computed.

[*]Most of the research was done while the author was at Tel-Aviv University and supported by Oded Regev's European Research Council (ERC) Starting Grant.

[†]The author was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B), 18300002, 2007.

ISSN 1433-8092

2. There exist a PRG $G : \{0,1\}^m \rightarrow \{0,1\}^n$ that is computable in time $\mathrm{poly}(n)$ and fools Boolean circuits of size $\mathrm{poly}(n)$, where $n = 2^{\Omega(m)}$.

It follows that both these items imply derandomization of probabilistic polynomial-time algorithms with only polynomial-time overhead in the running time (in the sequel we call such derandomizations *full*). Namely, BPP = P. This connection, which was coined *hardness vs. randomness*, supported the common belief (or maybe even is the origin of the belief) that in algorithmic settings, randomness does not enhance computational power in a significant way. Furthermore, it pointed out a tight relation between two central concepts in computational complexity: circuit lower bounds and pseudorandomness.

In fact, this connection is so deep and profound that it extends to many other settings. For example, the equivalence also holds in the nondeterministic setting. I.e., hardness against non-deterministic circuits implies PRGs that fool non-deterministic circuits and hence the derandomization of the class AM (of languages for which membership can be proven via a constant-round interactive proof) which is the natural probabilistic extension of NP [22, 26, 30]. Relativized versions are also true [20]. E.g., one can add an NP oracle to all the machines and circuits involved in the equivalence above and obtain the derandomization of the class $\mathrm{BPP}^{\mathrm{NP}}$ (of languages that can be computed in probabilistic polynomial-time with access to an NP oracle). And finally, the equivalence also extends to other settings of parameters. For example, one can weaken the lower bound in Item 1 to hold against circuits of size $\mathrm{poly}(n)$, and then weaken the quality of the PRG in Item 2 so it only has a polynomial stretch, i.e., $n = \mathrm{poly}(m)$. This in turn implies a weaker derandomization of BPP placing it in the class SUBEXP [2]. Furthermore, there is a smooth transition of tradeoffs between the hardness in Item 1 and the quality of the PRG in Item 2, where the exponential setting of parameters that we stated above is at the one extreme (called the *high-end*) and the polynomial setting is at the other (the *low-end*).

Unfortunately, it is a challenging task to prove lower bounds for circuit size in general, and the hardness vs. randomness paradigm has been useful in obtaining *unconditional* derandomizations only in very limited computational models [24, 31]. A natural question then arises: Do we really need to prove circuit lower bounds in order to derandomize more general randomized complexity classes such as BPP or AM? Several works investigated this question and showed that in some settings the answer is yes, i.e., derandomization itself implies circuit lower bounds! The first result of this flavor was given by Impagliazzo, Kabanets and Wigderson [13] who showed that if the class MA is contained in subexponential nondeterministic time then NEXP $\not\subseteq$ P/poly. A similar conclusion follows from the derandomization of the class prBPP (of promise problems that can be solved in probabilistic polynomial-time) since it implies the derandomization of the class MA. Kabanets and Impagliazzo [16] showed that if the problem of Polynomial Identity Testing, which is known to be in BPP, is in SUBEXP then either NEXP $\not\subset$ P/poly or computing the Permanent cannot be done by polynomial-size arithmetic circuit. Results of a similar flavor were given in [25, 1, 19].

While the lower bounds obtained from the derandomization assumptions in the above mentioned results are not strong enough to obtain PRGs that imply back the derandomization assumptions, they still suggest that the two-way connections between hardness and pseudorandomness also extend to derandomization. A natural question is how general this phenomena is? Is it as general as the equivalence between circuit lower bounds and pseudorandomness which holds in so many different settings? Can we extend it to other settings of parameters or computation models?

Note that all of the above mentioned results start with the assumption that a weak derandomization is possible (placing some probabilistic class in a subexponential class that does not require probability) and conclude in a lower bound for superpolynomial-size (either Boolean or arithmetic) circuits. Thus the connections hold in the low-end setting of parameters, and in particular, they only imply PRGs with polynomial stretch. (We mention that some of the results do not imply PRGs at all as they obtain lower bounds which are seemingly too weak for the construction of PRGs). An exception is the result of Kinne, Shaltiel and van Melkebeek [19] who gave an alternative proof to [16] for which the parameters scale better. Thus they obtain results also for parameter settings in between the very low-end and the very high-end. However, their proof still falls far short from proving a connection for the high-end (namely an exponential-size lower bounds from full derandomization), and furthermore, their lower bounds, just like [16], are with respect to arithmetic circuits and thus do not imply PRGs that fool Boolean circuits. Inspecting the proofs of all the above mentioned results, one can see that they do not imply stronger lower bounds and PRGs even if full derandomization is assumed. (See more on previous proof techniques in Section 1.3.)

1.2 Our Results

Arthur-Merlin games. In this paper we extend the connections between derandomization, circuit lower bounds and PRGs to the high-end setting, by showing that a full derandomization of a probabilistic class (and in fact a certain task, see below) implies exponential-size circuit lower bounds and PRGs with exponential stretch.

Theorem 1.1 If $\mathrm{prAM} \subseteq \mathrm{P}^{\mathrm{NP}}$ then there exist a constant $\delta > 0$ and a Boolean function in the class E^{NP} that cannot be computed by circuits of size $2^{\delta n}$ for infinitely many input lengths.

prAM is the class of promise problems that have Arthur-Merlin protocols, P^{NP} is the class of functions that can be computed in deterministic polynomial-time with an NP oracle and E^{NP} is its exponential analogue. Note that a full derandomization of prAM would place it in the class NP. Our derandomization assumption is weaker since clearly $\mathrm{NP} \subseteq \mathrm{P}^{\mathrm{NP}}$.

An immediate consequence of Theorem 1.1 is that the derandomization of the class prAM implies the existence of PRGs with *exponential* stretch that can be computed efficiently with an NP oracle, and fool deterministic Boolean circuits. Theorem 1.1 is an "almost" converse of the hardness vs. randomness tradeoffs of [22, 26, 30] who showed that the existence of a Boolean function in the class E^{NP} that requires *nondeterministic* circuits of size $2^{\Omega(n)}$ implies the existence of a PRG with exponential stretch that can be computed efficiently with an NP oracle, and fools *nondeterministic* Boolean circuits. This in turn implies that $\mathrm{prAM} \subseteq \mathrm{P}^{\mathrm{NP}}$.

A close inspection of our proof, and the proofs in [8] that it relies on, reveals that for obtaining an exponential-size lower bound, it is enough to derandomize the lower bound protocol of Goldwasser-Sipser [12] (and not necessarily the whole of prAM), which is an Arthur-Merlin protocol for proving that a set which can be recognized by a small circuit is large (see Lemma 3.3 for the exact formulation of this problem).

Theorem 1.2 If the Goldwasser-Sipser protocol can be done in nondeterministic polynomial-time, or more precisely if the promise problem Π from Lemma 3.3 is in NP then there exist a constant $\delta > 0$ and a Boolean function in the class E^{NP} that cannot be computed by circuits of size $2^{\delta n}$ for infinitely many input lengths.

Remark 1.3 We mention that proving in nondeterministic polynomial-time a lower bound on the size of a set which can be recognized by a small *nondeterministic* circuit implies a full derandomization of prAM (almost by definition) and hence the lower bound by Theorem 1.1. The point in Theorem 1.2 is that the lower bound already follows from derandomizing the lower bound protocol for sets that are recognized by small *deterministic* circuits. This follows from our proof of Theorem 1.1 (but not necessarily from its statement). See Section 4.2 for details.

Approximate counting. Counting the number of accepting paths of a nondeterministic Turing machine (TM) is an important computational problem as it is related to computational questions in combinatorics (e.g., counting various structures in graphs) and algebra (e.g., computing the permanent of a matrix).

The class ♯P is the class of functions computing the number of accepting paths of a given TM (on a given input). A canonical ♯P-complete function is the following: given a Boolean circuit C on n inputs, compute the size of the set $C^{-1}(1) = \{x \in \{0,1\}^n : C(x) = 1\}$. This is the problem ♯Circuit-SAT, which is the counting version of the canonical NP-complete problem Circuit-SAT (of deciding whether $|C^{-1}(1)| > 0$ or not). The class ♯P is extremely powerful as was shown by Toda [29]: with oracle to ♯P it is possible to compute every function in the polynomial-time hierarchy (PH) in deterministic polynomial-time. Thus unless PH collapses, we do not expect to be able to compute ♯Circuit-SAT within PH.

However, estimating ♯Circuit-SAT is easier. By a standard application of the Chernoff bound, estimating ♯Circuit-SAT up to an additive factor, namely computing an estimate γ such that $|C^{-1}(1)| - \epsilon 2^n \leq \gamma \leq |C^{-1}(1)| + \epsilon 2^n$, can be done in probabilistic poly$(|C|, 1/\epsilon)$-time. A seemingly harder problem (in fact NP-hard) is to obtain a relative-error approximator (as it is coined in [27]). This is a procedure that outputs an estimate γ such that $(1-\epsilon)|C^{-1}(1)| \leq \gamma \leq (1-\epsilon)^{-1}|C^{-1}(1)|$. The problem of obtaining relative-error approximators received a lot of attention as it is directly related to obtaining approximation schemes for problems in ♯P. A classic result in this area [28, 15, 4] is that it is possible to compute a relative-error approximator in probabilistic poly$(|C|, 1/\epsilon)$-time with access to an NP oracle. This in turn implies that every function in ♯P has a fully polynomial-time randomized approximation scheme with access to an NP oracle. Another way to look at this is that there is a randomized (Turing) reduction from the problem of relative-error approximators to the problem Circuit-SAT.

This suggests that relative-error approximation is much easier than exact computation of the number of accepting inputs (or paths in TMs), and its complexity is close to the complexity of deciding whether there are accepting inputs or not. It is a major open problem whether relative-error approximators can actually be computed in nondeterministic polynomial-time. Shaltiel and Umans [27] showed that under a hardness assumption against exponential-size nondeterministic circuits, relative error-approximators can be computed in deterministic poly$(|C|, 1/\epsilon)$-time with access to an NP oracle. That is, under their assumption, every function in ♯P has a fully polynomial-time approximation scheme with access to an NP oracle.

We complement this by showing that if relative-error approximators can be computed in nondeterministic polynomial-time then there are exponential-size lower bounds.[1] The proof of the following theorem follows easily from Theorem 1.2. See Section 4.2 for details.

Theorem 1.4 If there is a nondeterministic TM that on input a Boolean circuit C and a parameter

[1]We mention that there is a closely related problem to approximate counting called approximate sampling, in which the aim is to sample an almost uniform element in the set $C^{-1}(1)$ (given the Boolean circuit C). [15] showed a *randomized* reduction from approximate sampling to approximate counting. The derandomized object that is related to approximate sampling was defined in [27], and it is a generalization of the notion of PRG. By the equivalence between PRGs and lower bounds discussed above, it is clear that derandomizing approximate sampling implies exponential-size lower bounds. However this was not known to hold for approximate counting. The best known result in this direction is by [13] who showed that computing (even) additive-error approximators in nondeterministic polynomial-time (and even subexponential-time) implies that NEXP $\not\subseteq$ P/poly.

$0 < \epsilon < 1$, runs in nondeterministic time $\mathrm{poly}(|C|, 1/\epsilon)$, has at least one accepting path, and on every accepting path outputs a number γ, such that $(1-\epsilon)|C^{-1}(1)| \leq \gamma \leq (1-\epsilon)^{-1}|C^{-1}(1)|$, then there exist a constant $\delta > 0$ and a Boolean function in the class E^{NP} that cannot be computed by circuits of size $2^{\delta n}$ for infinitely many input lengths.

In other words while it is possible that there is a fully polynomial-time nondeterministic approximation scheme for ♯Circuit-SAT (and hence for every function in ♯P), showing it requires proving exponential-size circuit lower bounds.

1.3 Our and Previous Proof Techniques

In this section we discuss previous proof techniques and their limitations, followed by a short description of our proof and how it overcomes these limitations. Let us start with describing the general strategy of [13] to prove that the derandomization of MA implies that there is a function in NEXP that requires circuits of super-polynomial size. Later results in this line are based on a similar strategy (or directly on [13]).[2] This strategy dates back to Kannan [17] who used it to prove circuit lower bounds (without unproven assumptions). Consider two cases, either NEXP $\subseteq$ P/poly, or not. If it does not, we are done. Otherwise, [13] shows the following Karp-Lipton style collapse:[3] if NEXP $\subseteq$ P/poly then NEXP = MA. Now if MA $\subseteq$ NSUBEXP (i.e., there is derandomization) then NEXP = NSUBEXP which contradicts the nondeterministic time hierarchy.

Suppose that we want to use this strategy to derive a lower bound against exponential-size circuits in some uniform class $\mathcal{C}$. We will need in this case to condition on whether $\mathcal{C}$ (or some class that is contained in $\mathcal{C}$) can be computed by exponential-size circuits or not. If it does not we are done. If it does, we will need to prove some Karp-Lipton style collapse from the assumption that $\mathcal{C}$ can be computed by exponential-size circuits. The problem is that this assumption is too weak and we do not know of such collapses. This issue also arises when trying to prove lower bounds in the exponential-time hierarchy, and is the reason for the gap between the known super-polynomial size lower bounds (which are known for classes contained in the second level of the hierarchy [17, 6, 21, 7]) and the known exponential-size lower bounds (which are only known for classes in the third level of the hierarchy [17]). Miltersen, Vinodchandran and Watanabe [23], who investigated this issue argue that Karp-Lipton style collapses that are needed for Kannan's strategy hold with respect to size functions up to *half-exponential* (a function f is half-exponential if $f(f(n)) \in 2^{\Theta(n)}$) but do not seem to carry over to larger size bounds.

Thus in order to prove Theorem 1.1 we need a different strategy. Somewhat surprisingly, our proof also goes via an easy/hard case analysis but not with respect to the classes that we are interested in, namely E^{NP} and exponential-size circuits, but rather NP and fixed polynomial-size circuits. Consider two cases, either SAT can be computed by circuits of size, say n^{10}, or not. In the former case we are in a good position because we can use a Karp-Lipton style collapse. Chakaravarthy and Roy [8] showed that if SAT has polynomial-size circuits then the polynomial-time hierarchy (PH) collapses to the class $\mathrm{P}^{\mathrm{prAM}}$, and therefore by the derandomization assumption to P^{NP}. Kannan [17] showed that one can compute in the third level of PH a function on $O(\log n)$ bits whose circuit complexity is n^{δ} (for some $\delta > 0$), i.e., the function is computable in $\mathrm{poly}(n)$ time with three alternations. By the collapse this function can be computed in P^{NP}. By translation to exponential time bounds this

[2]An exception is the work of [19] who takes a different route and indeed obtain lower bounds for more general settings of parameters. However, they fall short of implying PRGs and they do not break the "half-exponential barrier" that we discuss below.

[3]Results that show the containment of some uniform class in a non-uniform class imply a collapse of high uniform classes into lower classes are called Karp-Lipton style collapses (after [18] who were the first to show such a result).

implies the desired lower bound.

The second case, in which SAT does not have circuits of size n^{10}, is more interesting. This is because at a first glance, the (fixed) polynomial-size lower bound for SAT seems to have nothing to do with exponential-size lower bounds. We show that in fact it does and the connection is via computational learning theory. Let us briefly discuss some notions from this theory. Let $s(n), s'(n)$ be size functions (where $s'(n) \geq s(n)$). An algorithm A exactly learns a Boolean function f with respect to the concept class s and hypothesis class s', if the following holds: if f can be computed by circuits of size $s(n)$, then A, on input 1^n, outputs a circuit of size $s'(n)$ that computes f at input length n (i.e., A has to learn a circuit for f from the hypothesis class but not necessarily from the concept class). A classic result in computational learning theory by Bshouty, Cleve, Gavaldà, Kannan, and Tamon [6] is that there is an algorithm that exactly learns SAT with respect to the concept class $\mathrm{SIZE}(n^k)$ and hypothesis class $\mathrm{SIZE}(n^{k+2})$ (for any $k > 1$). The algorithm runs in probabilistic expected polynomial-time with access to an NP oracle. Of course, we do not know if SAT has polynomial-size circuits. Indeed in the case that we consider, SAT cannot be computed by n^{10}-size circuits. So how does the algorithm of [6] behaves when SAT is not even in the hypothesis class $\mathrm{SIZE}(n^{k+2})$? Fortnow, Pavan and Sengupta [9] observed that in this case the algorithm outputs a $\mathrm{poly}(n)$-long list of SAT instances such that *every* circuit of size n^k fails to compute correctly the SAT-value of at least one of them. We call this *a list of counterexamples*.

We proceed in two steps. First we show that if there is a *deterministic* learning algorithm that outputs a polynomially-long list of counterexamples, then there is a function that requires exponentially large circuits (see Lemma 3.2). The complexity of computing this function is directly related to the complexity of the learning algorithm. In particular if the algorithm runs in deterministic polynomial-time with access to an NP oracle, then there is a function in E^{NP} that requires exponentially large circuits. Next we show (in Theorem 3.4), based on ideas from [6, 9, 8], that there is a *deterministic* algorithm that uses an oracle to prAM for learning counterexamples (recall that [9] only gives a randomized algorithm, so it is not good for us). By the hypothesis of the theorem, the algorithm is in fact in P^{NP} and the lower bound follows.

1.4 On the Search for Truth-Tables of Hard Functions

Proving exponential-size circuit lower bounds for functions in the exponential-time hierarchy is equivalent to the problem of finding in the polynomial-time hierarchy a truth-table of length n of a function on $\lfloor \log n \rfloor$ bits that cannot be computed by circuits of size n^{δ}, for some $\delta > 0$ (for every or at least infinitely many n's). Indeed Kannan's lower bound [17] can be viewed in this way: given a truth-table of length n one can verify in Σ_2^{P} that it is the first lexicographic truth-table of a function that requires circuits of size n^{δ}. Now one can run a binary search, using a Σ_2^{P} oracle, through all length n truth-tables to find in $\mathrm{poly}(n)$ time the lexicographic first hard one. This proves that there is a Boolean function in the class $\mathrm{E}^{\Sigma_2^{\mathrm{P}}}$ that requires circuits of size $2^{\Omega(n)}$,[4] and it is currently (as it has been for almost 30 years) the best known exponential-size lower bound.

Our proof can also be viewed in this light. We show that for searching exponentially hard truth-tables we can replace the Σ_2^{P} oracle with a prAM oracle. Note that $\mathrm{prAM} \subseteq \Pi_2^{\mathrm{P}}$ [10] (and clearly oracle access to Σ_2^{P} is equivalent to oracle access to Π_2^{P}), so our oracle is certainly not stronger than the oracle in Kannan's proof, and is widely believed to be much weaker. Indeed our result implies the lower bound for $\mathrm{E}^{\Sigma_2^{\mathrm{P}}}$. In Section 5 we explain why we nevertheless do not prove a new explicit lower bound.

[4]In [17] a weaker lower bound was proven. The lower bound for $\mathrm{E}^{\Sigma_2^{\mathrm{P}}}$ that we described is obtained by a small modification of Kannan's proof which is considered folklore (yet appears explicitly in [23]).

2 Basic Notions and Notation

For a Boolean function $f : \{0,1\}^* \to \{0,1\}$, we denote by f^n the restriction of f to instances of length n. For a (possibly infinite) family of circuits $\mathcal{C}$ we denote by $\mathcal{C}^n$ the circuits in $\mathcal{C}$ with exactly n input gates.

For an integer $n > 0$, we denote by $[n]$ the set $\{1, \ldots, n\}$. for a string $s \in \{0,1\}^*$ we denote by $|s|$ the length of s. For two strings $s, t \in \{0,1\}^*$ we denote by $s \circ t$ their concatenation.

2.1 Complexity Classes

We assume that the reader is familiar with standard complexity classes such as P, NP, E etc. For a class of algorithms $\mathcal{A}$ and a class of functions $\mathcal{F}$ we denote by $\mathcal{A}^{\mathcal{F}}$ the class of functions that are computable by some algorithm in $\mathcal{A}$ that is given an oracle access (i.e., unit cost) to a function in $\mathcal{F}$.

Often when we describe algorithms that use as oracle some function in a class $\mathcal{F}$ it is convenient to actually assume that the algorithm has unit cost access to several (constant number) of functions $f_1, \ldots, f_c$ all in $\mathcal{F}$. We can then think of the algorithm having access to a single function in $\mathcal{F}$ by binding the functions to a single function $f(i, x) = f_i(x)$ for $1 \leq i \leq c$.

For a size function $s : \mathbb{N} \to \mathbb{N}$, we denote the class of $s(n)$-size n-input Boolean circuits by SIZE$(s(n))$. For a fixed size function s, for every $n \in \mathbb{N}$, there exist a certain $m = \text{poly}(s(n))$ such that all the n-input circuits of size $s(n)$ can be described by strings of length m. For simplicity we assume that $m = s(n)$, and thus our size function will be the description size of the circuits (rather than say the number of gates).

2.2 Promise Problems

A promise problem Π is defined by two disjoint sets $\Pi^Y \subseteq \{0,1\}^*$ which we call the 'yes' instances of Π, and $\Pi^N \subseteq \{0,1\}^*$ which we call the 'no' instances of Π. A function $f : \{0,1\}^* \to \{0,1\}$ agrees with a promise problem Π, if $f(x) = 1$ for every $x \in \Pi^Y$, $f(x) = 0$ for every $x \in \Pi^N$ and $f(x)$ can take any value in $\{0,1\}$ if $x \notin \Pi^Y \cup \Pi^N$.

For a class of algorithms $\mathcal{A}$ and a class of promise problems $\mathcal{F}$, we say that a function $g : \{0,1\}^* \to \{0,1\}$ is in the class $\mathcal{A}^{\mathcal{F}}$, if there exist an algorithm $A \in \mathcal{A}$ and a promise problem $\Pi \in \mathcal{F}$, such that when A is given oracle access to *any* function f that agrees with Π, it computes the function g. In other words, while A may ask queries which are not in $\Pi^Y \cup \Pi^N$ and hence receive arbitrary answers, it must compute the same function g regardless of the values of these arbitrary answers.

We say that a class of promise problems $\mathcal{F}$ is contained in a class of Boolean functions $\mathcal{C}$, if for every promise problem $\Pi \in \mathcal{F}$, there exists a function $c \in \mathcal{C}$ which agrees with Π. As an example in our context, consider the conditional derandomization result that follows from [22]: under a certain hardness assumption against nondeterministic circuits, for every $\Pi \in$ prAM (defined below) there is a nondeterministic polynomial-time machine that answers correctly on Π^Y and Π^N. On instances not in $\Pi^Y \cup \Pi^N$ it answers something but these answers are arbitrary. Thus we get that under their assumption prAM $\subseteq$ NP.

The class prAM contains all the promise problems for which there is an Arthur-Merlin protocol whose completeness holds with respect to all the 'yes' instances and the soundness holds with respect to all the 'no' instances. The protocol may behave arbitrarily on instances which are not in $\Pi^Y \cup \Pi^N$. Formally,

Definition 2.1 We say that a promise problem Π is in the class prAM if there is a polynomial-time computable relation $R(\cdot, \cdot, \cdot)$ such that the following holds:

- **Completeness:** For every $x \in \Pi^Y$, $\Pr_r[\exists y$ such that $R(x,y,r)=1] \geq 2/3$
- **Soundness:** For every $x \in \Pi^N$, $\Pr_r[\exists y$ such that $R(x,y,r)=1] \leq 1/3$,

where $|r| = |y| = \text{poly}(|x|)$.

It is well known [3, 12] that the definition above is equivalent to the class of all the promise problems that have interactive protocols (in the model of [11]) with a constant number of rounds between an all-powerful prover (Merlin) and a probabilistic polynomial-time verifier (Arthur).

3 Learning Counterexamples

3.1 Learning Counterexamples Implies Exponential-Size Lower Bounds

In this section we show the connection between the problem of learning counterexamples for SAT and exponential-size lower bounds. First, we formally define the problem of learning counterexamples.

Definition 3.1 Let $f : \{0,1\}^* \to \{0,1\}$ be a function, and $\mathcal{C}$ a family of Boolean circuits such that $f \notin \mathcal{C}$. We say that an algorithm A learns $\ell = \ell(n)$ counterexamples for f with respect to the concept class $\mathcal{C}$, if for every n for which $f \notin \mathcal{C}^n$, on input 1^n, the algorithm outputs a list of at most ℓ strings $x_1, \dots, x_\ell$ of n-bit length such that for every circuit $C \in \mathcal{C}^n$, there exists $1 \leq i \leq \ell$ such that $C(x_i) \neq f(x_i)$.

The following lemma shows that deterministic learning counterexamples for SAT implies exponential-size lower bounds.

Lemma 3.2 Suppose that for some $c > k > 4$ there is a deterministic algorithm A which uses an NP oracle and runs in time $\text{poly}(n)$, such that A learns a list of n^c counterexamples for SAT^n with respect to the concept class $\text{SIZE}(n^k)$, for infinitely many $n \in \mathbb{N}$. Then there is a constant $\delta > 0$ (that depends only on k and c), and a Boolean function in the class E^{NP} that cannot be computed by circuits of size $2^{\delta n}$, for infinitely many input lengths n.

Before we give the formal proof we briefly present the intuition. By the hypothesis, the algorithm A generates, in polynomial-time with access to an NP oracle, a list of counterexamples $(\phi_1, ..., \phi_\ell)$ for some polynomial $\ell(n) := n^c > n^k$. It holds that every n^k-size circuit fails on at least one instance in the list. It is tempting to take the function $f(i) := \text{SAT}(\phi_i)$ as our hard function. However, this is not necessarily true. Since we cannot assume any particular property regarding the order of the ϕ_i's, it is hypothetically possible that the location of a formula in the list determines its satisfiability (e.g., every even formula in the list is satisfiable and every odd is unsatisfiable). Furthermore, since $\ell > n^k$ a circuit of size n^k cannot necessarily determine the index of a formula from the formula itself, thus it is possible that the list is indeed hard for circuits of size n^k but f itself is easy. Instead we show that if f is easy for circuits of size n^k then the hardness of the counterexamples stems from the fact that it is hard to generate their description (under some canonical representation of Boolean formulas). That is, we show that the function $h(i,j) =$ [the j-th bit in the description of ϕ_i] is sufficiently hard for Boolean circuits. Let us proceed with the proof.

Proof. Fix a sufficiently large n so that no n^k-size circuit solves SAT^n. In this case, A outputs a list of $\ell(n)$ counterexamples. Let $(\phi_1, ..., \phi_\ell)$ be the list sorted in lexicographical order so that $\phi_1 < \cdots < \phi_\ell$. Define $m := \lceil \log \ell(n) \rceil \leq \lceil c \log n \rceil$.

In the sequel we define several functions on different input lengths. A superscript denotes the input length of each function. Consider the following function $g^n : \{0,1\}^n \to \{0,1\}^m$ defined as $g^n(\phi) = i$ if

$\phi = \phi_i$ for some $1 \le i \le \ell$, and $g^n(\phi) = 0$ otherwise. We consider two cases, whether (I) an n^{k-1}-size circuit can compute g^n or (II) not.

Case (I): In this case, we prove that no circuit of size $n^{k-1} \ge 2^{\frac{k-1}{c}(m-1)}$ can compute the function $f^m(i) = \mathrm{SAT}(\phi_i)$. Assume that some n^{k-1}-size circuit C_f can compute f^m. By the hypothesis of Case (I), we have an n^{k-1}-size circuit C_g that computes g^n. Using C_f and C_g, we can obtain an n^k-size circuit C that computes all of the counterexamples $\{\phi_1, ..., \phi_\ell\}$ correctly, which contradicts the hardness of the counterexamples. The circuit C is constructed as follows. Let ϕ be a given instance of n-bit length.

1. Run $C_g(\phi)$. If the output is 0, then output 0 and quit. Otherwise let $i \in [\ell]$ be the output of $C_g(\phi)$.
2. Output $C_f(i)$.

Obviously, the size of this circuit C is $2n^{k-1} \le n^k$ and it correctly computes $\mathrm{SAT}(\phi_i)$ for any i.

Moreover, the function f can be computed in $\mathrm{poly}(n) = 2^{O(m)}$ time using an NP oracle as follows. Let $i \in \{0,1\}^m$ be an input. 1. Run A and lexicographically sort the output formulas. The resulting list is $(\phi_1, ..., \phi_\ell)$. 2. Invoke the NP oracle to determine if $\phi_i \in \mathrm{SAT}$, and output the result.

Therefore, f^m is hard against $2^{\frac{k-1}{c}(m-1)}$-size circuits and computable in $2^{O(m)}$ time using an NP oracle.

Case (II): In this case, we prove that no n^{k-3}-size circuit can compute yet another function h defined as $h^{m'}(i,j) =$ [the j-th bit in the description of $\phi_i \in \{0,1\}^n$], where $m' := m + \lceil \log n \rceil = \lceil \log \ell(n) \rceil + \lceil \log n \rceil = O(\log n)$. For contradiction, we assume that $h^{m'}$ can be computed by an n^{k-3}-size circuit C_h. Then, we can compute g^n by a small circuit C' that uses C_h, contradicting the hardness of g^n, i.e., the hypothesis for Case (II).

The circuit C' computes g^n as follows. Let ϕ be a given SAT instance of n-bit length.

1. Perform a binary search on the list $(\phi_1, \dots, \phi_\ell)$ to find the index i such that $\phi = \phi_i$, if ϕ is in the list. Each comparison in the binary search, against the formula with index i', is done by computing $\phi_{i'} = (C_h(i', 1), \dots, C_h(i', n))$ and checking whether ϕ is lexicographically equal, larger or smaller than $\phi_{i'}$.
2. Output the obtained index i if the binary search succeeds, otherwise output 0.

The binary search can be implemented by a circuit of size $O(|C_h| n \log \ell) = O(n^{k-2} \log n)$. Therefore, the size of C' is at most n^{k-1}, which contradicts the hardness of g^n. Also, $h^{m'}$ can be computable in $\mathrm{poly}(n) = 2^{O(m')}$ time using an NP oracle as follows: Let $(i,j) \in \{0,1\}^{m'}$ be a given instance. 1. Run A and sort the output formulas. The resulting list is $(\phi_1, ..., \phi_\ell)$. 2. Output the j-th bit of ϕ_i.

Therefore, $h^{m'}$ is hard against $n^{k-3} \ge 2^{\frac{k-3}{c+1}m'}$-size circuits and computable in $\mathrm{poly}(n) = 2^{O(m')}$ time using an NP oracle.

We showed that either f^m or $h^{m'}$ has the required hardness for a fixed input length. By setting $\delta := \frac{k-3}{c+1}$, we get that either f or h is a hard function for circuits of size $2^{\delta r}$ for infinitely many $r \in \mathbb{N}$, while both functions are computable in deterministic time $2^{O(r)}$ with access to an NP oracle. $\square$

3.2 Learning Counterexamples with an Oracle to Promise AM

In this section we show how to deterministically learn counterexamples for SAT with a prAM oracle.

We will need an Arthur-Merlin protocol that lower bounds the size of any set that is recognizable by a small circuit. Such a protocol was given by Goldwasser and Sipser [12]. The formulation that we use is taken from [5].

Consider the following promise problem Π on inputs (C, a, ϵ), where C is a description of a Boolean circuit with m inputs, $0 \leq a \leq 2^m$ is given in binary, and $0 \leq \epsilon < 1$ is given in unary representation.

- **Yes instances:** $(C, a, \epsilon) \in \Pi^Y$ if $|C^{-1}(1)| \geq a$.
- **No instances:** $(C, a, \epsilon) \in \Pi^N$ if $|C^{-1}(1)| \leq (1-\epsilon)a$.

Lemma 3.3 [12, 5] There is an Arthur-Merlin protocol for Π that runs in time $\mathrm{poly}(|C|, m, 1/\epsilon)$. That is, $\Pi \in \mathrm{prAM}$.

We now present the deterministic algorithm that learns counterexamples for SAT with a prAM oracle. Our algorithm is based on ideas from [6, 9, 8].

Theorem 3.4 Suppose that for some $k > 4$, SAT $\notin \mathrm{SIZE}(n^{k+2})$. There is a promise problem $\Gamma \in \mathrm{prAM}$ and a polynomial-time deterministic oracle algorithm A, such that for every function $f : \{0,1\}^* \to \{0,1\}$ that agrees with Γ, for every input length n for which SAT^n does not have circuits of size n^{k+2}, A^f learns dn^k counterexamples for SAT^n with respect to the concept class $\mathrm{SIZE}(n^k)$ for some constant $d > 1$.

Proof. Let us first set up the following notation. For a Boolean circuit C on n inputs, define the circuit $S(C)$ that on an input formula ϕ of length n, attempts to find the lexicographic first satisfying assignment for ϕ, via the downward self-reducibility property of SAT using C to solve the SAT instances along the search path. If $S(C)$ finds a satisfying assignment it outputs the assignment and otherwise it outputs 0. For a list of satisfiable formulas $L = (\phi_1, \ldots \phi_\ell)$ each of description length n, let T_L be the set of n^k-size circuits that are consistent with L. Namely, $C \in T_L$ if and only if $S(C)$ finds a satisfying assignment for every $\phi_j \in L$.

With this notation we can now describe the learning algorithm. Below we will define several promise problems in prAM and allow the algorithm oracle access to all of them (or to functions that agree with them to be more accurate). We can then bind them to a single promise problem in prAM as discussed in Section 2.

The algorithm has two stages. The first stage runs in iterations. Every iteration step i, passes to step $i+1$ a list L_i of satisfiable formulas $\phi_1, \ldots, \phi_i$ each of description length n, as well as a number $1 \leq \gamma_i \leq 2^{n^k}$, where γ_i is an estimate for $|T_{L_i}|$ such that

$$|T_{L_i}| \leq \gamma_i \leq \left(1 - \frac{1}{n^2}\right)^{-1} |T_{L_i}|. \tag{1}$$

Initially we set $L_0 := \emptyset$. This means that T_{L_0} contains all the circuits of description length n^k, and we therefore set $\gamma_0 := 2^{n^k}$.

The algorithm works in such a way that for every $i > 0$, $|T_{L_i}| \leq \frac{4}{5}|T_{L_{i-1}}|$. Thus for some $I \leq \lceil (\log_{5/4} 2) \cdot n^k \rceil$, $T_{L_I} = \emptyset$ at which stage we will terminate the loop and with it the first stage of the algorithm.

In the second stage, the algorithm uses its oracle as a SAT solver (clearly SAT $\in$ prAM) to generate for every $\phi_j \in L_I$ the list of formulas that are queried along the search path (via the downward self-reducibility property of SAT) for the lexicographic first satisfying assignment to ϕ_j (recall that every ϕ_j is satisfiable). We may assume w.l.o.g. that all the formulas thus generated are of description length n. The algorithm outputs all these formulas as the list of counterexamples. By the fact that $T_{L_I} = \emptyset$ it follows that for every $C \in \mathrm{SIZE}(n^k)$, $S(C)$ fails to find a satisfying assignment for at least one $\phi_j \in L_I$. This means that C errs on at least one query along the search path for a satisfying assignment for ϕ_j, and this query appears in the list that the algorithm outputs. It therefore follows that the algorithm indeed outputs a list of counterexamples for the concept class SIZE(n^k).[5]

It remains to describe iteration step $i > 0$, given a list $L_{i-1} = (\phi_1, \dots, \phi_{i-1})$ and γ_{i-1} as above. We need some more notation. For a list L of satisfiable formulas all of the same length n and one additional satisfiable formula ρ of length n, it is clear that $T_{L \cup \{\rho\}} \subseteq T_L$. Define the set $G_{L,\rho}$ to be $T_L \setminus T_{L\cup\{\rho\}}$.

Fortnow, Pavan and Sengupta [9] used a probabilistic argument (similar to the one in [6]) to prove the following lemma.

Lemma 3.5 [9] If SAT^n cannot be computed by circuits of size n^{k+2}, then for every list L of satisfiable formulas each of length n, there exist a formula ϕ of length n such $|T_{L\cup\{\phi\}}| \leq \frac{2}{3}|T_L|$.

By this Lemma there exist ϕ, such that

$$|G_{L_{i-1},\phi}| \geq \frac{1}{3}|T_{L_{i-1}}| \geq \frac{1}{4}\left(1 - \frac{1}{n^2}\right)^{-1} |T_{L_{i-1}}| \geq \frac{1}{4}\gamma_{i-1}. \tag{2}$$

(Recall that $|T_{L_{i-1}}| \leq \gamma_{i-1} \leq (1 - \frac{1}{n^2})^{-1}|T_{L_{i-1}}|$.)

We would like to find such a formula ϕ and then set $L_i := L_{i-1} \cup \{\phi\}$. We will not achieve quite that, but we will show how to find a ϕ such that

$$|G_{L_{i-1},\phi}| \geq \frac{1}{5}\gamma_{i-1} \geq \frac{1}{5}|T_{L_{i-1}}|. \tag{3}$$

Claim 3.6 There is a promise problem $\Pi_1 \in$ prAM and a deterministic polynomial-time procedure, that when given an estimate γ_{i-1} for $|T_{L_{i-1}}|$ that satisfies Inequality (1) and oracle access to any function that agrees with Π_1, outputs a Boolean formula ϕ of length n that satisfies Inequality (3).

Proof. The instances of Π_1 are of the form $(1^m, (\rho_1, \dots, \rho_\ell), p, a)$, where $m, \ell > 0$ are arbitrary integers, $\rho_j \in \{0,1\}^{m^k}$ for every $1 \leq j \leq \ell$, $p \in \{0,1\}^b$ for some integer $0 \leq b \leq m$, and a is an integer between 0 and 2^{m^k} (in binary representation). We define Π_1 as follows:

- **Yes instances:** $(1^m, (\rho_1, \dots, \rho_\ell), p, a) \in \Pi_1^Y$ if $\rho_1, \dots, \rho_\ell$ are all satisfiable Boolean formulas and there exist $s \in \{0,1\}^{m-b}$ such that $\rho = p \circ s$ is a satisfiable formula and $|G_{(\rho_1,\dots,\rho_\ell),\rho}| \geq a$.
- **No instances:** $(1^m, (\rho_1, \dots, \rho_\ell), p, a) \in \Pi_1^N$ if either at least one of $\rho_1, \dots, \rho_\ell$ is not satisfiable, or for every $s \in \{0,1\}^{m-b}$, $\rho = p \circ s$ is not a satisfiable formula, or $\rho_1, \dots, \rho_\ell$ are all satisfiable and for every $s \in \{0,1\}^{m-b}$ for which $\rho = p \circ s$ is a satisfiable formula, $|G_{(\rho_1,\dots,\rho_\ell),\rho}| \leq (1 - \frac{1}{m^2})a$

Claim 3.7 $\Pi_1 \in$ prAM.

Proof. The protocol is as follows. Merlin sends a string $s \in \{0,1\}^{m-b}$. Let $\rho = p \circ s$. Merlin also sends satisfying assignments for all of $\rho_1, \dots, \rho_\ell, \rho$. If he fails to do so, Arthur rejects.

Define the circuit C (which both Merlin and Arthur construct on their own) that on input a description of a circuit B of size m^k, checks whether $S(B)$ finds a satisfying assignment to all of $\rho_1, \dots, \rho_\ell$ but fails to find a satisfying assignment to ρ. If so it outputs 1 and otherwise 0. Note that C computes the characteristic function of $G_{(\rho_1,\dots,\rho_\ell),\rho}$. Arthur and Merlin run the lower bound protocol

[5]Note that at the end of the first stage we already have a list of counterexamples, but those are counterexamples for the search circuits $S(C)$, $C \in$ SIZE(n^k). In fact the list after the end of the first stage is very easy for decision circuits because it only contains satisfiable formulas. This is the reason that we need the second stage.

from Lemma 3.3 on input $(C, a, \frac{1}{m^2})$. Arthur accepts/rejects according to whether he accepts/rejects the lower bound protocol.

It is easy to verify that the protocol runs in time that is polynomial in its input length. We next argue about the completeness and soundness.

Completeness: If $\rho_1, \dots, \rho_\ell$ are all satisfiable Boolean formulas and there exist $s \in \{0,1\}^{m-b}$ such that $\rho = p \circ s$ is a satisfiable formula and $|G_{(\rho_1,\dots,\rho_\ell),\rho}| \geq a$, then Merlin can find and send such an s as well as satisfying assignments to $\rho_1, \dots, \rho_\ell, \rho$, and then the completeness follows from the completeness of the lower bound protocol.

Soundness: If one of $\rho_1, \dots, \rho_\ell$ is not satisfiable or there is no s such that $p \circ s$ is satisfiable, then Arthur will reject after the first message of Merlin with probability 1. Otherwise for every s, for which $\rho = p \circ s$ is satisfiable, $|G_{(\rho_1,\dots,\rho_\ell),\rho}| \leq (1 - \frac{1}{m^2})a$, and the soundness follows from the soundness of the lower bound protocol. □

We show how to find, with the help of Π_1, a formula ϕ that satisfies Inequality (3). We will do that iteratively where in each iteration we will set another bit of ϕ. Let $\mu_0 := \lfloor \frac{1}{4}\gamma_{i-1} \rfloor$. Recall, by Lemma 3.5, that there exist a formula that satisfies Inequality (2). The most significant bit of such a formula is either 0 or 1. In other words at least one of the following is true $(1^n, L_{i-1}, 0, \mu_0) \in \Pi_1^Y$ and/or $(1^n, L_{i-1}, 1, \mu_0) \in \Pi_1^Y$. We query the Π_1 oracle on the input $(1^n, L_{i-1}, 0, \mu_0)$. If the answer is 1 we set the MSB of ϕ to 0, otherwise we set it to 1. Note that if we set the MSB to 1 then necessarily it is the MSB of a formula that satisfies Inequality (2). However, if we set it to 0, this is not necessarily the case. The reason is that the query $(1^n, L_{i-1}, 0, \mu_0)$ may fall outside the promise of Π_1. What we are assured of though is that if the Π_1 oracle answered 1 on $(1^n, L_{i-1}, 0, \mu_0)$ then it is not in Π_1^N. That is, there is a satisfiable formula ϕ whose MSB is 0 such that $|G_{L_{i-1},\phi}| \geq (1 - \frac{1}{n^2})\mu_0$. We set $\mu_1 = (1 - \frac{1}{n^2})\mu_0$ and continue. In the j'th iteration, suppose that we already fixed a prefix p of length $j-1$ such that we know that there is a suffix that creates a satisfiable formula $\phi = p \circ s$ for which $|G_{L_{i-1},\phi}| \geq \mu_{j-1}$, then we query the Π_1 oracle on $(1^n, L_{i-1}, p \circ 0, \mu_{j-1})$ and set the next bit to 0 if the answer is 1 and otherwise we set it to 1. By the same argument as above we are guaranteed that the new prefix has a suffix that together they create a formula ϕ for which $|G_{L_{i-1},\phi}| \geq (1 - \frac{1}{n^2})\mu_{j-1}$. We then set $\mu_j := (1 - \frac{1}{n^2})\mu_{j-1}$ and continue. After n iterations we hold a formula ϕ of length n such that

$$|G_{L_{i-1},\phi}| \geq \left(1 - \frac{1}{n^2}\right)\mu_n \geq \left(1 - \frac{1}{n^2}\right)^n \mu_0 \geq \frac{1}{4}\left(1 - \frac{1}{n^2}\right)^n \gamma_{i-1} \geq \frac{1}{5}\gamma_{i-1} \geq \frac{1}{5}|T_{L_{i-1}}|.$$

□

By the claim above we can find ϕ that satisfies Inequality (3). We then set $L_i = L_{i-1} \cup \{\phi\}$. By the definition of $|G_{L_{i-1},\phi}|$, we get that $|G_{L_{i-1},\phi}| = |T_{L_{i-1}} \setminus T_{L_i}| \geq \frac{1}{5}|T_{L_{i-1}}|$ which implies that $|T_{L_i}| \leq \frac{4}{5}|T_{L_{i-1}}|$ as required.

Next we show how to compute an estimate γ_i for $|T_{L_i}|$. First we check whether $T_{L_i} = \emptyset$, in which case we terminate the main loop and move to the second stage of the algorithm. Note that this is a coNP statement: for every C of description length n^k, $S(C)$ fails to find a satisfying assignment for at least one $\phi_j \in L_i$. Thus we can query the prAM oracle to check that. If $T_{L_i} \neq \emptyset$, the next claim shows that we can compute γ_i as required (with oracle to prAM). This completes the description of the i'th iteration and hence the description of the algorithm.

Claim 3.8 There is a promise problem $\Pi_2 \in$ prAM and a deterministic polynomial-time procedure that when given oracle access to any function that agrees with Π_2 outputs a number $\gamma_i \in [2^{n^k}]$, such that $|T_{L_i}| \leq \gamma_i \leq (1 - \frac{1}{n^2})^{-1}|T_{L_i}|$.

Proof. Let Π_2 be the following promise problem on instances $(1^m, (\rho_1, \ldots, \rho_\ell), a)$, where $m, \ell > 0$ are arbitrary integers, $\rho_1, \ldots, \rho_\ell$ are all of length m, and a is an integer between 0 and 2^{m^k} (in binary representation):

- **Yes instances:** $(1^m, (\rho_1, \ldots, \rho_\ell), a) \in \Pi_2^Y$ if $\rho_1, \ldots, \rho_\ell$ are all satisfiable formulas and $|T_{(\rho_1,\ldots,\rho_\ell)}| \geq a$.
- **No instances:** $(1^m, (\rho_1, \ldots, \rho_\ell), a) \in \Pi_2^N$ if either at least one of $\rho_1, \ldots, \rho_\ell$ is not a satisfiable formula, or they are all satisfiable and $|T_{(\rho_1,\ldots,\rho_\ell)}| \leq (1 - \frac{1}{m^2})a$.

Claim 3.9 $\Pi_2 \in \mathrm{prAM}$.

Proof. The protocol is as follows. Merlin sends satisfying assignments for all of $\rho_1, \ldots, \rho_\ell$. If he fails to do so, Arthur rejects.

Define the circuit C (which both Merlin and Arthur construct on their own) that on input a description of a circuit A of size n^k, checks whether $S(A)$ finds satisfying assignments to all of the formulas $\rho_1, \ldots, \rho_\ell$. If so it outputs 1 and otherwise 0. Note that C computes the characteristic function of $T_{(\rho_1,\ldots,\rho_\ell)}$. Arthur and Merlin run the lower bound protocol from Lemma 3.3 on input $(C, a, \frac{1}{m^2})$. Arthur accepts/rejects according to whether he accepts/rejects the lower bound protocol.

It is easy to verify that the protocol runs in time that is polynomial in its input length. We next argue about the completeness and soundness.

Completeness: If $\rho_1, \ldots, \rho_\ell$ are all satisfiable Boolean formulas such that $|T_{(\rho_1,\ldots,\rho_\ell)}| \geq a$, then Merlin can find and send satisfying assignments to $\rho_1, \ldots, \rho_\ell$, and then the completeness follows from the completeness of the lower bound protocol.

Soundness: If one of $\rho_1, \ldots, \rho_\ell$ is not satisfiable then Arthur will reject after the first message of Merlin with probability 1. Otherwise, $|T_{(\rho_1,\ldots,\rho_\ell)}| \leq (1 - \frac{1}{m^2})a$, and the soundness follows from the soundness of the lower bound protocol. □

By the definition of Π_2, we know that for every $a \leq |T_{L_i}|$ a Π_2 oracle answers 1 on the query $(1^n, L_i, a)$, and for every $a \geq (1 - \frac{1}{n^2})^{-1}|T_{L_i}|$ a Π_2 oracle answers 0 on the query $(1^n, L_i, a)$. For values of a in between these two bounds we have no guarantee on the oracle's answers. The algorithm conducts a binary search on the set $[2^{n^k}]$ to find an a such that the Π_2 oracle answers 0 on $(1^n, L_i, a)$ but 1 on $(1^n, L_i, a-1)$ (forcing the answer on $(1^n, L_i, 0)$ to be 1). Such a search takes $O(n^k)$ time and we are guaranteed that for the a that we find,

$$|T_{L_i}| \leq a \leq \left(1 - \frac{1}{n^2}\right)^{-1} |T_{L_i}|.$$

We then set $\gamma_i := a$. □

□

4 Derandomization Implies Exponential-Size Lower Bounds

4.1 Arthur-Merlin Games

We now prove Theorem 1.1.

Proof. We condition on whether SAT $\in$ SIZE(n^{10}) or not.

Case 1: SAT $\in$ SIZE(n^{10}). By hypothesis, $\mathrm{P}^{\mathrm{prAM}} = \mathrm{P}^{\mathrm{P}^{\mathrm{NP}}} = \mathrm{P}^{\mathrm{NP}}$. For every $0 < \delta < 1$, Kannan [17] showed that on an input 1^n, we can compute in Σ_3^{P} the lexicographic first truth-table

of length n ($n = 2^m$) of a function (on m inputs) whose circuit complexity is at least n^δ. By [8], if SAT $\in$ SIZE(n^{10}), the polynomial-time hierarchy collapses to $\mathrm{P}^{\mathrm{prAM}}$, and hence to P^{NP} by our hypothesis. In particular $\Sigma_3^{\mathrm{P}} \subseteq \mathrm{P}^{\mathrm{NP}}$ and we can compute the truth-table of the hard function in this class. By translation to the exponential level, this implies that there is a function in E^{NP} that cannot be computed by circuits of size $2^{\delta n}$ (for all sufficiently large n).
Case 2: SAT $\notin$ SIZE(n^{10}). Let Γ be the promise problem in prAM from Theorem 3.4. By hypothesis, prAM $\subseteq \mathrm{P}^{\mathrm{NP}}$ so there is a function $f : \{0,1\}^* \to \{0,1\}$ in P^{NP} that agrees with Γ. By Theorem 3.4 there is a polynomial-time deterministic oracle algorithm A, such that for every input length n for which SAT^n does not have circuits of size $O(n^{10})$, A^f learns a poly(n)-long list of counterexamples for SAT^n with respect to the concept class SIZE(n^8). The function that A^f computes is in the class $\mathrm{P}^{\mathrm{P}^{\mathrm{NP}}} = \mathrm{P}^{\mathrm{NP}}$. This implies by Lemma 3.2, that there is a constant $\delta > 0$ and a Boolean function in the class E^{NP} that cannot be computed by circuits of size $2^{\delta n}$ (for infinitely many input lengths n). □

4.2 The Goldwasser-Sipser Protocol and Approximate Counting

A close inspection of our proof, and the proofs that it relies on (namely, [8]), reveals that the algorithm that computes the hard function asks the oracle questions of the following form: let P be some easily computable property of Boolean circuits. Given a circuit C with n inputs, a number $1 \leq a \leq 2^n$ and $0 \leq \epsilon \leq 1$, distinguish between the case that there is a Boolean circuit $C \in P$ such that $(C, a, \epsilon) \in \Pi^Y$ and the case that for every $C \in P$, $(C, a, \epsilon) \in \Pi^N$. Where Π is the promise problem from Lemma 3.3. (The algorithm also asks standard NP questions but these can also be stated as instances of Π: distinguish between the case $|C^{-1}(1)| \geq 1$ and the case $|C^{-1}(1)| = 0 \leq 1 - \epsilon$.) A conclusion is that it is enough to derandomize the Goldwasser-Sipser protocol (so it can be done in NP) to get the lower bound. This gives Theorem 1.2.

Theorem 1.4 follows in a similar way as we now explain. Let us first formally define the problem of approximate counting.

Definition 4.1 The problem of relative-error approximate counting is defined as follows:
input: A Boolean circuit C on n inputs and a parameter $0 < \epsilon < 1$.
output: A number $\gamma \in [2^n]$ such that $(1-\epsilon)|C^{-1}(1)| \leq \gamma \leq (1-\epsilon)^{-1}|C^{-1}(1)|$.

We say that relative-error approximation can be done in nondeterministic polynomial-time if there exist a nondeterministic TM that on input C and ϵ runs in nondeterministic poly($|C|, 1/\epsilon$)-time, has at least one accepting path, and on each accepting path outputs an estimate γ as above (γ may differ from one accepting path to another, as long as the condition holds).

It is easy to see that if a relative-error approximator can be computed in nondeterministic polynomial-time then the oracle described above can be implemented in NP: guess a circuit C and check whether $C \in P$, if not reject. Then choose an ϵ' such that $\epsilon > 1 - (1-\epsilon')^2$, and run the nondeterministic procedure for the approximator with parameters C and ϵ'. Let γ be its answer (unless it rejects in which case the outer procedure rejects), then accept if and only if $\gamma \geq (1-\epsilon')a$.

If there exist a Boolean circuit $C \in P$ such that $(C, a, \epsilon) \in \Pi^Y$, then by the definition of Π, $|C^{-1}(1)| \geq a$. It hence holds that $\gamma \geq (1-\epsilon')|C^{-1}(1)| \geq (1-\epsilon')a$ for the γ that is returned by the approximator and we will accept. On the other hand, if for every $C \in P$, $(C, a, \epsilon) \in \Pi^N$, it must be the case that (if C passed the first test of being in P), $\gamma \leq (1-\epsilon')^{-1}(1-\epsilon)a < (1-\epsilon')a$, and we will reject.

Theorem 1.4 now immediately follows.

5 Concluding Remarks

Our proof shows that there is a deterministic exponential-time algorithm A and a promise problem in prAM, such that for every function f that agrees with it, A^f computes a function that cannot be computed by circuits of size $2^{\delta n}$. The dependence on the specific f comes from the fact that the counterexamples in Theorem 3.4 depend on the values of the function outside the promise. This is the reason that we do not get a lower bound for an explicit function in the class $\mathrm{E}^{\mathrm{prAM}}$. Recall that a function is in this class if for *every* oracle that agrees with the promise, the algorithm computes the same function (i.e., the values of the function do not depend on values of the oracle outside the promise). Nevertheless, our proof does imply the best *known* exponential-size lower bound, namely the one for the class $\mathrm{E}^{\Sigma_2^{\mathrm{P}}}$. This is because the Π_2^{P} simulation of our prAM oracle [10] gives an explicit function in $\mathrm{E}^{\Sigma_2^{\mathrm{P}}}$ that requires circuits of size $2^{\Omega(n)}$. Proving an exponential-size lower bound for the class $\mathrm{E}^{\mathrm{prAM}}$ (and thus improving the best known lower bound) remains an open problem.

Another interesting open problem is to prove a true converse to [22, 26, 30]. Namely, show that a full derandomization of prAM implies lower bounds against exponential-size *nondeterministic* circuits.

Acknowledgments

We thank Scott Aaronson, Amnon Ta-Shma, Ronen Shaltiel and Osamu Watanabe for helpful comments and discussions.

References

[1] Vikraman Arvind and Partha Mukhopadhyay. Derandomizing the isolation lemma and lower bounds for circuit size. In *Proceedings of the 11th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2008) and the 12th International Workshop on Randomization and Computation (RANDOM 2008)*, LNCS 5171, pages 276–289, 2008.

[2] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential simulation unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[3] László Babai and Shlomo Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[4] Mihir Bellare, Oded Goldreich, and Erez Petrank. Uniform generation of NP-witnesses using an NP-oracle. *Information and Computation*, 163:2000, 510–526.

[5] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM Journal on Computing*, 36(4):1119–1159, 2006.

[6] Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52(3):421–433, 1996.

[7] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Proceedings of the 13rd Annual IEEE Conference on Computational Complexity*, pages 8–12, 1998.

[8] Venkatesan T. Chakaravarthy and Sambuddha Roy. Finding irrefutable certificates for S_2^p via Arthur and Merlin. In *Proceedings of the 25th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 157–168, 2008.

[9] Lance Fortnow, A. Pavan, and Samik Sengupta. Proving SAT does not have small circuits with an application to the two queries problem. *Journal of Computer and System Sciences*, 74(3):358–363, 2008.

[10] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research 5: Randomness and Computation*, pages 429–442, 1989.

[11] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[12] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof system. In Silvio Micali, editor, *Advances in Computing Research, Vol. 5: Randomness and Computation*, pages 73–90. JAI Press, 1989.

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

[14] Russell Impagliazzo and Avi Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[15] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

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

[17] R. Kannan. Circuit-size lower bound and non-reducibility to sparse sets. *Information and Control*, 55(1–3):40–56, 1982.

[18] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual Symposium on Theoretical Computer Science*, pages 302–309, 1980.

[19] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators and typically-correct derandomization. In *Proceedings of the 12th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2009) and the 13th International Workshop on Randomization and Computation (RANDOM 2009)*, pages 574–587, 2009.

[20] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. 31(5):1501–1526, 2002.

[21] Johannes Köbler and Osamu Watanabe. New collapse consequences of NP having small circuits. *SIAM Journal on Computing*, 28(1):311–324, 1998.

[22] Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[23] Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *Proceedings of the 5th Annual International Conference on Computing and Combinatorics (COCOON)*, pages 210–220, 1999.

[24] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11:63–70, 1991.

[25] Rahul Santhanam. Circuit lower bounds for Merlin-Arthur classes. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 275–283, 2007.

[26] Ronen Shaltiel and Chris Umans. Simple extractors for all min-entropies and a new pseudo-random generator. *Journal of ACM*, 52(2):172–216, 2005.

[27] Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006.

[28] Larry J. Stockmeyer. The complexity of approximate counting. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 118–126, 1983.

[29] S. Toda. On the computational power of PP and $\oplus$P. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 514–519, 1989.

[30] Chris Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.

[31] Emanuele Viola. Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SIAM Journal on Computing*, 36(5):1387–1403, 2007.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de