Lower Bounds for Swapping Arthur and Merlin*

Scott Diehl†
University of Wisconsin-Madison
sfdiehl@cs.wisc.edu

June 4, 2007

Abstract

We prove a lower bound for swapping the order of Arthur and Merlin in two-round Merlin-Arthur games using black-box techniques. Namely, we show that any AM-game requires time $\Omega(t^2)$ to black-box simulate MA-games running in time t. Thus, the known simulations of MA by AM with quadratic overhead, dating back to Babai's original paper on Arthur-Merlin games, are tight within this setting. The black-box lower bound also yields an oracle relative to which MA-TIME$[n] \not\subseteq$ AM-TIME$[o(n^2)]$.

Complementing our lower bounds for swapping Merlin in MA-games, we prove a time-space lower bound for simulations that drop Merlin entirely. We show that for any $c < \sqrt{2}$, there exists a positive d such that there is a language recognized by linear-time MA-games with one-sided error but not by probabilistic random-access machines with two-sided error that run in time n^c and space n^d. This improves recent results that give such lower bounds for problems in the second level of the polynomial-time hierarchy.

*This paper is the full version of an extended abstract to appear at RANDOM 2007.

†Supported by NSF Career award CCR-0133693 and Cisco Systems Distinguished Graduate Fellowship.

1 Introduction

Interactive protocols extend the traditional, static notion of a proof by allowing the verifier to enter into a conversation with the prover, wherein the goal is to convince the computationally limited verifier of a claim's validity in a probabilistic sense. Such an allowance affords the power to prove membership in what we believe to be significantly more difficult languages: Traditional polynomial-time proof systems (with a deterministic verifier) capture the class of languages recognizable in nondeterministic polynomial time, whereas interactive protocols have been shown to capture the entire class of languages recognizable in polynomial space [10, 17].

A particularly interesting subclass of interactive protocols is the class of public-coin protocols in which the number of communication rounds between the prover and the verifier is bounded by a constant— We know such protocols as *Arthur-Merlin games*. For example, graph nonisomorphism, which is not known to be in NP, has a two-round Arthur-Merlin game [7]. In fact, any language that has an Arthur-Merlin game with a constant number of rounds can be decided by an Arthur-Merlin game with only two rounds [1, 2]. Thus, there are at most two classes of languages recognized by Arthur-Merlin games, determined by the order in which the parties act: Those recognized by two-round games where the prover (Merlin) acts first, MA, and those recognized by two-round games where the verifier (Arthur) acts first, AM.

We know that the latter type of game is at least as powerful as the former: Any MA-game can be transformed into an AM-game recognizing the same language. Thus, we can prove at least as many languages when Arthur goes first as when Merlin goes first. This transformation does come at a cost, though: We must allow for some polynomial factor more time to verify the prover's claim in the resulting AM-game than in the original MA-game. For example, in Babai's transformation [1], one first reduces the error probability of the MA-game to exponentially small (by taking the majority vote of parallel trials) and then applies a union bound to show that, since the probability of proving an invalid claim is so small, switching the parties' order does not give the prover an unfair advantage. Since the error reduction necessitates verifying a number of trials of the MA-game that is linear in the number of bits sent by Merlin, this transformation incurs a quadratic-time overhead in general.

1.1 Time Overhead and MA Versus AM

Motivated by recent results in time-space lower bounds for probabilistic machines [4, 19], the main goal of this paper is to investigate the polynomial-time overhead needed in any such MA-to-AM transformation, and, in particular, whether or not it can be made subquadratic. We answer this question in the negative for a broad class of simulations, encompassing all currently known transformations: *black-box* simulations. Informally, a simulation is black-box if its outcomes are determined only by some number of calls to a given subroutine; In the case where we wish to simulate MA, this subroutine is the underlying predicate of an arbitrary MA-game.

Theorem 1 (Main Result). *For any time-constructible t, there is no black-box simulation of* MA*-games running in time t by* AM*-games running in time $o(t^2)$.*

Babai's simulation of MA by AM is indeed black-box, since it makes its decision by the majority vote of a linear number of trials of the MA-game. The same holds for all other known simulations, such as those using pseudorandom generators [13] and those using extractors [8]. Thus, Theorem 1 implies that those with a quadratic overhead (such as Babai's) are optimal within this setting;

No tweaking or optimization of other known techniques—even by improvements to extractor or pseudorandom generator parameters—can possibly yield better performance.

We point out that the black-box restriction on the lower bound is the best one can hope for without solving a (seemingly) much more difficult open problem: An unrestricted (non-black-box) version of Theorem 1 would imply a time hierarchy for both MA and AM. For example, the known simulation of MA by AM guarantees that any language with a linear-time MA-game also has a quadratic-time AM-game; On the other hand, a non-black-box lower bound would provide such a language that has no subquadratic-time AM-game, witnessing a hierarchy for AM. Therefore, removing the black-box restriction would be at least as hard as solving the long-standing open problem of proving a fully-uniform time hierarchy for Arthur-Merlin games [11].

Theorem 1 allows the construction of a relativized world in which linear-time MA-games can solve a language that subquadratic-time AM-games cannot.

Corollary 2. *There exists an oracle relative to which there is a language recognizable by a linear-time* MA*-game but not by any subquadratic-time* AM*-game.*

As far as we know, these results are the first to compare the running-time of MA-games to equivalent AM-games. Santha compared the power of MA and AM by showing that there is an oracle relative to which AM is not contained in MA [15], but this result does not concern the overhead of the opposite inclusion. Canetti et al. gave a lower bound for black-box samplers [3], which implies that the error reduction in Babai's simulation (or in Goldreich and Zuckerman's simulation via extractors [8]) cannot be done more efficiently in this setting; However, amplification is not the only black-box technique through which we can simulate MA—for example, one can use pseudorandom generators [13]—so this does not even suffice to lower bound all *known* techniques (while our result does).

1.2 Time-Space Lower Bounds and MA Versus BPP

As mentioned in Section 1.1, the lower bound of Theorem 1 is motivated by our investigation of lower bounds on the time required by a probabilistic, polynomial-time verifier to solve problems *without* the help of the prover (i.e., BPP). Since Boolean satisfiability is in NP but is widely held to require exponential time to solve probabilistically (with bounded error), we strongly expect the verifier to be severely handicapped by the prover's absence. Despite this, a nontrivial lower bound for satisfiability on probabilistic machines has eluded the community so far.

However, progress has been made in proving concrete lower bounds for such hard problems by restricting the machines solving them to use a small amount of space. For example, a recent line of work gives a time lower bound of $n^{1.759}$ for *deterministic* machines to solve satisfiability in subpolynomial ($n^{o(1)}$) space [4, 5, 20]. However, for *probabilistic* machines, we only know of such lower bounds for problems in the second level of the polynomial-time hierarchy: Diehl and Van Melkebeek proved a time lower bound of $n^{2-o(1)}$ for subpolynomial-space probabilistic machines to solve QSAT_2, the problem of verifying the validity of a quantified Boolean formula with one quantifier alternation [4]. We bring these lower bounds closer to satisfiability by deriving time-space lower bounds for simulations of MA by probabilistic machines.[1]

[1] This was independently discovered by Emanuele Viola and Thomas Watson [personal communication].

Theorem 3. *For every constant $c < \sqrt{2}$, there exists a constant $d > 0$ such that there is a language recognized by a linear-time* MA*-game with one-sided error that cannot be recognized by probabilistic machines with two-sided error running in time $O(n^c)$ and space $O(n^d)$.*

Theorem 3 follows as a corollary to the quadratic time-space lower bound for QSAT_2 on probabilistic machines by Diehl and Van Melkebeek. This is because a time-n^c probabilistic algorithm for linear-time MA-games yields a time-n^{c^2} probabilistic algorithm for linear-time in the second level of the polynomial-time hierarchy; Thus, a lower bound of $n^{2-o(1)}$ for QSAT_2 implies one of $n^{\sqrt{2}-o(1)}$ for the class linear-time MA-games. The argument requires some additional technical steps to achieve the lower bound for one-sided error MA-games.

At first glance, it may seem strange that we don't already have a lower bound for SAT; After all, the lower bound for QSAT_2 seems as though it should imply a lower bound for SAT since the two are intimately connected. Indeed, this is true in the *deterministic* setting: Similar to the above case of MA, a deterministic lower bound of n^2 for QSAT_2 implies one of $n^{\sqrt{2}}$ for SAT. However, this is not necessarily the case when we consider *probabilistic* machines. The main culprit is that the known arguments one uses to construct an algorithm for QSAT_2 from one for SAT utilize the simulation of MA by AM; By Theorem 1, this carries at least a quadratic-time overhead by black-box techniques. In this manner, we only know how to translate a probabilistic algorithm for SAT running in time n^c into one for QSAT_2 running in time n^{2c^2}; Thus, an n^2 lower bound for QSAT_2 gives nothing for SAT. In order to obtain a lower bound for SAT on probabilistic machines by this line of argument, we require either a stronger lower bound for QSAT_2 or a subquadratic simulation of MA by AM. Recalling Theorem 1, the latter is impossible to achieve with any known simulation. As such, the class MA is currently as close to NP as we can stretch the lower bound for QSAT_2.

1.3 Techniques

The proof of Theorem 1 is inspired by recent work of Viola [19], where he proves a quadratic-time lower bound for black-box simulations of BPP in the second level of the polynomial-time hierarchy. Viola's result involves a technique inspired by a switching lemma of Segerlind, Buss, and Impagliazzo [16]—later improved upon by Razborov [14]—which uses random restrictions that set very few variables.

The lower bound for BPP yields the interesting corollaries that black-box simulations of standard two-sided error AM- or MA-games by one-sided error games of the respective type require a quadratic-time overhead. This is because the two-sided error games contain BPP, while the one-sided error games are (by definition) contained in the second level of the polynomial-time hierarchy. However, since the transformation to the second level requires a quadratic-time overhead for *both* classes, this does not rule out a linear-overhead, black-box simulation of MA by AM in the standard two-sided error model. Therefore, something more is needed.

Our proof can be summed up as follows: First, we transform the setting from AM-games into systems of disjunctive normal-form (DNF) formulas. Notice that we can view the behavior of an AM-game after the verifier sends his message as a nondeterministic computation where each guess corresponds to a possible prover's message. The black-box setting enforces that the outcome after the nondeterministic guess is determined by some number of trials $k(n)$ of the underlying predicate of the MA-game being simulated. Thus, by the standard polynomial-time hierarchy-to-circuit connection [6], the result of the game after the verifier's move can be viewed as the output of an exponential-size, depth-two OR-of-ANDs circuit with bottom fan-in k— i.e., a k-DNF.

Next, we show that the ideas underlying the switching lemma of Segerlind et al. [16, 14] and its adaptation by Viola [19] present us with a dichotomy for any such k-DNF: Either (i) its terms are spread out over the input variables and it accepts very often on random inputs, or (ii) its terms are focused on a small set of variables whose influence is therefore elevated above the rest. Our proof takes advantage of this dichotomy by constructing two distributions on the outcomes of MA-games: One that generates mostly "no" instances and one that generates mostly "yes" instances. We show that, for small k, if a k-DNF is of type (i), then it accepts on most instances of the "no" distribution; If the k-DNF is of type (ii), then it depends (with non-negligible probability) on too few variables to discern the difference between the "yes" and "no" distributions.

We conclude by showing that this means the verifier's actions either put the prover in a position to wrongfully convince the verifier of false claims or prevent the prover from helping the verifier separate fact from fallacy with bounded error— either case is a failure. Therefore, the number of trials k made by the AM-game cannot be small, which leads to the lower bound.

1.4 Organization

The rest of the paper is organized as follows. Section 2 focuses on Theorem 1: Section 2.1 precisely defines the black-box model, while Section 2.2 gives the technical details of the proof. Section 2.3 provides some remarks about the proof, including refinements to the statement of Theorem 1 and comparisons to previous techniques. Section 2.4 gives the oracle result of Corollary 2. We switch to time-space lower bounds in Section 3, proving Theorem 3, and wrap up in Section 4 by discussing some directions for future work.

2 Black-Box Lower Bound

This section begins by precisely defining the models and promise problems to which our lower bound arguments apply.

2.1 Black-Box Model and the $\exists$-ApprMaj Problem

We define the class of languages recognizable by a time-bounded AM-game. For any time bound t, we say that a language $L \in$ AM-TIME$[t]$ if there exists a deterministic Turing machine V that takes three inputs and runs in time $t(n)$ such that for any x,

$$x \in L \quad \Rightarrow \quad \Pr_{|z|=t(n)}[\exists y \in \{0,1\}^{t(n)} \text{ such that } V(x,y,z)=1] \geq 2/3 \tag{1}$$

$$x \notin L \quad \Rightarrow \quad \Pr_{|z|=t(n)}[\exists y \in \{0,1\}^{t(n)} \text{ such that } V(x,y,z)=1] \leq 1/3, \tag{2}$$

where $n = |x|$ and the probabilities are taken over the uniform distribution.

Similarly, the class of languages recognized by MA-games bounded by time t, MA-TIME$[t]$, are the languages L where there is a deterministic, time $t(n)$ Turing machine V such that for any x,

$$x \in L \quad \Rightarrow \quad \exists y \in \{0,1\}^{t(n)} : \Pr_{|z|=t(n)}[V(x,y,z)=1] \geq 2/3 \tag{3}$$

$$x \notin L \quad \Rightarrow \quad \forall y \in \{0,1\}^{t(n)} : \Pr_{|z|=t(n)}[V(x,y,z)=1] \leq 1/3. \tag{4}$$

An AM- or MA-game has *one-sided error* if the acceptance probability in the case that $x \in L$ (i.e., the right-hand side of (1) or, respectively, (3)) is replaced by 1 in the above definitions. The resulting classes are referred to as AM-TIME$_1[t]$ and MA-TIME$_1[t]$, respectively.

We often refer to the Turing machine V as the underlying predicate, the string x as the problem input, y as the prover's message, and z as the verifier's message. The machine model we consider is that which allows *random-access* to the input and worktape (although our lower bounds hold for any fixed reasonable model), and our results assume that any time-bounds involved are constructible.

We now precisely define the notion of a black-box simulation. This is meant to capture any method of showing that MA is contained in AM by the design of a "modular" AM-game, into which one can plug *any* MA-game's underlying predicate V as a subroutine (i.e., it is oblivious to the MA-games it simulates[2]). Such a simulation uses only the results of queries to V to decide an answer and is indifferent to the computational details of how these answers are obtained: It should arrive at the right answer provided only that V satisfies the promise of an MA-game given by conditions (3) and (4). Therefore, a black-box AM-simulation of MA is an AM-game that solves the *promise problem* of whether or not a given subroutine satisfies (3) or (4) on input x.

We cast this promise problem as one on the (exponentially long) characteristic vector of the given subroutine on a fixed x. To do so, we first define the promise problem ApprMaj corresponding to the acceptance condition of a probabilistic computation with bounded error. This allows us to define the desired promise problem $\exists$-ApprMaj as one on matrices, where each row corresponds to a possible prover message, and each column corresponds to a possible verifier message.

Definition 4. ApprMaj *is the promise problem on Boolean strings where*

$$\begin{aligned} \text{ApprMaj}_Y &= \{r \mid \textit{at least } 2|r|/3 \textit{ bits of } r \textit{ are set to 1}\}\ , \\ \text{ApprMaj}_N &= \{r \mid \textit{at most } |r|/3 \textit{ bits of } r \textit{ are set to 1}\}. \end{aligned}$$

$\exists$-ApprMaj *is the promise problem on Boolean matrices where*

$$\begin{aligned} \exists\text{-ApprMaj}_Y &= \{M \mid \textit{at least one row of } M \textit{ belongs to } \text{ApprMaj}_Y\}, \\ \exists\text{-ApprMaj}_N &= \{M \mid \textit{every row of } M \textit{ belongs to } \text{ApprMaj}_N\}. \end{aligned}$$

Thus, the definition of an AM-game that efficiently black-box simulates MA fitting our intuition is one that solves $\exists$-ApprMaj on appropriately-sized matrices with few queries. We charge the simulation t time units for each query, where t is the running-time of the simulated MA-game, to capture the fact that a black-box simulation must calculate the bits of the $\exists$-ApprMaj instance on the fly by running trials of the MA-game.

Definition 5. *We say that an* AM*-game black-box q-simulates* MA*-games running in time $t(n)$ if there is a polynomial $p(n)$ and an oracle machine S such that for any input $M \in \{0,1\}^{2^t \times 2^t}$,*

$$\begin{aligned} M \in \exists\text{-ApprMaj}_Y &\Rightarrow \Pr_{|z|=p(n)}[\exists y \in \{0,1\}^{p(n)} \textit{ such that } S^M(y,z)=1] \geq 2/3 \\ M \in \exists\text{-ApprMaj}_N &\Rightarrow \Pr_{|z|=p(n)}[\exists y \in \{0,1\}^{p(n)} \textit{ such that } S^M(y,z)=1] \leq 1/3, \end{aligned}$$

where S queries M in at most $q(n)$ locations.

We say that an AM*-game running in time $t'(n)$ black-box simulates* MA*-games running in time $t(n)$ if there is an* AM*-game that black-box q-simulates such* MA*-games for $q = t'/t$ and $p = t'$.*

[2]We point out the difference to a relativizing inclusion, where we can have a different simulation for each oracle.

We prove the lower bound of Theorem 1 by showing that any AM-game black-box q-simulating MA-games running in time t must have $q = \Omega(t)$; Therefore, the running time of such a simulation must be at least $\Omega(t^2)$. In fact, the other computational aspects of the AM-simulation, such as the bounds on the message lengths or the running-time of the underlying predicate V, turn out to be irrelevant to the number of required queries. Therefore, we simplify the situation by proving the query lower bound via a lower bound on the *bottom fan-in* of depth-3 circuits for $\exists$-ApprMaj (where by bottom fan-in we mean the fan-in of the gates adjacent to the input literals). In particular, the circuits we consider have the same correspondence to AM-games as standard constant-depth circuits have to the polynomial-time hierarchy [6].

Definition 6. *An* AM*-circuit is a depth-3 circuit with literals as inputs (variables or their complements) whose output gate is an oracle gate for* ApprMaj *and each depth-2 subcircuit is an OR of AND's (i.e., a DNF); Furthermore, we are guaranteed that the input to the* ApprMaj *gate formed by the depth-2 subcircuits on any input X is either in* ApprMaj_Y *or* ApprMaj_N.

The aforementioned connection to AM-games is given by the following proposition.

Proposition 7. *If an* AM*-game black-box q-simulates* MA*-games running in time t, then there is a family of* AM*-circuits with bottom fan-in q computing* $\exists$-ApprMaj *on matrices of size $2^t \times 2^t$.*

Proof. We model the verifier's probabilistic move by the top ApprMaj gate, and the prover's nondeterministic move by the middle-layer's OR gate. The verification that takes place after the prover's message is sent is a function that depends on at most q oracle bits; Such a function can be decided by a depth-two OR-of-ANDs circuit with bottom fan-in q. Furthermore, the guarantee that the inputs to the top gate are either in ApprMaj_Y or ApprMaj_N is satisfied by the conditions imposed on the game by Definition 5. □

2.2 Proof of the Main Result

Proposition 7 allows us to prove Theorem 1 by showing a lower bound on the bottom fan-in of AM-circuits for $\exists$-ApprMaj.

Theorem 8. *Any* AM*-circuit solving* $\exists$-ApprMaj *on $2^t \times 2^t$ matrices has bottom fan-in $\Omega(t)$.*

As outlined in Section 1.3, the main idea behind the proof of Theorem 8 stems from an analysis of the k-DNF subcircuits of an AM-circuit on different input distributions. For an AM-circuit with bottom fan-in k to compute $\exists$-ApprMaj, Definition 6 requires most of its k-DNF subcircuits to accept when the input is drawn from a distribution producing mostly "yes" instances of $\exists$-ApprMaj and most to reject when the input is drawn from a distribution producing mostly "no" instances. By the union bound, a nontrivial fraction of the subcircuits output the correct answer most of the time for *both* the "yes" and "no" distributions. We construct specific "yes" and "no" distributions such that any k-DNF with $k = o(t)$ cannot do both at the same time; Therefore, such an AM-circuit for $\exists$-ApprMaj cannot exist.

We begin by defining the distributions producing mostly "yes" instances and "no" instances of $\exists$-ApprMaj that we use to prove the lower bound.

Definition 9. *Let $\mathcal{D}_N^{n \times m}$ be the distribution on $n \times m$ matrices obtained by assigning each entry independently to 1 with probability 1/6 and 0 with probability 5/6.*

Furthermore, let $\mathcal{D}_Y^{n\times m}$ be the distribution on $n \times m$ matrices obtained by choosing one row uniformly at random and setting each entry within this row to 1, while the remaining entries are set as in $\mathcal{D}_N^{n\times m}$.

When n and m are clear from context, we simply use $\mathcal{D}_Y$ and $\mathcal{D}_N$.

We claim that when $m = \Omega(\log n)$, $\mathcal{D}_Y$ generates instances in $\exists$-ApprMaj_Y and $\mathcal{D}_N$ generates instances in $\exists$-ApprMaj_N with high probability.

Claim 10. *There exists a constant $\alpha > 0$ such that for large enough n and $m \geq \alpha \log n$,*

$$\Pr_{M\in\mathcal{D}_Y^{n\times m}}[M \in \exists\text{-ApprMaj}_Y] = 1 \text{ and } \Pr_{M\in\mathcal{D}_N^{n\times m}}[M \in \exists\text{-ApprMaj}_N] \geq (1 - 1/n).$$

Proof. The case of $\mathcal{D}_Y$ is trivial, since we always set some row to all 1's. For the case of $\mathcal{D}_N$, we expect there to be $m/6$ 1's in each row. Therefore, a row of M has more than $m/3$ 1's (i.e., is in not ApprMaj_N) with probability at most $2^{-\beta m}$ for some $\beta > 0$ by the Chernoff bound. Then the probability that *some* row is not in ApprMaj_N (and therefore, $M \notin \exists$-ApprMaj_N) is at most $n2^{-\beta m}$ by the union bound; The claim follows for $\alpha = 2/\beta$. □

To analyze the relationship between the probability that a given k-DNF φ accepts on $\mathcal{D}_Y$ and the probability that it accepts on $\mathcal{D}_N$, we introduce a "lazy" procedure, Eval, that tries to evaluate $\varphi(M)$ (see Figure 1). Given φ, input M, and a parameter s, Eval attempts to reduce the problem of evaluating a k-DNF to that of evaluating a $(k-1)$-DNF by querying the values of a set of variables that *cover* the terms of φ (a cover, or hitting set, of a DNF ψ is a subset Γ of the variables such that each term of ψ contains at least one variable in Γ). Eval then restricts φ appropriately to obtain a $(k-1)$-DNF φ', whose evaluation on the remaining variables is the same as the evaluation of φ, and recurses. However, Eval is "lazy" in the sense that it refuses to query more than s variables at each step; If at any step it becomes impossible to achieve the above term-size reduction in such few queries, Eval simply gives up and outputs the current subformula. Thus, Eval has three possible outputs: 0, 1, or a restriction of φ that has a large minimum cover. We point out that if Eval outputs 0, then $\varphi(M) = 0$, and if Eval outputs 1, then $\varphi(M) = 1$; However, the converse does not hold in either case due to Eval's laziness.

Figure 1: The "lazy" procedure, Eval.

Procedure **Eval**(φ, M, s)
If $\varphi \equiv 0$, **output** 0.
Else if $\varphi \equiv 1$, **output** 1.
Else if every cover of φ has size greater than s, **output** φ.
Else
$\Gamma \leftarrow$ a cover of φ of size at most s.
Query the variables in Γ to obtain the partial assignment $\Gamma(M)$.
$\varphi' \leftarrow \varphi|_{\Gamma(M)}$.
Eval(φ', M, s).

Our proof requires the cover size s to balance two desires: (i) We need s large enough so that if Eval outputs a formula ψ, then ψ is almost certainly satisfied by an input from $\mathcal{D}_N$, and (ii) we need s small enough so that Eval has not queried many variables when it arrives at a 0/1 answer.

Property (i) ensures that the non-Boolean output case of Eval cannot happen very often for φ a k-DNF in an AM-circuit computing $\exists$-ApprMaj on $\mathcal{D}_N$, so the output of $\text{Eval}(\varphi, M, s)$ almost always matches the evaluation of $\varphi(M)$ when $M \in \mathcal{D}_N$. Property (ii) guarantees that the event of $\text{Eval}(\varphi, M, s)$ answering either 0 or 1 depends very weakly on most rows, which is not enough for Eval to distinguish the one-row difference between samples from $\mathcal{D}_Y$ and $\mathcal{D}_N$. Therefore, if φ behaves well on $\mathcal{D}_N$ and outputs 0 most of the time, then $\text{Eval}(\varphi, M, s)$ not only outputs 0 most of the time for $M \in \mathcal{D}_N$ by (i), but also for $M \in \mathcal{D}_Y$ by (ii), so φ rejects too often on $\mathcal{D}_Y$.

We first quantify the effect of property (i) on the correctness of Eval on $\mathcal{D}_N$.

Lemma 11. *If φ is a k-DNF, then*

$$\Pr_{M \in \mathcal{D}_N}[\text{Eval}(\varphi, M, s) \neq 0] \leq \frac{\Pr_{M \in \mathcal{D}_N}[\varphi(M) = 1]}{1 - e^{-\frac{s}{k6^k}}}.$$

The proof begins by showing that any DNF ψ output by Eval must have many variable-disjoint terms due to its large cover size. These terms present too many independent events that must align in order for ψ to reject reliably on $\mathcal{D}_N$.

Lemma 12. *Let φ be a k-DNF. Then*

$$\Pr_{M \in \mathcal{D}_N}[\varphi(M) = 0 | \text{Eval}(\varphi, M, s) \notin \{0, 1\}] \leq e^{-\frac{s}{k6^k}}.$$

Proof. Suppose that Eval outputs some formula ψ. Notice that $\psi \not\equiv 0$ is a restriction of φ, so it is a k'-DNF for $k' \leq k$. Thus, a term of ψ is satisfied by $\mathcal{D}_N$ with probability at least $1/6^k$. We claim that ψ must have more than s/k variable-disjoint terms Γ. The lemma follows if this is the case, since the events that a term in Γ is satisfied are independent, so

$$\Pr_{M \in \mathcal{D}_N}[\psi(M) = 0] \leq \Pr_{M \in \mathcal{D}_N}[\text{Each clause in } \Gamma \text{ is not satisfied}] \leq (1 - 1/6^k)^{s/k} \leq e^{-\frac{s}{k6^k}}.$$

To prove the claim, consider a maximal set Γ of variable-disjoint terms of ψ. There are at most $|\Gamma|k$ variables in these terms, and they must form a cover of ψ; If they do not, then there is a term t that is not covered by the variables of Γ, i.e., t does not contain any of the variables in Γ. Thus, $\Gamma \cup \{t\}$ is a set of variable-disjoint terms, which contradicts the maximality of Γ. Furthermore, since ψ is a formula output by Eval, it cannot have a cover of size at most s. Therefore, we have that $|\Gamma|k > s$, so $|\Gamma| > s/k$ as claimed. □

Thus, since the formulas on which Eval does not give a 0/1 output are very likely to accept $\mathcal{D}_N$ by Lemma 12, Eval cannot give such an output often on a formula φ that largely rejects $\mathcal{D}_N$. Therefore, Eval must correctly output 0 for φ on most of these instances despite its laziness, yielding Lemma 11.

Proof of Lemma 11. We have noted that if $\varphi(M) = 1$, then $\text{Eval}(\varphi, M, s)$ must output either 1 or a formula, so

$$\Pr_{M \in \mathcal{D}_N}[\varphi(M) = 1] = \Pr_{M \in \mathcal{D}_N}[\text{Eval} = 1] + \Pr_{M \in \mathcal{D}_N}[\text{Eval} \notin \{0, 1\}] \cdot \underbrace{\Pr_{M \in \mathcal{D}_N}[\varphi(M) = 1 | \text{Eval} \notin \{0, 1\}]}_{(*)}.$$

By Lemma 12, we have that $(*)$ is at least $(1 - e^{-\frac{s}{k6^k}})$, so we conclude that

$$\begin{aligned}\Pr_{M\in\mathcal{D}_N}[\varphi(M)=1] &\geq \left(\Pr_{M\in\mathcal{D}_N}[\text{Eval}=1] + \Pr_{M\in\mathcal{D}_N}[\text{Eval}\notin\{0,1\}]\right)(1-e^{-\frac{s}{k6^k}}) \\ &= \Pr_{M\in\mathcal{D}_N}[\text{Eval}\neq 0](1-e^{-\frac{s}{k6^k}}).\end{aligned}$$

□

We now turn to quantify the effect of property (ii) on the relationship between Eval's behavior on $\mathcal{D}_Y$ and $\mathcal{D}_N$.

Lemma 13. *Let φ be a k-DNF. Then for any $\delta \geq \frac{sk}{n}$,*

$$\Pr_{M\in\mathcal{D}_Y}[\text{Eval}(\varphi,M,s)=0] \geq (1-\frac{sk}{\delta n})\left(\Pr_{M\in\mathcal{D}_N}[\text{Eval}(\varphi,M,s)=0]-\delta\right).$$

Proof. Recall that when the formula φ passed to Eval has small cover size, Eval reduces the term size of φ by one to obtain φ' (since Γ is a cover of φ). Therefore, Eval can make at most k recursive calls on a k-DNF. At each such call, Eval queries at most s variables, for a total of at most sk variables queried during the entire run. Different outcomes for the sample M cause Eval to form different subformulas φ' with (possibly) different covers, so the variables queried by Eval follow some probability distribution determined by the input distribution. We show that most variables, and in fact most rows, are queried with very small probability.

For a fixed k-DNF φ, s, and $\delta > 0$, define B as

$$B \doteq \{i | \Pr_{M\in\mathcal{D}_N}[\text{Eval}(\varphi,M,s) \text{ queries row } i] \geq \delta\}, \tag{5}$$

the set of rows queried by Eval with probability at least δ. Since any individual run of Eval queries at most sk variables, we have that

$$\sum_i \Pr_{M\in\mathcal{D}_N}[\text{row } i \text{ is queried by Eval}(\varphi,M,s)] \leq sk.$$

Therefore, B cannot be too large:

$$|B| \leq \frac{sk}{\delta}. \tag{6}$$

So (5) and (6) tell us that, for most rows i, Eval queries a variable in row i with very small probability when $\delta > \frac{sk}{n}$.

We would like to isolate the cases of $\mathcal{D}_Y$ that choose a row outside of B to set to 1. Towards this end, define D_Y^i to be the distribution on $n \times m$ matrices where each entry in row i is assigned 1 and each other entry is assigned 0 with probability 5/6 (and 1 otherwise). Then sampling M from $\mathcal{D}_Y$ is equivalent to choosing $i \in \{1,\ldots,n\}$ at random and sampling M from $\mathcal{D}_Y^i$. This allows us to divide the probability that Eval rejects on inputs from $\mathcal{D}_Y$ into cases by row. Discarding the rows in B, we have

$$\Pr_{M\in\mathcal{D}_Y}[\text{Eval}(\varphi,M,s)=0] \geq \sum_{i\notin B} \Pr_{M\in\mathcal{D}_Y^i}[\text{Eval}(\varphi,M,s)=0]/n. \tag{7}$$

Now consider how $\text{Eval}(\varphi,M,s)$ differs when M is drawn from $\mathcal{D}_N$ and when it is drawn from $\mathcal{D}_Y^i$ where $i \notin B$. Notice that the only time that Eval queries variables which are distributed

differently in $\mathcal{D}_N$ and $\mathcal{D}_Y^i$ is when it queries a variable in row i. We know that this happens with probability less than δ under $\mathcal{D}_N$ by (5); This is also the case under $\mathcal{D}_Y^i$, since the distribution is identical to $\mathcal{D}_N$ outside of row i. Thus, the outcome of Eval on $\mathcal{D}_N$ and $\mathcal{D}_Y^i$ can be different on at most a δ fraction of inputs for $i \notin B$,

$$\Pr_{M \in \mathcal{D}_Y^i}[\text{Eval}(\varphi, M, s) = 0] \geq \Pr_{M \in \mathcal{D}_N}[\text{Eval}(\varphi, M, s) = 0] - \delta. \tag{8}$$

The lemma follows by combining (6), (7), and (8). □

The two properties captured by Lemma 11 and Lemma 13 formalize the contradictory dichotomy that allows us to prove Theorem 8.

Proof of Theorem 8. Let $n = m = 2^t$ and C be an AM-circuit with bottom fan-in k purported to solve $\exists$-ApprMaj. Without loss of generality, we assume that 9/10 of C's DNF subcircuits accept on $M \in \exists\text{-ApprMaj}_Y$ while 9/10 reject on $M \in \exists\text{-ApprMaj}_N$—this can be achieved by naïve amplification while increasing the bottom fan-in by only a constant factor.

Consider sampling M from $\mathcal{D}_Y$ or $\mathcal{D}_N$ and a random depth-2 subcircuit φ connected to the output gate of C. Then by Claim 10,

$$\Pr_{M \in \mathcal{D}_Y, \varphi \in_U C}[\varphi(M) = 1] \geq \frac{9}{10} \text{ and } \Pr_{M \in \mathcal{D}_N, \varphi \in_U C}[\varphi(M) = 0] \geq \frac{9}{10}(1 - \frac{1}{n}) > 5/6$$

for large enough n. Therefore, the probability that a random subcircuit is correct on most instances of $\mathcal{D}_Y$ is greater than 1/2 (and similarly for $\mathcal{D}_N$):

$$\Pr_{\varphi \in_U C}[\Pr_{M \in \mathcal{D}_Y}[\varphi(M) = 1] > 2/3] > 1/2 \text{ and } \Pr_{\varphi \in_U C}[\Pr_{M \in \mathcal{D}_N}[\varphi(M) = 0] > 2/3] > 1/2.$$

Thus, by a union bound, there is a k-DNF φ^* where

$$\Pr_{M \in \mathcal{D}_Y}[\varphi^*(M) = 0] \leq 1/3 \text{ and } \Pr_{M \in \mathcal{D}_N}[\varphi^*(M) = 1] \leq 1/3 \tag{9}$$

Our proof proceeds by showing that such a k-DNF $\varphi*$ cannot exist when $k = o(\log n)$. By Lemma 11, we have that

$$\Pr_{M \in \mathcal{D}_N}[\text{Eval}(\varphi^*, M, s) = 0] \geq 1 - 1/\left(3(1 - e^{-\frac{s}{k6^k}})\right). \tag{10}$$

Plugging (10) into Lemma 13 gives

$$\Pr_{M \in \mathcal{D}_Y}[\text{Eval}(\varphi^*, M, s) = 0] \geq \underbrace{(1 - \frac{sk}{\delta n})}_{(*)} \left(1 - 1/\left(3\underbrace{(1 - e^{-\frac{s}{k6^k}})}_{(**)}\right) - \delta\right). \tag{11}$$

We claim that for small enough k, we can set δ and s so that the right-hand side of (11) is at least 1/2. Since Eval outputs 0 no more often than $\varphi^*(M) = 0$, this contradicts that $\Pr_{M \in \mathcal{D}_Y}[\varphi^*(M) = 0] \leq 1/3$ (from (9)), completing the lower bound for such k.

All that remains is to find the largest k so that we can choose δ and s appropriately. In order for the right-hand side of (11) to be large enough, we need $(*)$ and $(**)$ to be close to 1 and δ to be small. The condition on $(*)$ requires $s = O(\frac{\delta n}{k})$, while the condition on $(**)$ requires $s = \Omega(k6^k)$. Choosing δ to be a small constant, say, $\delta = 1/100$, we see that such a setting is possible when $k^2 6^k = O(n)$— i.e., when $k = c \log n$ for small enough c. □

2.3 Remarks on Theorem 1

We now highlight a few details of the preceding analysis. First, notice that the time complexity of the simulated MA-game's verification procedure V enters into the running-time of the AM-simulation only when accounting for the resources needed to compute each query in the black-box setting of Definition 5— The circuit lower bound of Theorem 8 depends solely on the dimensions of the instance of $\exists$-ApprMaj, which correspond to the number of bits sent by the prover and verifier in the MA-game.

Additionally, notice that the only place that the number of columns m in the instance of $\exists$-ApprMaj (corresponding to the number of bits flipped by the verifier in an MA-game) enters into the proof is in Proposition 10, where a lower bound on m is needed to ensure that the distribution $\mathcal{D}_N$ behaves properly. Thus, Theorem 8 actually gives a lower bound of $\Omega(n)$ for $n \times m$ instances of $\exists$-ApprMaj, where $m \geq \alpha \log n$ for α as defined in Proposition 10. This implies that the black-box lower bound of Theorem 1 holds even when simulating MA-games where the verifier flips very few bits.

The above two observations show that the crucial factor in determining the number of queries needed by an AM-game black-box simulating MA is the *number of bits sent by the prover*. We point out that this behavior is shared by the known simulations of MA by AM (such as Babai's).

Furthermore, the instances produced by $\mathcal{D}_Y$ satisfy an even stricter promise than $\exists$-ApprMaj$_Y$, since *every* bit in some row is set to 1. This corresponds to the promise on the "yes" side fulfilled by MA-games with *one-sided error*. Therefore, Theorem 1 holds even for simulations of the weaker class of time-t MA-games with one-sided error [3]

Finally, we take a moment to compare our techniques to those of Viola's [19]. Our approach is similar in that it uses ideas inspired by recent small-restriction switching lemmas [16, 14] to show that k-DNF's behave similarly enough on certain "yes" and "no" instances of the problem in question. In the setting of normal depth-3 circuits that Viola considers (for the problem ApprMaj), it is sufficient to give an argument that explicitly constructs a "yes" instance rejected by one particular depth-2 subcircuit (in the case where the top gate is AND) assuming that the subcircuit behaves correctly on "no" instances. However, one incorrect subcircuit is insufficient to fool an AM-circuit (as required by our setting), since the top gate only rejects if *most* subcircuits reject. Our approach addresses this issue via the aforementioned *probabilistic* construction of the bad "yes" instances. By producing instances that are independent of the specific subcircuits they are designed to fool, we simultaneously violate many more of them than we could by an explicit construction.

The bounded error of the AM-circuit setting also leads to a quantitatively better statement of the lower bound than in the Viola's standard depth-3 setting: The latter result states that no depth-3 circuit can compute ApprMaj with small bottom fan-in unless it is very large; In contrast, Theorem 8 states that no AM-circuits with small bottom fan-in can compute $\exists$-ApprMaj, *regardless of their size*.

2.4 Oracle Separation

In this section, we prove Corollary 2 by constructing an oracle relative to which there is a language recognizable by a linear-time MA-game but not by any subquadratic-time AM-game.

[3] Since the usual definition of an MA-game has two-sided error and applies the same bound to the message bits and the time bound of the verification procedure, we chose not to draw any of these distinctions in Theorem 1.

Formally, a *relativized* MA- *or* AM-*game* is one where the verification procedure V is given oracle access to some set A. We denote the class of languages recognized by such games relative to A in time t as MA-TIME$^A[t]$ and AM-TIME$^A[t]$. It is important to note that we follow the convention that V's oracle tape is erased after each query. This creates a situation that is already very similar to the black-box setting: It enforces a time requirement to make each query, namely, $|x|$ time is needed to query membership of a string x in A.

We view Theorem 8 as ensuring that a relativized AM-game requires $\Omega(n)$ queries at length n to decide whether or not A encodes an $n \times n$ instance of $\exists$-ApprMaj$_Y$ or $\exists$-ApprMaj$_N$; Therefore, since it takes time $\Omega(n)$ to write down each query, such an AM-game requires time $\Omega(n^2)$. However, it is easy for a relativized MA-game to decide in linear time which instance A encodes. The proof uses this argument to diagonalize in stages against each possible relativized AM-game.

Proof of Corollary 2. We construct an oracle A so that the characteristic vector of A at every even input length $n = 2m$, when viewed in row-major order as an $m \times m$ matrix, either encodes an instance of $\exists$-ApprMaj$_Y$ or $\exists$-ApprMaj$_N$. Notice that this setting allows the following game for deciding which instance is encoded to satisfy the promise required of an MA-game: The prover sends a row number, the verifier guesses a column number, and the result is whether this entry is in the oracle. Thus, for A as above and

$$L^A = \{0^n | n = 2m \text{ and } \exists x \in \{0,1\}^m \text{ s.t. } \Pr_{y \in_U \{0,1\}^m}[xy \in A] \geq 2/3\},$$

we have $L^A \in$ MA-TIME$^A[n]$.

On the other hand, Theorem 8 requires $\Omega(n)$ queries for an AM-game to decide $\exists$-ApprMaj, and hence $\Omega(n^2)$ time in our oracle model. Thus, we can choose A to encode instances of $\exists$-ApprMaj diagonalizing against AM-TIME$^A[o(n^2)]$. Specifically, let $P_1, P_2, \ldots$ be an enumeration of all relativized AM-TIME$[o(n^2)]$-games. (Note that this enumeration is not computable due to the promise the machines must fulfill; However, this does not matter for our construction.) We construct A in stages, so that at stage i we fix A at some (previously unset) input length n^* so that P_i^A fails on length n^*; Such an $\frac{n^*}{2} \times \frac{n^*}{2}$ instance of $\exists$-ApprMaj exists (for sufficiently large n^*) by Theorem 8. When considering the next machine, we choose an input length of at least $\omega((n^*)^2)$ to ensure A is not changed on any values that P_i^A can query on input length n^*. (We set any even input length unset by this procedure to encode, say, an instance of $\exists$-ApprMaj$_N$ to ensure that A satisfies the necessary properties to be decidable in MA-TIME$^A[n]$.)

Therefore, A satisfies $L^A \in$ MA-TIME$^A[n]$ and $L^A \notin$ AM-TIME$^A[o(n^2)]$ by construction. □

We point out that Corollary 2 can be extended in the same manner as Theorem 1 describe in Section 2.3.

3 Time-Space Lower Bound

In this section, we turn from the black-box model to the space-bounded model and prove the time-space lower bounds for probabilistic simulations of MA given by Theorem 3. As mentioned in Section 1.3, Theorem 3 follows as a corollary to the following lower bound of [4]:

Theorem 14 ([4]). *For any constant $c < 2$, there exists a positive constant d such that there is a language in Σ_2TIME$[n]$ that cannot be solved by probabilistic machines with two-sided error running in time n^c and space n^d.*

We extend this lower bound to MA in a proof by contradiction: If the desired lower bound does not hold for MA, then we show that $\Sigma_2\text{TIME}[n]$ can be computed efficiently enough to contradict Theorem 14. This requires a slight improvement of a lemma from [4] to achieve an efficient simulation of two-sided, space-bounded probabilistic computations by MA-games with one-sided error. Let $\text{BPTISP}[t, s]$ denote the class of languages recognized by a probabilistic machine with two-sided error running in time t and space s:

Lemma 15. *For any time bound t and space bound s,*

$$\text{BPTISP}[t, s] \subseteq \text{MA-TIME}_1[ts \text{ polylog}(t)].$$

Proof. In [4], a careful analysis of Lautemann's simulation of BPP in Σ_2^p [9] and Nisan's space-bounded pseudorandom generator [12] were used to show that $\text{BPTISP}[t, s] \subseteq \Sigma_2\text{TIME}[ts \text{ polylog}(t)]$. Recall that Lautemann's simulation relies on the fact that, provided it has small enough error, a probabilistic algorithm accepts an input if and only if the set of accepting random strings is large enough so that there is a small number of shifts that covers the entire universe of random strings. By further requiring the error be smaller by a factor of 1/3 (which can be achieved while only increasing the running time by a constant factor), the condition on the rejecting side is strengthened to say that every small set of shifts cannot cover even a third of the universe of random strings. This ensures bounded error on the "no" side of the simulation, allowing us to portray it as an MA-game with one-sided error. The improvements of [4] retain this property, placing the simulation in $\text{MA-TIME}_1[ts \text{ polylog}(t)]$. □

With this simulation in hand, we are ready to prove the lower bound for MA.

Proof of Theorem 3. We assume that linear-time MA-games (with one-sided error) can indeed be simulated by small-space probabilistic machines (with two-sided error) running in time $O(n^c)$. Specifically, the assumption is that

$$\text{MA-TIME}_1[n] \subseteq \text{BPTISP}[n^c, n^d], \tag{12}$$

for some constants $c \geq 1$ and $d > 0$. Notice that, since probabilistic classes are closed under complement, the hypothesis (12) gives that co-MA can be efficiently simulated by space-bounded probabilistic machines; An important corollary is that this provides such simulations for conondeterministic computations. In turn, the simulation given by Lemma 15 yields efficient MA-games with one-sided error for coNP:

$$\text{coNTIME}[n] \subseteq \text{BPTISP}[n^c, n^d] \subseteq \text{MA-TIME}_1[n^{c+d} \text{ polylog}(n)]. \tag{13}$$

Consider an arbitrary Σ_2-machine N running in linear time. We view this as a game where the prover goes first and, instead of a probabilistic verifier (as in MA), we have a conondeterministic "verifier"— On input x, the prover sends a string y of length n, and the outcome of the game is determined by a linear-time conondeterministic computation performed by the verifier on input (x, y). However, (13) allows the conondeterministic verification to be done by a one-sided error MA-game at the cost of raising the running-time to the power of $c + d$ (modulo a polylog factor). Thus, by asking the prover for the original proof *and* that required by the MA-simulation of the conondeterministic stage, we have devised an MA-simulation of N with one-sided error. In turn,

the hypothesis (12) provides that this game can be simulated by a space-bounded probabilistic machine by again raising the running-time to the power of c. All in all, we have that

$$\Sigma_2\text{TIME}[n] \subseteq \text{MA-TIME}_1[n^{c+d}\text{polylog}(n)] \subseteq \text{BPTISP}[n^{c^2+cd}\text{polylog}(n), n^{cd+d^2}],$$

which contradicts Theorem 14 when $c^2 < 2$ and d is small enough. □

4 Future Work

The most compelling open problem related to this work is determining whether or not a subquadratic-overhead simulation of MA by AM is at all possible— i.e., to either prove a general quadratic lower bound for the overhead of *any* simulation of MA by AM or to use some non-black-box technique to actually achieve a subquadratic AM-simulation. Either case yields something interesting: In the former case, one gets a hierarchy for MA and AM (see Section 1.1); In the latter case, one makes progress towards time-space lower bounds for SAT on probabilistic machines (see Section 1.2).

Furthermore, we ask if there are any common restrictions on the verifier that a non-black-box approach could take advantage of to allow a subquadratic simulation of a restricted class of MA-games. Especially interesting is the case of a space-bounded verifier: A subquadratic AM-simulation that exploits the verifier's space bound could also be used to achieve time-space lower bounds for SAT on probabilistic machines.

Another direction is to extend the black-box lower bound to relationships between other complexity class inclusions where the best known simulation has quadratic overhead. In particular, inclusions that proceed by amplifying the error of a probabilistic computation to exponentially small seem particularly amenable, since we already know that the amplification cannot be done faster in a black-box manner [3]. We propose finding black-box lower bounds for the overhead of inclusions such as $\text{MA} \subseteq \text{PP}$, $\oplus \cdot \text{BPP} \subseteq \text{BP} \cdot \oplus\text{P}$, etc. The latter is particularly interesting, since it applies to the time-overhead of the collapse of the polynomial-time hierarchy to $\text{BP} \cdot \oplus\text{P}$ in Toda's Theorem [18]; It is currently unknown how to achieve such a collapse with sublinear factor increase in time per alternation, which could yield new time-space lower bounds for counting problems.

Acknowledgments

The author would like to thank Bess Berg, Jeff Kinne, and Emanuele Viola for their helpful discussions, and especially Dieter van Melkebeek for his extensive advice, support, and enthusiasm. We would also like to thank the anonymous referees for their comments.

References

[1] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on the Theory of Computing*, pages 421–429. ACM, 1985.

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

[3] R. Canetti, G. Even, and O. Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53:17–25, 1995.

[4] S. Diehl and D. van Melkebeek. Time-space lower bounds for the polynomial-time hierarchy on randomized machines. *SIAM Journal on Computing*, 36(3):563–594, 2006.

[5] L. Fortnow, R. Lipton, D. van Melkebeek, and A. Viglas. Time-space lower bounds for satisfiability. *Journal of the ACM*, 52:835–865, 2005.

[6] M. L. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[7] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38:691–729, 1991.

[8] O. Goldreich and D. Zuckerman. Another proof that BPP $\subseteq$ PH (and more). Technical Report TR-97-045, Electronic Colloquium on Computational Complexity, 1997.

[9] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17:215–217, 1983.

[10] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39:859–868, 1992.

[11] D. van Melkebeek and K. Pervyshev. A generic time hierarchy for semantic models with one bit of advice. In *Proceedings of the 21st IEEE Conference on Computational Complexity*, pages 129–142. IEEE, 2006.

[12] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12:449–461, 1992.

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

[14] A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. Unpublished manuscript, 2002–2003.

[15] M. Santha. Relativized Arthur-Merlin versus Merlin-Arthur games. *Information and Computation*, 80(1):44–49, 1990.

[16] N. Segerlind, S. Buss, and R. Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM Journal on Computing*, 33(5):1171–1200, 2004.

[17] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39:869–877, 1992.

[18] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[19] E. Viola. On probabilistic time versus alternating time. Technical Report TR-05-137, Electronic Colloquium on Computational Complexity, 2005.

[20] R. Williams. Better time-space lower bounds for SAT and related problems. In *Proceedings of the 20th IEEE Conference on Computational Complexity*, pages 40–49. IEEE, 2005.