Electronic Colloquium on Computational Complexity, Report No. 5 (2007)

Circuit Lower Bounds for Merlin-Arthur Classes

Rahul Santhanam
Simon Fraser University
rsanthan@cs.sfu.ca

January 16, 2007

Abstract

We show that for each $k > 0$, $\mathsf{MA}/1$ (MA with 1 bit of advice) doesn't have circuits of size n^k. This implies the first superlinear circuit lower bounds for the promise versions of the classes MA, AM and $\mathsf{ZPP}_{\|}^{\mathsf{NP}}$.

We extend our main result in several ways. For each k, we give an explicit language in $(\mathsf{MA} \cap \mathsf{coMA})/1$ which doesn't have circuits of size n^k. We also adapt our lower bound to the average-case setting, i.e., we show that $\mathsf{MA}/1$ cannot be solved on more than $1/2 + 1/n^k$ fraction of inputs of length n by circuits of size n^k. Furthermore, we prove that MA does not have arithmetic circuits of size n^k for any k.

As a corollary to our main result, we obtain that derandomization of MA with $O(1)$ advice implies the existence of pseudo-random generators computable using $O(1)$ bits of advice.

1 Introduction

Proving circuit lower bounds within uniform complexity classes is one of the most fundamental and challenging tasks in complexity theory. Apart from clarifying our understanding of the power of non-uniformity, circuit lower bounds have direct relevance to some longstanding open questions. Proving super-polynomial circuit lower bounds for NP would separate P from NP. The weaker result that for each k there is a language in NP which doesn't have circuits of size n^k would separate BPP from NEXP, thus answering an important question in the theory of derandomization.

However, as of now, there is no superlinear circuit lower bound known for any language in NP. Researchers have attempted to understand the difficulty of proving lower bounds by formalizing "obstacles" to traditional techniques, such as the relativization obstacle [BGS75] and the naturalness obstacle [RR97].

Given these obstacles, we are forced to temper our ambitions. There are two distinct lines of research which have made incremental progress over the years toward the ultimate goal of a non-trivial circuit lower bound for NP. In the first line of research, lower bounds were proved in restricted circuit models [Ajt83, FSS84, Hås86, Raz85, Raz87, Smo87]. The hope is that by gradually easing the restrictions on the circuit model, we will ultimately achieve a lower bound in the general model. This line of research has made little progress in the last decade and more, with many of the commonly used techniques unable to circumvent the naturalness obstacle.

A second line of research has had more success in recent years. The aim here is to "approach NP from above" by proving fixed polynomial size circuit lower bounds in the general model for smaller and smaller classes containing NP. To compare with the first line of research, we are satisfied with fixed polynomial size lower bounds rather than superpolynomial lower bounds, for languages that

ISSN 1433-8092

are computable in classes not "too far above" NP, but we insist that the lower bounds hold in the model of general Boolean circuits. This line of research was initiated by Kannan [Kan82] who showed that for each k, there is a language in $\Sigma_2 \cap \Pi_2$ which doesn't have circuits of size n^k. This was improved by Kobler and Watanabe [KW98], who used the learning algorithm of Bshouty et al. [BCG+96] to prove n^k size circuit lower bounds for $\mathsf{ZPP}^{\mathsf{NP}}$, which is contained in $\Sigma_2 \cap \Pi_2$. This was further improved to a lower bound for the class $\mathsf{S}_2\mathsf{P}$ by Cai [Cai01], based on an observation by Sengupta. Both $\mathsf{S}_2\mathsf{P}$ and $\mathsf{ZPP}^{\mathsf{NP}}$ are believed to equal P^{NP}, under a strong enough derandomization assumption.

An incomparable result was recently obtained by Vinodchandran [Vin05], who showed n^k size lower bounds for the class PP of languages accepted by probabilistic polynomial time machines with unbounded error. Aaronson [Aar05] gave a different proof of the same result.

The logical next step would be to prove circuit lower bounds for the class MA, the probabilistic version of the class NP. Such lower bounds would be of interest because MA equals NP under a widely believed derandomization assumption, and hence they could be construed as making substantial progress towards proving circuit lower bounds for NP. Moreover, such a result would simulataneously strengthen all known lower bounds in this line of research, since it is known that $\mathsf{MA} \subseteq \mathsf{S}_2\mathsf{P}$ [RS98, GZ97] and that $\mathsf{MA} \subseteq \mathsf{PP}$ [Ver92]

In the present work, we do not quite achieve this, but we achieve something very close. We show n^k size lower bounds for languages accepted by Merlin-Arthur machines running in polynomial time and using just *one bit of non-uniformity*.

Theorem 1. *For each k,* $\mathsf{MA}/1 \not\subseteq \mathsf{SIZE}(n^k)$.

Theorem 1 implies that for each k there is a *promise problem* (Y, N) in MA (where $Y, N \subseteq \{0,1\}^*$, $Y \cap N = \varnothing$ and the Merlin-Arthur machine is required to accept on instances in Y, reject on instances in N and may have arbitrary behavior on other instances), which does not have circuits of size n^k. To the best of our knowledge, this is the first natural example of a circuit lower bound for a promise problem in a class which does not immediately imply a corresponding lower bound for a language in the class. We discuss this issue further in Section 2.

We now briefly discuss previous techniques in this line of research and indicate in what respect our approach is novel.

Suppose we wish to prove a lower bound for a class C, where $\mathsf{NP} \subseteq \mathsf{C}$. First, we observe that a language without n^k size circuits can be computed in the polynomial hierarchy (PH) directly using diagonalization. Then we argue based on whether $\mathsf{NP} \subseteq \mathsf{SIZE}(poly)$ or not. If $\mathsf{NP} \not\subseteq \mathsf{SIZE}(poly)$, since $\mathsf{NP} \subseteq \mathsf{C}$, we already have a super-polynomial lower bound for a language in C. If $\mathsf{NP} \subseteq \mathsf{SIZE}(poly)$, we try to show that this implies a collapse of PH to our class C, and hence the diagonalizing language is in C, implying a lower bound.

Thus the strength of the result we obtain is directly related to the strength of the collapse consequence of NP having small circuits. Karp and Lipton [KL82] showed that $\mathsf{NP} \subseteq \mathsf{SIZE}(poly)$ implies $\mathsf{PH} \subseteq \Sigma_2 \cap \Pi_2$ - this yields Kannan's result [Kan82]. Strengthenings of the Karp-Lipton theorem [BCG+96, Cai01] yield the lower bounds for $\mathsf{ZPP}^{\mathsf{NP}}$ and $\mathsf{S}_2\mathsf{P}$ respectively. Thus the natural way to get lower bounds for MA would be to show a Karp-Lipton style collapse of PH to MA.

However, this is a notoriously difficult problem that has long been open; also, unlike known Karp-Lipton style consequences of $\mathsf{NP} \subseteq \mathsf{SIZE}(poly)$, such a collapse would not relativize [BFT98]. We circumvent this problem entirely and adopt a different, somewhat counter-intuitive approach. Instead of reasoning based on whether $\mathsf{NP} \subseteq \mathsf{SIZE}(poly)$, we reason based on whether $\mathsf{C} \subseteq \mathsf{SIZE}(poly)$ for a "large" complexity class C that *contains* MA - in our proof, $\mathsf{C} = \mathsf{PSPACE}$. The advantage of doing this is that for such "large" complexity classes, a Karp-Lipton style collapse

to MA is known, following from work on interactive proofs [LFKN92, Sha92, BFL91]. In the case PSPACE $\subseteq$ SIZE($poly$), we are done, since PSPACE = MA in this case, and PSPACE $\supseteq$ PH contains languages of circuit complexity at least n^k.

However, the case PSPACE $\not\subseteq$ SIZE($poly$), which was easy when we were arguing about NP instead, is now far from straightforward. We do not know if PSPACE $\subseteq$ MA, and indeed this is very unlikely, so we cannot conclude directly that MA doesn't have polynomial-size circuits. Our main contribution is to show how to use a stronger "parametrized" version of a Karp-Lipton style collapse in this case also. Remarkably, a single bit of advice to an MA machine gives us enough power to get a circuit lower bound in this case.

We use variations of our technique to make progress on other problems in the area of fixed polynomial circuit lower bounds. One interesting question is to find *explicit* languages with circuit lower bounds - this is known for the case of Σ_2 [CW04] but not for any lower class. We improve the situation considerably for *promise problems*, by giving explicit promise problems in MA $\cap$ coMA with circuit lower bounds.

Theorem 2. *For each k, there is an explicit promise problem (Y, N) in* MA $\cap$ coMA *such that (Y, N) doesn't have circuits of size n^k.*

We also give stronger *average-case* circuit lower bounds than were known previously. Since we don't know an average-case to worst-case reduction for NP, arguments based on whether NP $\subseteq$ SIZE($poly$) are not particularly well-suited to showing average-case lower bounds. Before this, the best known average-case lower bounds were for languages in P^{Σ_2}. We show average-case circuit lower bounds for (MA $\cap$ coMA)/1 and for Σ_2.

Theorem 3. *For each k,* (MA $\cap$ coMA)/1 $\not\subseteq$ $heur_{1/2+1/n^k}$ − SIZE(n^k).

Given the difficulty of proving Boolean circuit lower bounds, there's been a lot of work on using the structure inherent in algebraic domains to prove arithmetic circuit lower bounds, which tend to be weaker. Scott Aaronson [Aar06] pointed out to us that our technique could be adapted to show arithmetic circuit lower bounds for MA *without* any advice. In this context, we say that MA doesn't have arithmetic circuits of size n^k if either MA doesn't have Boolean circuits of size n^k or there is some sequence of polynomials p of low degree over $\mathbb{Z}$ such that the graph of p is computable in MA and p doesn't have arithmetic circuits of size n^k.

Theorem 4. *For any $k > 0$,* MA *doesn't have arithmetic circuits of size n^k over $\mathbb{Z}$.*

Theorem 1 also has applications in the theory of derandomization. It can be used to show that any derandomization of MA/1 implies a pseudo-random generator computable using 2 bits of advice, as well as a "gap theorem" for MA/$O(1)$.

Finally, we note that the techniques we use neither relativize nor naturalize, and hence there is no negative evidence that they cannot be pushed to yield Boolean circuit lower bounds for uniform MA or even for NP. This issue is discussed in more detail in Section 7.

2 Preliminaries

2.1 Complexity Classes, Promise Problems and Advice

We assume a basic familiarity with complexity classes such as P, RP, BPP, NP, MA, AM, Σ_2,PP, $\sharp P$ and PSPACE. The Complexity Zoo (`http://qwiki.caltech.edu/wiki/Complexity_Zoo`) is an excellent resource for basic definitions and statements of results.

Given a complexity class C, coC is the class of languages L such that $\bar{L} \in \mathsf{C}$. Given a function $s : \mathbb{N} \to \mathbb{N}$, $\mathsf{SIZE}(s)$ is the class of Boolean functions $f = \{f_n\}$ such that for each n, f_n has Boolean circuits of size at most $s(n)$. Given a language L and an integer n, $L|_n = L \cap \{0,1\}^n$.

In order to deal with promise classes in a general way, we take as fundamental the notion of a complexity measure. A complexity measure CTIME is a mapping which assigns to each pair (M, x), where M is a time-bounded machine (here a time function $t_M(x)$ is implicit) and x an input, one of three values "0" (accept), "1" (reject) and "?" (failure of CTIME promise). We distinguish between *syntactic* and *semantic* complexity measures. Syntactic measures have as their range $\{0,1\}$ while semantic measures may map some machine-input pairs to "?". The complexity measures DTIME and NTIME are syntactic (each halting deterministic or non-deterministic machine either accepts or rejects on each input), while complexity measures such as BPTIME and MATIME are semantic (a probabilistic machine may accept on an input with probability $1/2$, thus failing the bounded-error promise). For syntactic measures, any halting machine defines a language, while for semantic measures, only a subset of halting machines define languages.

A promise problem is a pair (Y, N), where $Y, N \subseteq \{0,1\}^*$ and $Y \cap N = \varnothing$. We say that a promise problem (Y, N) belongs to a class $\mathsf{CTIME}(t)$ if there is a machine M halting in time t on all inputs of length n such that M fulfils the CTIME promise on inputs in $Y \cup N$, accepting on inputs in Y and rejecting on inputs in N.

Note that for a syntactic complexity measure CTIME, a circuit lower bound for a promise problem implies the same lower bound for a language, since the machine M deciding the promise problem also defines a language in this case.

For a complexity class C, $\mathsf{Promise-C}$ is the class of promise problems which belong to C. Sometimes, when C is a syntactic class, we abuse notation and use C and $\mathsf{Promise-C}$ interchangeably.

A language L is in $\mathsf{CTIME}(t)/a$ if there is a machine M halting in time t taking an auxiliary *advice* string of length a such that for each n, there is some advice string $b_n, |b_n| = a$ such that M fulfils the CTIME promise for each input x with advice string b_n and accepts x iff $x \in L$.

2.2 Average-Case Complexity

We define a notion of what it means for a language to be solvable on average under the uniform distribution.

Definition 5. *Given a language L and functions $s : \mathbb{N} \to \mathbb{N}$ and $q : \mathbb{N} \to [0,1]$, $L \in heur_q - \mathsf{SIZE}(s)$ if for each n, there is a circuit C_n of size at most $s(n)$ that solves $L|_n$ on at least a $q(n)$ fraction of inputs.*

Random self-reducibility is a property of a language which ensures that its worst case complexity is not too much greater than its average case complexity. We require a slightly more general notion.

Definition 6. *A language L is piecewise random self-reducible with n^b queries and piecewise density at least $1/n^c$ if for each n, there is a partition $S_1 \ldots S_k$ of $\{0,1\}^n$ such that for each i, $|S_i| \geqslant 2^n/n^c$, and there are query functions $q(x,j,r), j = 1 \ldots n^b, |r| \leqslant poly(|x|)$ and combiner function f computable in deterministic polynomial time such that the following hold:*

1. *For each $i, 1 \leqslant i \leqslant k$, for any $x \in S_i$, each query function q(x,j,r) is distributed uniformly over S_i when r is chosen at random.*
2. *For each x, $L(x) = f(L(q(x,1,r)), L(q(x,2,r)) \ldots L(q(x,n^b,r)))$ with probability at least $1 - 2^{-|x|}$ over the choice of r.*

2.3 Arithmetic Complexity

An arithmetic circuit C with variables $x_1, x_2 \dots x_n$ over an integral domain I is a node-labelled directed acyclic graph in which each vertex has indegree 2 or 0. Each of the sources (nodes of degree 0) is labelled with x_i, for some $1 \leqslant i \leqslant n$, or with a constant in I. Each of the interior nodes is labelled with '*', '+' or '-'. We interpret each node as computing a polynomial over I. A source node labelled x_i computes the polynomial x_i; a source node labelled with a constant c computes the polynomial c. Polynomials computed by interior nodes are defined by induction (over the sum of the depths of the node's predecessors). If the predecessors of a node v labelled '*' compute p_1 and p_2 respectively, then v computes $p_1 * p_2$, and analogously for nodes labelled '+' and '-'. There is a distinguished sink node called the output node; the polynomial computed by the output node is defined to be the polynomial p_C defined by the circuit. The size of a circuit C is the number of nodes in the graph, and is denoted by $size(C)$.

There are two possible definitions for whether a polynomial p is computed by an arithmetic circuit C - p is the same as p_c as a sum of monomials (modulo the order of monomials), or p agrees with p_C over all inputs in I^n. We only consider infinite domains I, in which case the two definitions are equivalent.

A sequence $p = p_n$ of polynomials, where each p_n is on n variables, has arithmetic circuits of size $s(n)$ over I if for each n, there is an arithmetic circuit C_n computing p_n of size at most $s(n)$. We say p is of feasible degree if the degree of p_n is at most $poly(n)$. We will work only with polynomials of feasible degree.

Given an arithmetic circuit, a natural algorithmic task is Arithmetic Circuit Identity Testing (ACIT) - testing if the circuit computes the zero polynomial. The Schwartz-Zippel lemma [Sch80, R.E79] can be used to show that ACIT over $\mathbb{Z}$ is in coRP.

Lemma 7. *[IM83] ACIT over $\mathbb{Z}$ is in* coRP*.*

One of the most well-studied polynomials is the *permanent*, which is conjectured not have polynomial-size arithmetic circuits [Val79a]. The permanent PER is defined on n^2 variables $\{x_{ij}\}, i = 1 \dots n, j = 1 \dots n$ as follows: $PER(\vec{x}) = \sum_{\pi} \prod_{i=1}^{n} x_{i\pi(i)}$, where the sum is taken over all permutations π on $\{1 \dots n\}$. We also define 0-1-PER, which is PER restricted to inputs in $\{0, 1\}$.

Apart from its significance in algebraic complexity, the permanent has interesting properties in the Boolean setting, such as being complete for the class of functions counting the number of accepting paths of a non-deterministic machine.

Theorem 8. *[Val79b] 0-1-PER over $\mathbb{Z}$ is complete for $\sharp P$.*

Kabanets and Impagliazzo [KI04] showed that checking if an arithmetic circuit computes PER over $\mathbb{Z}$ is many-one reducible to ACIT over $\mathbb{Z}$, and is hence in coRP, using Lemma 7.

Lemma 9. *[KI04] The language $\{\langle C, 1^n\rangle | C$ computes PER on n^2 variables over $\mathbb{Z}\}$ is in* coRP*.*

We require a notion of computing a sequence of polynomials in a uniform complexity class.

Definition 10. *Given a family of functions $f = \{f_n\}$, where each $f_n : \mathbb{Z}^n \to \mathbb{Z}$, the graph of f is the language $Gh(f) = \{\langle \vec{x}, v\rangle | f(\vec{x}) = v\}$*

Definition 11. *A sequence of polynomials $p = \{p_n\}$ is said to be computable in* MA *if $Gh(p) \in$* MA*. We say* MA *does not have arithmetic circuits of size $s(n)$ if either* MA *does not have Boolean circuits of size $s(n)$ or there is a sequence of polynomials p of feasible degree computable in* MA *such that p does not have arithmetic circuits of size $s(n)$.*

A couple of remarks are in order here. First, the definition can clearly be generalized to any uniform complexity class, but we will only be concerned with MA in this paper. Second, the clause that "MA does not have Boolean circuits of size $s(n)$" may seem a bit unnatural initally. However, we regard Boolean circuit lower bounds as harder than arithmetic circuit lower bounds, and hence we are satisfied if we obtain Boolean circuit lower bounds for a Boolean function in MA, even if it is unclear how to extend the function to a polynomial whose graph is in MA.

3 Main Result

In this section, we prove Theorem 1.

We need the following technical result which follows from work on interactive proofs [LFKN92, Sha92, TV02, FS04]. In the jargon of program checking [BK95], the result states that there is a PSPACE-complete language with function-restricted interactive proofs where the prover only answers questions of the same length as the input.

Lemma 12. *There is a* PSPACE*-complete language L and a probabilistic polynomial-time oracle Turing machine M such that for any input x:*

1. *M only asks its oracle queries of length $|x|$.*
2. *If M is given L as oracle and $x \in L$, then M accepts with probability 1.*
3. *If $x \notin L$, then irrespective of the oracle given to M, M rejects with probability at least 1/2.*

If the restriction on length of oracle queries is removed, the above result holds for *any* PSPACE-complete language - this was used to show the Karp-Lipton style result that if $\mathsf{PSPACE} \in \mathsf{SIZE}(poly)$, then $\mathsf{PSPACE} = \mathsf{MA}$ [LFKN92, Sha92]. We sketch the proof. Let L be a PSPACE-complete language with polynomial-size circuits, and M a probabilistic polynomial-time oracle machine implementing the function-restricted interactive protocol for L. The Merlin-Arthur protocol for L on input x proceeds as follows. Merlin sends Arthur polynomial-size circuits corresponding to all possible lengths of oracle queries of M on input x. Since M is a polynomial-time machine, it can only query polynomially many input lengths, and this is in sum a polynomial amount of information. Arthur simulates M, using the circuits to answer the oracle queries of M. Since by assumption, L has polynomial-size circuits, Merlin can get Arthur to accept with probability 1 on an input $x \in L$ by sending the correct circuits for L on all the required input lengths. Conversely, if $x \notin L$, Merlin *commits* to a specific oracle by sending circuits to Arthur, and hence Arthur rejects with high probability.

In the proof of Theorem 1, we need a smooth parametrization of the above argument. Namely, we need to show that if the PSPACE-complete language L has circuits of size s, then L has a Merlin-Arthur protocol which runs in time $poly(s)$. The same argument as before comes close to achieving that, except that on inputs of length n, the Merlin-Arthur protocol takes time $poly(s(poly(n)))$ rather than time $poly(s(n))$. This is not good enough for our purposes since we cannot make any a priori assumptions on the behavior of the function s. In particular, we cannot assume that $s(poly(n)) = O(poly(s(n)))$. By using Lemma 12, where there is a constraint on the length of the oracle queries, we get around this problem, and running the same argument as before gives us what we need.

Now we choose $s(n)$ to be the minimum circuit size of the language L. For this choice of s, we have that $L \in \mathsf{SIZE}(s)$ but $L \notin \mathsf{SIZE}(s-1)$. By the argument sketched in the previous paragraph, we have that $L \in \mathsf{MATIME}(poly(s))$ but $L \notin \mathsf{SIZE}(s-1)$. If s were, say, n^{k+1}, then this would already

give us Theorem 1. However, we have no control over the behavior of the function s. There are two kinds of problems that arise. These problems are analogous to those that arise when trying to show hierarchy theorems for BPTIME, and are dealt with in much the same way [Bar02, FS04, GST04]. If s is small, say s is linear, then we do not obtain anything non-trivial about the hardness of the language L. However, in this case, by using the unparametrized Karp-Lipton argument, we obtain that PSPACE = MA, and since PSPACE doesn't have n^k size circuits, neither does MA. The other problem is when s is super-polynomial. In this case, we use a translation argument to scale the separation $\mathsf{MATIME}(poly(s)) \not\subseteq \mathsf{SIZE}(s-1)$ downward. The translation argument is not completely uniform because of our lack of information about the function s, however we do the translation advice-efficiently and derive Theorem 1.

We now give the formal proof.

Proof of Theorem 1.

Let L be a PSPACE-complete language as in the statement of Lemma 12. We argue two different ways, based on whether $L \in \mathsf{SIZE}(poly)$ or not.

If $L \in \mathsf{SIZE}(poly)$, then since L is PSPACE-complete, $\mathsf{PSPACE} \subseteq \mathsf{SIZE}(poly)$, and hence $\mathsf{PSPACE} = MA$ [LFKN92, Sha92]. Since, by a simple diagonalization argument, PSPACE doesn't have circuits of size n^k, we get in this case that $\mathsf{MA} \not\subseteq \mathsf{SIZE}(n^k)$.

Now assume $L \notin \mathsf{SIZE}(poly)$. We define the following padded version of L:

$$L' = \{x1^y | x \in L, y \geqslant |x| > 0, y \text{ is a power of 2, the minimum circuit size of } L|_{|x|} \text{ is between } (y+|x|)^{k+1} \text{ and } (2y+|x|)^{k+1}\}$$

We show that $L' \in \mathsf{MA}/1$ but $L' \notin \mathsf{SIZE}(n^k)$.

First we argue the upper bound. We define a machine M' with one bit of advice such that when the advice is set to the right value, M' operates in Merlin-Arthur polynomial time and decides correctly whether its input belongs to L'.

Let x' be an input of length m. The advice bit for length m will be set to 1 if m is of the form $y+n$, where $y \geqslant n$ is a power of 2, and the minimum circuit size of $L|_n$ is between $(y+n)^{k+1}$ and $(2y+n)^{k+1}$.Otherwise the advice bit is set to 0. Note that if m is of the above form, then y and n are determined uniquely, and hence the required property of the minimum circuit size depends solely on m.

The machine operates as follows: if the advice bit is set to 0, it immediately rejects. If the advice bit is set to 1, then it first checks if x' is of the form $x1^y$, where $|x| = n$. If not, it rejects. If yes, in the Merlin (non-deterministic) phase, it guesses a size s between $(n+y)^{k+1}$ and $(n+2y)^{k+1}$ and a circuit C of size s with n input bits. Then, in the Arthur (probabilistic) phase, it simulates the oracle machine M from Lemma 12, using the circuit to answer the oracle queries. Since the circuit is of polynomial size in m, the simulation takes only polynomial time.

If $x' \in L'$, then $x \in L$ and hence when M' guesses the circuit for $L|_n$ correctly, it accepts x' with probability 1. Conversely, if $x' \notin L'$, then either x' does not satisfy the conditions on n and y, in which case the advice bit is 0 and M' rejects, or x' is not of the form $x1^y$ for some $|x| = n$, in which case also M' rejects, or $x \notin L$. If $x \notin L$, then by guessing a circuit in the non-deterministic phase, M' commits to an oracle for the simulation of M, and by Lemma 12, M' rejects with probability at least $1/2$. Thus M' accepts if and only if $x' \in L'$, which proves the upper bound on complexity of L'.

Next we argue the lower bound. Assume, for the sake of contradiction, that $L' \in \mathsf{SIZE}(m^k)$. Let $C_m, m = 1 \ldots \infty$, be a series of circuits for L' such that C_m decides L' correctly at length m and $|C_m| \leqslant m^k$.

Let $s(n)$ be the mininum circuit size of L_n. Since $L \notin \mathsf{SIZE}(poly)$, there is an infinite sequence of input lengths I such that for any input length $n \in I$, $s(n) > (n+1)^{k+1}$. Now consider the following sequence of circuits $C'_n, n \in I$ for deciding L on input lengths in I. Given an $n \in I$, the unique value y such that y is a power of 2 and $(n+y)^{k+1} \leqslant s(n) < (n+2y)^{k+1}$ is hardcoded into C'_n. Since $2^n > s(n) > (n+1)^{2k}$, such a value y exists. Given input x of length n, C'_n pads x with y 1's and simulates C_{n+y} on $x1^y$. It follows from the definition of L' that C'_n decides correctly if $x \in L$. The size of C'_n is at most the size of C_{n+y}, which is at most $(n+y)^k < s(n)$. Thus $|C'_n| < s(n)$, which is a contradiction to the definition of $s(n)$ as the minimum circuit size of L_n. □

Theorem 1 implies circuit lower bounds for the promise version of MA. Intuitively, an MA machine with small advice induces a promise problem when we consider the advice as part of the input. If the language decided by the advice-taking machine is hard, so is the promise problem induced by the machine.

Lemma 13. *If* $\mathsf{MA}/O(n) \not\subseteq \mathsf{SIZE}(n^k)$*, then* $\mathsf{Promise}-\mathsf{MA} \not\subseteq \mathsf{SIZE}(O(n^k))$.

Proof. Let L be a language in $\mathsf{MA}/O(n)$ such that $L \notin \mathsf{SIZE}(n^k)$. M be an MA machine taking cn bits of advice on inputs of length n, for some constant $c > 0$, and $\{b_i\}, i = 1 \ldots \infty, |b_i| = ci$ be a sequence of advice strings such that M with advice b_n decides $L|_n$ correctly.

We use M to define a promise problem $X = (\Pi_{YES}, \Pi_{NO})$. The promise is considered not to hold on input lengths not of the form $(c+1)n$ for some integer n, i.e., inputs of such lengths are not in $\Pi_{YES} \cup \Pi_{NO}$. Any input y of length $(c+1)n$ for some n is broken up as xb, where $|x| = n$ and $|b| = cn$. $y \in \Pi_{YES}$ if M accepts on x with advice b; $y \in \Pi_{NO}$ if M rejects on x with advice b. If M doesn't satisfy the MA promise on x with advice b, then $y \notin \Pi_{YES} \cup \Pi_{NO}$.

We show that X doesn't have circuits of size $m^k/2(c+1)^k$ on inputs of length m. Assume for the purpose of contradiction that there is a sequence of circuits $\{C'_m\}, m = 1 \ldots \infty, |C'_m| \leqslant m^k/2(c+1)^k$ deciding X on input length m. We construct a sequence of circuits $\{C_n\}, |C_n| \leqslant n^k/2$ deciding $L|_n$. On input x of length n, C_n pads x with the correct advice b_n and then simulates $C'_{(c+1)n}$ on input xb_n. The key point is that all inputs of the form $xb_n, |x| = n$ are in $\Pi_{YES} \cup \Pi_{NO}$ - if $x \in L$, the input is in Π_{YES}, otherwise it is in Π_{NO}. Since the circuit $C'_{(c+1)n}$ correctly decides inputs in $\Pi_{YES} \cup \Pi_{NO}$, the circuit C_n correctly decides if its input $x \in L$. The size of C_n is the size of $C'_{(c+1)n}$, which is at most $n^k/2$. Thus we derive that L has circuits of size $n^k/2$, a contradiction. □

Combining Lemma 13 with Theorem 1 we obtain:

Theorem 14. *For each* $k > 0$, $\mathsf{Promise}-\mathsf{MA} \not\subseteq \mathsf{SIZE}(n^k)$.

Since $\mathsf{Promise}-\mathsf{MA} \subseteq \Sigma_2$, Theorem 14 strengthens Kannan's classic circuit lower bound [Kan82] for Σ_2. Since $\mathsf{Promise}-\mathsf{MA} \subseteq \mathsf{PP}$ [Ver92], Theorem 14 strengthens Vinodchandran's recent circuit lower bound [Vin05] for PP. Theorem 14 does not strictly strengthen the best known circuit lower bound in a uniform complexity class - Cai's lower bound [Cai01] for $\mathsf{S_2P}$ because $\mathsf{S_2P}$ is not a syntactic class. However, since $\mathsf{Promise}-\mathsf{MA} \subseteq \mathsf{Promise}-\mathsf{S_2P}$, Theorem 14 does imply that $\mathsf{Promise}-\mathsf{S_2P}$ doesn't have circuits of size n^k for any constant k.

In addition, Theorem 13 implies circuit lower bounds for several promise classes which were not known to require superlinear circuit size. Babai showed that $\mathsf{MA} \subseteq \mathsf{AM}$ and indeed his technique shows that $\mathsf{Promise}-\mathsf{MA} \subseteq \mathsf{Promise}-\mathsf{AM}$, hence Theorem 14 shows that $\mathsf{Promise}-\mathsf{AM}$ requires circuits of size n^k, for each $k > 0$. It is also known that $\mathsf{Promise}-\mathsf{MA} \subseteq \mathsf{Promise}-\mathsf{ZPP}_{\|}^{\mathsf{NP}}$

[NW94, AK97, GZ97] and that $\mathsf{Promise}-\mathsf{MA} \subseteq \mathsf{BPP}_{path}$ [HHT97]. Thus Theorem 14 also yields circuit lower bounds for $\mathsf{Promise}-\mathsf{ZPP}_{\|}^{\mathsf{NP}}$ and $\mathsf{Promise}-\mathsf{BPP}_{path}$. In fact, for each of these classes, the natural analogue of Theorem 1 holds, i.e., the language for which we show a circuit lower bound is decidable with a single bit of advice.

4 Extensions

In this section, we give several extensions of Theorem 1. We show that circuit lower bounds can in fact be obtained for explicit languages in $(\mathsf{MA} \cap \mathsf{coMA})/1$, and that the lower bounds can be made to hold on average rather than just in the worst case.

To prove these results, we need the following strengthened version of Lemma 12 which follows from work of Trevisan-Vadhan and Fortnow-Santhanam [TV02, FS04], together with a couple of tricks.

Lemma 15. *There is a paddable and piecewise random self-reducible* PSPACE*-complete language L and probabilistic oracle machines M and M' such that on input x of length n:*

1. *M and M' only ask questions to their oracle of length n.*
2. *If M (resp. M') is given L as oracle and $x \in L$ (resp. $x \notin L$), M (resp. M') accepts with probability 1.*
3. *If $x \notin L$ (resp. $x \in L$), then irrespective of the oracle, M (resp. M') rejects with probability at least $1/2$.*

Lemma 15 is stronger than Lemma 12 in three respects. The PSPACE-complete language L is required to be downward self-reducible and random self-reducible. Also, both L and $\bar{L}$ are required to have function-restricted interactive proofs (using the terminology of Blum and Kannan [BK95]) where the prover is only asked questions of the same length as the input, whereas this property was only required to hold for L in Lemma 12.

Lemma 15 almost immediately implies the following strengthening of Theorem 1:

Theorem 16. *For each $k > 0$, $(\mathsf{MA} \cap \mathsf{coMA})/1 \not\subseteq \mathsf{SIZE}(n^k)$.*

Proof Sketch.

The proof proceeds along the same lines as the proof of Theorem 1. Consider the language L in the statement of Lemma 15. If $L \in \mathsf{SIZE}(poly)$, then $\mathsf{PSPACE} \subseteq \mathsf{SIZE}(poly)$ and hence $\mathsf{PSPACE} = \mathsf{MA} \cap \mathsf{coMA}$ (since PSPACE is closed under complement). In this case, the theorem follows by direct diagonalization.

If $L \notin \mathsf{SIZE}(poly)$, we consider the padded language L' as in the proof of Theorem 1. The proof that $L' \notin \mathsf{SIZE}(n^k)$ is the same as before. We must also show that $L' \in (\mathsf{MA} \cap \mathsf{coMA})/1$. We show that there is an advice-taking machine M_1 solving L' in $\mathsf{MA}/1$ and an advice-taking machine M_2 solving $\bar{L'}$ in $\mathsf{MA}/1$, where M_1 and M_2 use the *same* bit of advice. As before, the advice bit is used to code information whether the input length satisfies a certain property related to the minimum circuit size. Since the minimum circuit size of L' is the same as the minimum circuit size for $\bar{L'}$, both machines M_1 and M_2 can use the same advice bit.

The definition of M_1 is the same as before. M_2 acts in an analogous way except that it accepts the input if the advice bit is set to 0 or the input is not of the correctly padded form, and it uses machine M' from Lemma 15 to execute the Merlin-Arthur protocol rather than machine M. □

We use Lemma 15 to strengthen Theorem 1 in various other directions below.

4.1 Constructivity

When proving a circuit lower bound for a complexity class, we would like to identify an explicit language in the complexity class for which the lower bound holds. Apart from the philosophical satisfaction that constructivizing the lower bound gives us, we often gain deeper insight into the proof which may help in proving a stronger lower bound. For instance, if we could find an explicit language in MA without linear-size circuits, we could try to prove circuit lower bounds for NP by derandomizing the MA algorithm for the language, which may be much easier than showing $\mathsf{MA} = \mathsf{NP}$.

Kannan's original proof [Kan82] that Σ_2 doesn't have circuits of size n^k for any k was non-constructive. Recently, Cai and Watanabe [CW04] found a way to constructivize it. Other lower bounds such as Cai's lower bound for $\mathsf{S_2P}$ [Cai01] and Vinodchandran's lower bound for PP [Vin05] are yet to be constructivized.

We show that the proof of Theorem 1 can be constructivized.

It may not be immediately apparent what it means for a language solvable with advice to be explicit. We say a language solvable with advice is explicit if we can demonstrate an advice-taking machine accepting the language. This is a general definition which applies to any semantic class with advice. The definition is natural in the sense that proving an explicit circuit lower bound for a semantic class with small advice implies an explicit lower bound for any uniform syntactic class containing the semantic class. Thus, our constructivization implies the result of Cai and Watanabe [CW04] through a different proof, as well as the new result that for any k there is an explicit language in PP without circuits of size n^k.

Our proof proceeds by defining a single machine to handle both cases in the proof of Theorem 1.

Theorem 17. *For each $k > 0$, there is an explicit language L_k in $\mathsf{MA}/1$ such that $L_k \notin \mathsf{SIZE}(n^k)$.*

Proof. We proceed along the lines of the proof of Theorem 1, but handle the two cases together. Let L be a PSPACE-complete language as in Lemma 15. Let L' be a language in PSPACE which doesn't have circuits of size n^k almost everywhere. Such a language can be constructed by direct diagonalization. Since L is PSPACE-complete, there is a polynomial-time reduction from L' to L; since L is paddable, we can assume that there is a polynomial $p(n) > n$ such that all queries are of length $p(|x|)$ on input x. We define the following language L_k:

$$L_k = \{x | x \in L', |x| \text{ is odd }, L|_n \text{ has circuits of size at most } (6n)^{k+1} \text{ for all } n \geqslant |x|\} \cup$$

$$\{xx1^y | x \in L, y \geqslant 2|x| > 0, y \text{ is a power of 2, the minimum circuit size of } L|_{|x|} \text{ is between } (y+2|x|)^{k+1}$$

$$\text{and } (2y + 2|x|)^{k+1}\}$$

We need to show that $L_k \notin \mathsf{SIZE}(m^k)$, and $L_k \in \mathsf{MA}/1$. We show the lower bound first. Either there is an $n_0 \geqslant 0$ such that $L|_n$ has circuits of size at most $(6n)^{k+1}$ for all $n \geqslant n_0$, or not. In the first case, let n_1 be such that L'_n does not have circuits of size n^k for any $n \geqslant n_1$. Then, for all odd $m \geqslant max(n_0, n_1)$, L_k coincides with L' on inputs of length m, and hence L_k doesn't have circuits of size m^k, by the construction of L'. In the second case, there is an infinite set I of input lengths n such that $L|_n$ doesn't have circuits of size $(6n)^{k+1}$. This implies that for any input length $n \in I$, there is a y_n which is a power of 2 such that the minimum circuit size of $L|_n$ is between $(2n + y_n)^{k+1}$ and $(2n + 2y_n)^{k+1}$. If L_k had circuits of size m^k, we could decide $x \in L|_n$ by a circuit

which runs the circuit for L_k on $xx1^{y_n}$. The size of such a circuit would be at most $(2n + y_n)^k$, which is a contradiction to the assumption on y_n.

Now we show the upper bound. We define a machine M_k taking one bit of advice that decides L_k. We discuss odd input lengths and even input lengths separately. If the input length m is odd, the advice bit is 1 iff $L|_n$ has circuits of size at most $(6n)^{k+1}$ for all $n \geqslant m$. If the advice bit is 0 for an odd length m, M_k immediately rejects. If the advice bit is 1, it first implements the reduction from L' to L on its input x, generating an instance $f(x)$ such that $|f(x)| = p(|x|)$ and $f(x) \in L$ iff $x \in L'$. Since the advice bit is 1 and $p(m) > m$, we are assured that $L|_{p(m)}$ has circuits of size at most $(6p(m))^{k+1}$. M_k now executes a Merlin-Arthur protocol at this input length, guessing a circuit of size at most $(6p(m))^{k+1}$ and running the probabilistic oracle machine M from Lemma 15. It uses the guessed circuit to answer any oracle queries made by M. If M accepts, it accepts, otherwise it rejects. Using Theorem 15, we see that M_k accepts iff $f(x) \in L$ iff $x \in L'$ iff $x \in L_k$ (since the advice bit is 1 at this length).

If the input length m is even, the proof of the upper bound is along the same lines as the second case in the proof of Theorem 1. The advice bit is set to 1 precisely when the input length $m = 2n + y$ for $y \geqslant 2n$ a power of 2, and the minimum circuit size of $L|_n$ is between $(y + 2n)^{k+1}$ and $(2y + 2n)^{k+1}$. If the advice bit is set to 0, M_k rejects. If the advice bit is set to 1, M_k checks if its input $x' = xx1^y$ for some string x and integer y, where y is a power of 2 and $|y| \geqslant 2|x|$. If not, it rejects. If yes, it runs a Merlin-Arthur protocol for checking if $x \in L$, by guessing a circuit with input length $|x|$ of size between $(y + 2n)^{k+1}$ and $(2y + 2n)^{k+1}$, and then running the probabilistic oracle machine M from Lemma 15 on input x with oracle queries answered by simulating the circuit. This Merlin-Arthur protocol accepts iff $x \in L$, and runs in polynomial time in the input length m.

□

Theorem 17 can be strengthened to show that for each k there is an explicit language in $(\mathsf{MA} \cap \mathsf{coMA})/1$ without circuits of size n^k, just by combining the ideas in the proofs of Theorem 16 and Theorem 17. Together with Lemma 13, this yields Theorem 2.

4.2 Average-Case Hardness

Theorem 1 shows that $\mathsf{MA}/1$ does not have fixed polynomial size circuits in the worst case. It is natural to ask if we can prove a stronger result showing hardness on the average. Such results are useful, for instance, in the theory of cryptography. For "high" complexity classes such as PSPACE and EXP, it is known that hardness on average is equivalent to hardness in the worst case. This is not known for classes in the polynomial hierarchy; moreover, there has been a lot of work recently [FF93, BT03, Vio05] showing that natural attempts to show worst-case to average-case reductions within the polynomial hierarchy are doomed to fail.

Nevertheless, we do manage to extend Theorem 1 to an average-case hardness result. We accomplish this by taking advantage of the worst-case to average-case connection for the PSPACE-complete language L in Lemma 15. Intuitively, if L has polynomial-size circuits, then $\mathsf{PSPACE} = \mathsf{MA}$ and since PSPACE is average-case hard with respect to circuits of fixed polynomial size, so is MA. If L does not have polynomial-size circuits, then using the worst-case to average-case reduction for L, L is solvable in size $poly(s)$ but is average-case hard for size s, where s is some superpolynomial function. Translating this separation downward as in the proof of Theorem 1 preserves the property that hardness holds on average. However, this only yields a mild average-case hardness result - in order to get a stronger one, we amplify hardness further using standard techniques.

We need the following lemma which follows by applying a standard hardness amplification [STV01] to a hard language in PSPACE obtained using direct diagonalization.

Lemma 18. *For each constant k,* $\mathsf{PSPACE} \not\subseteq heur_{1/2+1/n^k} - \mathsf{SIZE}(n^k)$.

Now we strengthen Theorem 1 to show that $(\mathsf{MA} \cap \mathsf{coMA})/1$ is mildly average-case hard for circuits of size n^k.

Theorem 19. *There is a constant $a > 0$ such that for each constant k,* $(\mathsf{MA} \cap \mathsf{coMA})/1 \not\subseteq heur_{1-1/n^a} - \mathsf{SIZE}(n^k)$.

Proof. The proof is a modification of the proof of Theorem 16. Consider the language L in the statement of Lemma 15. Let f be a piecewise random self-reduction for L making at most n^b queries for some constant b, and let c be a constant such that the density of each piece of f is at least $1/n^c$.

If $L \in \mathsf{SIZE}(poly)$, then $\mathsf{PSPACE} = \mathsf{MA} \cap \mathsf{coMA}$, and in this case the theorem follows from Lemma 18.

If $L \notin \mathsf{SIZE}(poly)$, consider the following padded language L':

$$L' = \{xz | x \in L, |z| \geqslant |x|, |z| \text{ is a power of } 2, L|_{|x|} \text{ has minimum circuit size between } (|x| + |z|)^{k+1} \text{ and } (|x| + 2|z|)^{k+1}\}$$

The proof that $L' \in (\mathsf{MA} \cap \mathsf{coMA})/1$ is the same as in the proof of Theorem 16, except that we allow any pad of an appropriate length in the definition of L' rather than restricting it to be all 1's.

Next we show the average-case lower bound. Assume, for the purpose of contradiction, that $L' \in heur_{1-1/m^a} - \mathsf{SIZE}(m^k)$, where a is a constant to be fixed later. For each m, let D_m be a circuit correctly deciding at least a $1 - 1/m^a$ fraction of inputs of L' of length m.

Since $L \notin \mathsf{SIZE}(poly)$, there is an infinite set I of input lengths such that for each $n \in I$, the minimum circuit size of $L|_n$ is at least $(3n)^{k+1}$. This implies that for each $n \in I$, there is a unique integer y_n such that y_n is a power of 2, $y_n \geqslant n$, and the minimum circuit size of $L|_n$ is between $(n + y_n)^{k+1}$ and $(n + 2y_n)^{k+1}$.

We construct circuits of size less than $(n + y_n)^{k+1}$ for solving $L|_n$ on certain $n \in I$ and thus derive a constradiction. First we construct randomized circuits and then convert them into deterministic circuits. On input x of length n, the randomized circuit C'_n does the following. It applies the piecewise random self-reduction f to x to obtain queries $q_1, q_2 \ldots q_l, l \leqslant n^b$ of length n. It then generates random strings $r_1, r_2 \ldots r_l$ each of length y_n and simulates D_{n+y_n} on each of $q_1 r_1, q_2 r_2 \ldots q_l r_l$ in turn to obtain answers $a_1, a_2 \ldots a_l$. It then computes $f(x, a_1, a_2 \ldots a_l)$ and outputs the answer.

We analyze the success probability and the size of C'_n. Since the density of each piece of the piecewise random self-reduction is at least $1/n^c$, the probability that there is an $a_i, 1 \leqslant i \leqslant l$ such that $a_i \neq L(q_i)$ is atmost $n^b n^c/(n + y_n)^a$. This is also an upper bound on the probability that C' makes a mistake on x, by the definition of "piecewise random self-reduction". Choosing $a = b + c + 1$, the probability that C'_n is wrong is at most $1/n$. Also, the size of C'_n is at most $poly(n) + l(n + y_n)^k = O(poly(n)(n + y_n)^k)$.

Now consider the randomized circuit C''_n that runs C'_n n times independently and outputs the majority answer. By Chernoff bounds, the probability that C''_n makes an error is less than 2^{-n}. Hence there is some setting of the random bits of C''_n to obtain a deterministic circuit C_n which decides $L|_n$ correctly. The size of C_n is atmost $n^d(n + y_n)^k$ for some constant d and large enough n.

Since $L \notin \mathsf{SIZE}(poly)$, there is an infinite subset I' of I such that for each $n \in I'$, $y_n > n^d$. This implies that for large enough $n \in I'$, the size of C_n is less than $(n+y_n)^{k+1}$, which is a contradiction to the definition of y_n.

□

Next we amplify the hardness of the average-case hard language in Theorem 19. Any language is solvable on at least 1/2 the inputs of any length by constant-sized circuits, since we can define the circuit on input length n to just output 1 if at least half of the inputs of that length are in the language, and 0 otherwise. However, we can hope to obtain hardness close to 1/2. The tool we use for this purpose is the celebrated Yao XOR Lemma [Yao82], which states that if a Boolean function is mildly hard, then the parity of the function values on several independent inputs is hard to predict with probability significantly more than 1/2.

Lemma 20. *Let $f^{\oplus t}$ denote the Boolean function which on input of the form $x_1x_2\dots x_t$, $|x_1| = |x_2| \dots = |x_t| = n$, outputs the parity of $f(x_1), f(x_2)\dots f(x_t)$, and outputs 0 on all other inputs. For any constants $a, k > 0$, there exists constants k' and l such that if $f \notin heur_{1-1/n^a} - \mathsf{SIZE}(n^{k'})$, then $f^{\oplus n^l} \notin heur_{1/2+1/m^k} - \mathsf{SIZE}(m^k)$, where $m = n^{l+1}$ is the input size of $f^{\oplus n^l}$.*

We apply the lemma to the hard Boolean function in Theorem 19. The key observation is that if the original function is in $(\mathsf{MA} \cap \mathsf{coMA})/1$, so is the new function.

Proof of Theorem 3. To derive the hardness, we apply Lemma 20 to the hard Boolean function in Theorem 19. Let a be the constant in the statement of Theorem 19. Let k' and l be the corresponding constants from Lemma 20. Theorem 19 gives us a Boolean function $f \notin heur_{1-1/n^a} - \mathsf{SIZE}(n^{k'})$. By Lemma 20, $f^{\oplus n^l} \notin heur_{1/2+1/m^k} - \mathsf{SIZE}(m^k)$, where m is the input length of $f^{\oplus n^l}$.

This gives us the required lower bound, now we need to show that $f^{\oplus n^l} \in (\mathsf{MA} \cap \mathsf{coMA})/1$ if $f \in (\mathsf{MA} \cap \mathsf{coMA})/1$. Let M_1 and M_1' be MA machines deciding f and $\neg f$ respectively with a shared bit of advice. We define MA machines M_2 and M_2' deciding $f^{\oplus n^l}$ and $\neg f^{\oplus n^l}$ respectively with a shared bit of advice. For an input length m which is not of the form n^{l+1} for some integer n, M_2 and M_2' ignore their advice - M_2 rejects and M_2' accepts.

Now consider an input length $m = n^{l+1}$ for some integer n. The advice bit for M_2 and M_2' at length m is the same as the advice bit for M_1 and M_1' at length n. An input x of length m is interpreted as $x_1x_2\dots x_t$, where $t = n^l$ and each $x_i, 1 \leqslant i \leqslant t$ is of length n. In its non-deterministic phase, M_2 guesses a partition of $\{x_1, x_2 \dots x_t\}$ into two sets S and S' such that $|S|$ is odd. It also guesses, for each input $y \in S$, a proof by M_1 that $f(y) = 1$, and for each input $y \in S'$, a proof by M_1' that $f(y) = 0$. It then executes the probabilistic phase of M_1 to confirm the proofs of the strings in S, and the probabilistic phase of M_1' to confirm the proofs of the strings in S'; it only accepts if all these probabilistic computations accept. If $f^{\oplus t}(x) = 1$, then there is some correct partition into an odd-sized set S and a set S' and correct proofs for each of the strings in these sets, hence the probabilistic phase of M_2 will accept with probability 1. On the other hand, if $f^{\oplus t}(x) = 0$, there is no correct partition, and hence atleast one of the proofs must be wrong, which implies M_2 will accept with probability at most 1/2. Note that since t is polynomially bounded, M_2 runs in polynomial time.

The construction of M_2' is exactly analogous, except that it guesses a partition where the size of the set S is even. The proof that M_2' decides $\neg f^{\oplus t}$ follows along the same lines.

Thus we have MA machines M_2 and M_2', sharing a bit of advice, which decide $f^{\oplus t}$ and $\neg f^{\oplus t}$ respectively. This shows that $f^{\oplus t} \in (\mathsf{MA} \cap \mathsf{coMA})/1$. □

The proof technique of Theorem 19 can be used to show average-case circuit lower bounds for Σ_2 (without advice). First, we observe that since $\mathsf{MA}/1 \subseteq \Sigma_2/1$, Theorem 19 implies an average-case circuit lower bound for $\Sigma_2/1$. Now, we can define a language in Σ_2 that is mildly hard using the idea in the proof of Theorem 14 of appending the advice to the input. Because Σ_2 is a syntactic class, implementing this idea yields a language rather than a promise problem.

Lemma 21. *There is a constant c such that for each k, $\Sigma_2 \not\subseteq heur_{1-1/n^c} - \mathsf{SIZE}(n^k)$.*

Proof. Fix k. It follows from Theorem 19 that there is a constant a and a language $L \in \Sigma_2/1$ such that $L \notin heur_{1-1/(n+1)^a} - \mathsf{SIZE}((n+1)^k)$. Let M be the advice-taking Σ_2 machine deciding L.

We define the following language L': $xb \in L'$ if and only if M accepts x when given advice bit b. We claim that $L' \notin heur_{1-1/2n^a} - \mathsf{SIZE}(n^k)$. Suppose, to the contrary, that there is a family of circuits $\{C'_n\}$ such for each n, C'_n is of size at most n^k and decides $L'|_n$ correctly on at least $1-1/2n^a$ fraction of inputs.We define circuits $\{C_n\}$ for L as follows. For each n, C_n has the correct advice bit b_n for M on inputs of length n hard-coded into it.On input x, it runs C'_{n+1} on input xb_n, accepting if C'_{n+1} accepts and rejecting if C'_{n+1} rejects. Since by assumption, C'_{n+1} decides at least $1 - 1/2(n+1)^a$ fraction of all inputs of length $n+1$ correctly, it must decide at least a $1 - 1/(n+1)^a$ fraction of inputs of the form xb_n correctly. Hence C_n succeeds on at least a $1 - 1/(n+1)^a$ fraction of inputs of length n and has size at most $(n+1)^k$, which contradicts the assumption on the hardness of L.

By setting $c = a+1$, we have that $L' \notin heur_{1-1/n^c} - \mathsf{SIZE}(n^k)$, since $n^{a+1} \geqslant 2n^a$ for all $n > 1$. Also $L' \in \Sigma_2$ by construction, thus proving the lemma. □

We amplify the hardness of the language L' in the proof of Lemma 21 using monotone hardness amplifiers as in the work of O'Donnell [O'D04]. O'Donnell focusses on hardness amplification within NP but his results relativize and yield hardness amplification within Σ_2.

Lemma 22. *[O'D04] For any constants $k > 0$ and $a > 0$ there is a constant k' such that if $\Sigma_2 \not\subseteq heur_{1-1/n^a} - \mathsf{SIZE}(n^{k'})$, then $\Sigma_2 \not\subseteq heur_{1/2+1/\sqrt{n}} - \mathsf{SIZE}(n^k)$.*

Combining Lemma 21 and Lemma 22, we derive our main average-case hardness result for Σ_2.

Theorem 23. *For every k, $\Sigma_2 \not\subseteq heur_{1/2+1/\sqrt{n}} - \mathsf{SIZE}(n^k)$.*

5 An Arithmetic Circuit Lower Bound

In this section, we prove arithmetic circuit lower bounds for MA, *without* advice.

We argue based on the non-uniform complexity of PER, rather than the language L in Lemma 15. If PER over $\mathbb{Z}$ has polynomial-sized arithmetic circuits, then it also has polynomial-size Boolean circuits. In this case, we can show that $\mathsf{P}^{\mathsf{PP}} = \mathsf{MA}$, and it follows from results of Toda [Tod89] and Kannan [Kan82] that MA doesn't have Boolean circuits of size n^k for any k.

If PER over $\mathbb{Z}$ doesn't have polynomial-size arithmetic circuits, then we use a translation argument as in the proof of Theorem 1. The key point is that we do not need a bit of advice to tell us whether the pad length is sufficient, because we can use Lemma 9 to keep the MA promise even when the pad length is insufficient. Instead of using a Karp-Lipton style argument for showing the padded function is computable in MA, we guess an arithmetic circuit of the appropriate size for PER and verify that it works. If the verification succeeds, we compute PER using the circuit, otherwise we compute the zero polynomial.

Proof of Theorem 4. Fix an arbitrary k. We consider two cases. If PER over $\mathbb{Z}$ has polynomial-size arithmetic circuits, then 0-1-PER over $\mathbb{Z}$ has polynomial-size Boolean circuits [1]. Using Theorem 8, we get that $\sharp P \subseteq SIZE(poly)$, and hence $\mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\sharp P} \subseteq \mathsf{SIZE}(poly)$. By a Karp-Lipton style argument using function-restricted interactive proofs for P^{PP} [LFKN92], we get that $\mathsf{P}^{\mathsf{PP}} \subseteq \mathsf{MA}$. Toda [Tod89] showed that $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PP}}$, hence we get that $\mathsf{PH} \subseteq \mathsf{MA}$. Kannan [Kan82] exhibited a language in PH without circuits of size n^k, thus we derive that $\mathsf{MA} \not\subseteq \mathsf{SIZE}(n^k)$.

In the other case, there are infinitely many values $n = t^2$ such that PER on n inputs does not have arithmetic circuits of size $4^k n^{2k}$ over $\mathbb{Z}$. For each $n = t^2$, let $s(n)$ be the size of the smallest arithmetic circuit for PER on n inputs over $\mathbb{Z}$. By Ryser's formula, we know $s(n) \leqslant n2^{\sqrt{n}}$.

We define a sequence of polynomials $p = \{p_m\}$ such that $Gh(p) \in \mathsf{MA}$ and p does not have arithmetic circuits of size m^k over $\mathbb{Z}$. For each m, decompose $m = m_1 + m_2$, where $1 \leqslant m_1 \leqslant m_2$ and m_2 is a power of 2. Such a decomposition exists for each m and is unique. If $m_1 \neq t^2$ for some t, then we set p_m to be the zero polynomial. If $m_1 = t^2$ for some t, we define p_m depending on whether $(m_1 + m_2) \geqslant s(m_1)^{1/2k}$. If yes, we set p_m to be Permanent defined on the first m_1 of its m inputs, otherwise we set p_m to be the zero polynomial. Clearly p is of feasible degree.

First we show that $Gh(p) \in \mathsf{MA}$ and then that p does not have arithmetic circuits of size m^k. We define a Merlin-Arthur machine M running in polynomial time which accepts $Gh(p)$. Given an input $\langle \vec{x}, v \rangle$, M sets $m = |\vec{x}|$, and then decomposes $m = m_1 + m_2$ as described earlier. If m_1 is not a perfect square, M accepts if $v = 0$ and rejects otherwise. if m_1 is a perfect square, M uses its non-deterministic phase to guess an arithmetic circuit C of size m^{2k}. In its probabilistic phase, it first simulates the coRP machine M' corresponding to Lemma 9 on $\langle C, 1^t \rangle$ to test if the circuit computes PER over m_1 variables correctly or not. If $m^{2k} \geqslant s(m_1)$, i.e., if $m_1 + m_2 \geqslant s(m_1)^{1/2k}$, then there is some guessed circuit C for which M' accepts with probability 1, otherwise M' rejects with probability close to 1. If M' accepts, M evaluates C on $\vec{x}$ modulo a large enough random prime (so that the evaluation can be done in polynomial time) to obtain a value v'. If M' rejects, M accepts if $v = 0$, and rejects otherwise. In the other case, If $v' = v$, it accepts, otherwise it rejects. The machine M runs in polynomial time. If $m^{2k} \geqslant s(m_1)$, there is some guessed circuit C such that M accepts with probability close to 1 on $\langle \vec{x}, v \rangle$ such that v equals the Permament on first m_1 variables of $\vec{x}$, and rejects with probability close to 1 on all guessed circuits for other v. If $m^{2k} < s(m_1)$, for each guessed circuit, M accepts with probability close to 1 if $v = 0$ and rejects with probability close to 1 otherwise. The probabilistic phase of M has two-sided error, but MA with 2-sided error equals MA with 1-sided error, thus we are done.

Next, we show that p does not have arithmetic circuits of size m^k. Suppose, to the contrary, that p does have arithmetic circuits of size m^k. For each m, let C_m be a circuit of size at most m^k computing p. Then we can solve PER on n variables over $\mathbb{Z}$ with arithmetic circuits of size less than $s(n)$ for infinitely many n, as follows. Let m_2 be the least power of 2 such that $n \leqslant m_2$ and $n + m_2 \geqslant (s(n))^{1/2k}$. Such an m_2 exists and is unique, since $s(n) \leqslant n2^{\sqrt{n}}$. Now add m_2 dummy variables to the input, and simulate C_{n+m_2} on the new input set. By definition of p and assumption on C, C_{n+m_2} computes PER on n variables. The size of C_{n+m_2} is atmost $(2(n + s(n)^{1/2k}))^k < (2n)^k \sqrt{s(n)}$. This is less than $s(n)$ for any value of n for which $s(n) \geqslant 4^k n^{2k}$. By assumption, there are infinitely many such values, hence we have a constradiction to the definition of $s(n)$ as the minimum arithmetic circuit size for each n.

□

[1] We replace the arithmetic gates with the corresponding Boolean circuits. To make sure that intermediate values do not get too large, we do all the computations modulo a large enough random prime, which is representable in a polynomial number of bits and can be coded into the circuit

6 Implications for Derandomization

Much of the work on derandomizing probabilistic classes such as BPP and MA focusses on the construction of pseudo-random generators. Pseudo-random generators suffice for derandomization; a natural question is whether they are also necessary. Impagliazzo, Kabanets and Wigderson [IKW02], and Kabanets and Impagliazzo [KI04] show results of the form that derandomization implies circuit lower bounds, but the circuit lower bounds obtained are not strong enough to yield pseudo-random generators. In this section, we use Theorem 1 to show that derandomizing MA does imply the existence of pseudo-random generators, modulo a constant amount of advice.

Definition 24. *A function $G_n : \{0,1\}^{s(n)} \to \{0,1\}^n$ is a pseudo-random generator (PRG) with seed length s for size n if for each circuit C of size at most n, $|\Pr_{x\in\{0,1\}^n}[C(x) = 1] - \Pr_{y\in\{0,1\}^{s(n)}}[C(G_n(y)) = 1]| < 1/n$.*

A family $G = \{G_n\}, n = 1 \ldots \infty$ of such functions is a PRG with seed length s if for each n, G_n is a PRG with seed length s for size n. G is an i.o.PRG if for infinitely many n, G_n is a PRG for size n.

G is a strong non-deterministic PRG if the function $f_G : (x, i) \to G(x)_i$, where $x \in \{0,1\}^{s(n)}$ and $1 \leqslant i \leqslant n$, can be computed in $\mathsf{NTIME}(2^{O(s(n))}) \cap \mathsf{coNTIME}(2^{O(s(n))})$. G is a strong non-deterministic PRG using advice $a(n)$ if f_G can be computed in $(\mathsf{NTIME}(2^{O(s(n))}) \cap \mathsf{coNTIME}(2^{O(s(n))}))/a(n)$.

Nisan and Wigderson [NW94] showed how to construct PRGs from hard functions.

Theorem 25. *[NW94] If $(\mathsf{NE}\cap\mathsf{coNE})/a(n) \not\subseteq$ i.o.$\mathsf{SIZE}(poly)$ (resp. $(\mathsf{NE}\cap\mathsf{coNE})/a(n) \not\subseteq \mathsf{SIZE}(poly)$), then for each $\epsilon > 0$ there is a strong nondeterministic PRG (resp. a strong nondeterministic i.o.PRG) with seed length n^ϵ using advice $a(s(n))$.*

Strong non-deterministic PRGs can be used to derandomize MA (even with advice).

Proposition 26. *If there is a strong non-deterministic PRG (resp. a strong non-deterministic i.o.PRG) with seed length s using advice $a(n)$, then for any advice length function $b(n)$, $\mathsf{MA}/b(n) \subseteq \mathsf{NTIME}(2^{O(s(n))})/(a(poly(n)) + b(n))$ (resp. $\mathsf{MA}/b(n) \subseteq \mathsf{NTIME}(2^{O(s(n))})/(a(poly(n)) + b(n) + O(\log(n)))$.*

We will show a partial converse to Proposition 26, namely that a non-trivial derandomization of MA/1 implies a strong non-deterministic PRG with non-trivial seed length using 2 bits of advice.

Lemma 27. *If $\mathsf{MA}/1 \subseteq \mathsf{NE}/1$, then $(\mathsf{MA} \cap \mathsf{coMA})/1 \subseteq (\mathsf{NE} \cap \mathsf{coNE})/2$. If $\mathsf{MA}/O(1) \subseteq \mathsf{NE}/O(1)$, then $(\mathsf{MA} \cap \mathsf{coMA})/O(1) \subseteq (\mathsf{NE} \cap \mathsf{coNE})/O(1)$.*

Proof. We show the first implication. The second implication follows along analogous lines.

Assume $\mathsf{MA}/1 \subseteq \mathsf{NE}/1$. By complementing both sides, it follows that $\mathsf{coMA}/1 \subseteq \mathsf{coNE}/1$. Therefore $(\mathsf{MA}\cap\mathsf{coMA})/1 \subseteq \mathsf{MA}/1\cap\mathsf{coMA}/1 \subseteq \mathsf{NE}/1\cap\mathsf{coNE}/1$. Now $\mathsf{NE}/1\cap\mathsf{coNE}/1 \subseteq (\mathsf{NE}\cap\mathsf{coNE})/2$ - the $\mathsf{NE} \cap \mathsf{coNE}$ computation uses its first bit of advice for the non-deterministic part of the computation, and the second bit of advice for the co-nondeterministic part. Recall that in the model of advice we are using, the computation needs to satisfy the promise only when the advice is correct. □

Theorem 28. *If $\mathsf{MA}/1 \subseteq \mathsf{NE}/1$, then for each $\epsilon > 0$, there is a strong nondeterministic i.o.PRG with seed length n^ϵ using 2 bits of advice. If $\mathsf{MA}/O(1) \subseteq \mathsf{NE}/O(1)$, then for each $\epsilon > 0$, there is a strong nondeterministic i.o.PRG with seed length n^ϵ using O(1) bits of advice.*

Proof. Again we show the first implication, and the second follows analogously.

If $\mathsf{MA}/1 \subseteq \mathsf{NE}/1$, then by Lemma 27, $(\mathsf{MA} \cap \mathsf{coMA})/1 \subseteq (\mathsf{NE} \cap \mathsf{coNE})/2$. By Theorem 16, for each k, there is a language in $(\mathsf{NE} \cap \mathsf{coNE})/2$ which doesn't have circuits of size n^k. By Theorem 25, it follows that for each $\epsilon > 0$, there is a strong nondeterministic i.o.PRG with seed length n^ϵ using 2 bits of advice. □

In terms of advice, the previous best known result was by Impagliazzo, Kabanets and Wigderson [IKW02], who showed that derandomization of MA implies the existence of a strong nondeterministic PRG using n^ϵ bits of advice for arbitrarily small $\epsilon > 0$.

There are two directions in which Theorem 28 can be improved. First, we could hope to derive a PRG from the derandomization assumption, rather than just an i.o.PRG. This would imply an *equivalence* between strong non-deterministic PRGs with sub-polynomial seed length using $O(1)$ bits of advice and simulation of $\mathsf{MA}/O(1)$ in $\mathsf{NE}/O(1)$. We would obtain this result if we could prove that for each k, $\mathsf{MA}/1 \not\subseteq i.o.\mathsf{SIZE}(n^k)$, i.e., if we could get a circuit lower bound that works on all input lengths rather than just infinitely many of them.

Second, we could hope to eliminate the advice and derive a PRG from the assumption that $\mathsf{MA} \subseteq \mathsf{NE}$. Such a result using our methodology would involve obtaining circuit lower bounds in MA rather than in $\mathsf{MA}/1$.

Theorem 28 implies a "gap theorem" for $\mathsf{MA}/O(1)$, in that a mild derandomization of this class implies an even stronger derandomization.

Corollary 29. *If* $\mathsf{MA}/O(1) \subseteq \mathsf{NE}/O(1)$*, then* $\mathsf{MA}/O(1) \subseteq i.o.\mathsf{NSUBEXP}/O(1))$.

Proof Sketch. By Theorem 28, if $\mathsf{MA}/O(1) \subseteq \mathsf{NE}/O(1)$, then for each $\epsilon > 0$ there is a strong nondeterministic i.o.PRG with seed length n^ϵ using $O(1)$ bits of advice. Now Proposition 26 gives that $\mathsf{MA}/O(1) \subseteq i.o.\mathsf{NSUBEXP}/O(\log(n))$. By slightly modifying the proof of Theorem 1 and analyzing input lengths on which the padded language is hard, it is possible to get a simulation in $i.o.\mathsf{NSUBEXP}/O(1)$. The details are rather technical, and we omit them here. □

7 Discussion and Further Work

Any successful technique showing circuit lower bounds for a complexity class must evade two classic "obstacles" - the *relativization* obstacle [BGS75] and the *naturalness* obstacle [RR97]. The first obstacle applies to techniques that relativize, i.e., the technique works even when the circuits and the machines defining the complexity class are given access to the same oracle. Wilson [Wil85] constructed an oracle relative to which NP has linear size circuits, hence no successful techique showing circuit lower bounds for NP can relativize. The second obstacle applies to techniques which attempt to show lower bounds by defining a "natural" property of Boolean functions which holds for most functions on a given number of bits, is efficiently verifiable and implies circuit lower bounds for any function satisfying it. Razborov and Rudich [RR97] observed that all known techniques proving circuit lower bounds against restricted class of circuits implicitly define natural properties, and showed that any natural property implying nontrivial circuit lower bounds against general circuits can be used to break cryptographic protocols that are widely believed to be secure. Between them, the relativization obstacle and the naturalness obstacle rule out most of the lower bound techniques that have been developed to date.

Our proof of Theorem 1 evades both obstacles. Aaronson [Aar05] constructed an oracle relative to which PP has linear-size circuits, and hence so does $\mathsf{Promise} - \mathsf{MA}$. Thus there can be no

relativizing proof of Theorem 1. Our proof technique also uses diagonalization in an essential way, and diagonalization is an inherently non-natural proof technique. Also, our lower bounds are proved for certain appropriately translated versions of complete problems for high complexity classes - completeness is a property of uniform Boolean functions which holds only in exceptional cases, in contradiction to what we require of a natural property.

There are earlier examples of lower bound techniques which use a similar combination of ingredients [BFT98, Vin05, Aar05] and evade both obstacles. However, our technique is the first of this kind to establish non-trivial circuit lower bounds for a class that is close to NP. Given the paucity of candidate techniques, it would be worthwhile to investigate if our technique can be extended to prove circuit lower bounds for NP.

In an orthogonal direction, it would be very interesting to investigate further obstacles and limitations for lower bound techniques. The diagonalization and naturalness obstacles often serve to create the impression that circuit lower bounds are fundamentally difficult to prove. However, our proof is not technically complex. This suggests that we must either revise our intuitions about the difficulty of proving circuit lower bounds, or find new evidence to back these intuitions.

In terms of potential technical improvements of Theorem 1, the obvious one is to eliminate the bit of advice and obtain a lower bound for uniform MA. Other natural improvements include deriving a circuit lower bound that works on all input lengths rather than just on infinitely many of them, and showing that MAE doesn't have circuits of size $2^{\epsilon n}$ for some $\epsilon > 0$. Thus far, it is only known that MAE doesn't have circuits of polynomial size [BFT98].

8 Acknowledgments

I am indebted to Valentine Kabanets for his support, as well as for several interesting discussions. Scott Aaronson made several useful comments and suggestions, most importantly that an arithmetic circuit lower bound could be shown using the techniques in the proof of Theorem 1.

References

[Aar05] Scott Aaronson. Oracles are subtle but not malicious. *Electronic Colloquium on Compuational Complexity*, 12(40), 2005.

[Aar06] Scott Aaronson. Personal communication. 2006.

[Ajt83] Miklos Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[AK97] Vikraman Arvind and Johannes Köbler. On pseudorandomness and resource-bounded measure. In *Proceedings of 17th Conference on the Foundations of Software Technology and Theoretical Computer Science*, pages 235–249. Springer, 1997.

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

[BCG+96] Nader Bshouty, Richard Cleve, Ricard Gavalda, Sampath Kannan, and Christino Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52(2):268–286, 1996.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BFT98] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Proceedings of 13th Annual IEEE Conference on Computational Complexity*, pages 8–12, 1998.

[BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[BK95] Manuel Blum and Sampath Kannan. Designing programs that check their work. *Journal of the Association for Computing Machinery*, 42:269–291, 1995.

[BT03] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. In *Proceedings of 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 308–317, 2003.

[Cai01] Jin-Yi Cai. $s_2^p \subseteq zpp^{NP}$. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, pages 620–629, 2001.

[CW04] Jin-Yi Cai and Osamu Watanabe. On proving circuit lower bounds against the polynomial-time hierarchy. *SIAM Journal on Computing*, 33(4):984–1009, 2004.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, 1993.

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 316–324, 2004.

[FSS84] Merrick Furst, James Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

[GST04] Oded Goldreich, Madhu Sudan, and Luca Trevisan. From logarithmic advice to single-bit advice. *Electronic Colloquium on Computational Complexity, TR04-093*, 2004.

[GZ97] Oded Goldreich and David Zuckerman. Another proof that bpp subseteq ph (and more). *Electronic Colloquium on Computational Complexity*, 4(45), 1997.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 6–20, 1986.

[HHT97] Yenjo Han, Lane Hemaspaandra, and Thomas Thierauf. Threshold computation and cryptographic security. *SIAM Journal on Computing*, 26(1):59–78, 1997.

[IKW02] 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.

[IM83] Oscar Ibarra and Shlomo Moran. Probabilistic algorithms for deciding equivalence of straight-line programs. *Journal of the ACM*, 30(1):217–228, 1983.

[Kan82] Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1):40–56, 1982.

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

[KL82] Richard Karp and Richard Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(2):191–209, 1982.

[KW98] Johannes Kobler and Osamu Watanabe. New collapse consequences of np having small circuits. *SIAM Journal on Computing*, 28(1):311–324, 1998.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the Association for Computing Machinery*, 39(4):859–868, 1992.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[O'D04] Ryan O'Donnell. Hardness amplification within np. *Journal of Computer and System Sciences*, 69(1):68–94, 2004.

[Raz85] Alexander Razborov. Lower bounds for the monotone complexity of some boolean functions. *Soviet Mathematics Doklady*, 31:354–357, 1985.

[Raz87] Alexander Razborov. Lower bounds on the size of bounded-depth networks over the complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[R.E79] R.E.Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of an International Symposium on Symbolic and Algebraic Manipulation*, pages 216–226, 1979.

[RR97] Alexander Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

[RS98] Alexander Russell and Ravi Sundaram. Symmetric alternation captures BPP. *Computational Complexity*, 7(2):152–162, 1998.

[Sch80] Jacob Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the Association for Computing Machinery*, 39(4):869–877, 1992.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual Symposium on Theory of Computing*, pages 77–82, 1987.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the xor lemma. *Journal of Computer System Sciences*, 62(2):236–266, 2001.

[Tod89] Seinosuke Toda. On the computational power of PP and $\oplus P$. In *30th Annual IEEE Symposium on Foundations of Computer Science*, pages 514–519, 1989.

[TV02] Luca Trevisan and Salil Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity*, volume 17, 2002.

[Val79a] Leslie Valiant. Completeness classes in algebra. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 249–261, 1979.

[Val79b] Leslie Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Ver92] Nikolai Vereshchagin. On the power of PP. In *Structure in Complexity Theory Conference*, pages 138–143, 1992.

[Vin05] Variyam Vinodchandran. A note on the circuit complexity of pp. *Theoretical Computer Science*, 347(1-2):415–418, 2005.

[Vio05] Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3):147–188, 2005.

[Wil85] Christopher Wilson. Relativized circuit complexity. *Journal of Computer and System Sciences*, 31(2):169–181, 1985.

[Yao82] Andrew Yao. Theory and application of trapdoor functions. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/