Uniform hardness vs. randomness tradeoffs for Arthur-Merlin games

Dan Gutfreund
School of Computer Science and Engineering,
The Hebrew University of Jerusalem, Israel, 91904
danig@cs.huji.ac.il.

Ronen Shaltiel*
Dept. of Applied Mathematics and Computer Science,
Weizmann Institute of Science, Rehovot, 76100, Israel.
ronens@wisdom.weizmann.ac.il

Amnon Ta-Shma
Computer Science Dept,
Tel-Aviv University, Israel, 69978
amnon@post.tau.ac.il.

Abstract

Impagliazzo and Wigderson proved a uniform hardness vs. randomness "gap result" for BPP. We show an analogous result for AM: Either Arthur-Merlin protocols are very strong and everything in $E = DTIME(2^{O(n)})$ can be proved to a sub-exponential time verifier, or else Arthur-Merlin protocols are weak and every language in AM has a polynomial time nondeterministic algorithm in the uniform average-case setting (i.e., it is infeasible to come up with inputs on which the algorithm fails). For the class $AM \cap coAM$ we can remove the average-case clause and show under the same assumption that $AM \cap coAM = NP \cap coNP$.

A new ingredient in our proof is identifying a novel resiliency property of hardness vs. randomness tradeoffs. We observe that the Miltersen-Vinodchandran generator has this property.

1 Introduction

One of the most basic goals of computational complexity is understanding the relative power of probabilistic complexity classes such as BPP, MA and AM. In particular, a long line of research is aimed at showing that randomness does not add substantial computational power. Much research is aimed at achieving this by using the mildest possible unproven assumptions.

One very fruitful sequence of results uses the "Hardness versus Randomness" paradigm, first suggested by Blum and Micali [9, 36]. The approach is to show that one can take a Boolean function that is hard for small circuits and use it to construct a *pseudo-random generator* that "stretches" a short string of truly random bits into a long string of "pseudo-random bits" that cannot be distinguished from uniform by small circuits. Such generators allow deterministic simulation of probabilistic classes. Loosely speaking, these constructions differ in:

- The type of circuits "fooled" by the generator. To derandomize BPP and MA one needs to "fool" deterministic circuits, and to derandomize AM one needs to fool co-nondeterministic circuits.
- The "stretch" of the generator. Generators with polynomial "stretch" (t bits to t^c bits) are called "low-end" generators and give subexponential time deterministic simulation (e.g., BPP $\subseteq$ SUBEXP $= \cap_{\delta>0}$DTIME(2^{n^δ}) or AM $\subseteq$ NSUBEXP $= \cap_{\delta>0}$NTIME(2^{n^δ})). Generators with exponential stretch (t bits to $2^{\Omega(t)}$ bits) are called "high-end" generators and give polynomial time deterministic simulation (e.g., BPP = P or AM = NP).
- The precise assumption on the hard function. Typically "High-end" generators require lower bounds on larger circuits (circuits of size $2^{\Omega(n)}$) whereas "Low-end" generators may require only super-polynomial lower bounds. Generators that fool co-nondeterministic circuits typically require hardness for co-nondeterministic circuits.

Today, after a long line of research [9, 36, 28, 6, 18, 33, 23, 25, 16, 17, 32, 35], we have powerful and elegant constructions of "low-end" and "high-end" generators

*Research supported by the Koshland Scholarship.

that derandomize BPP, MA and AM using "necessary assumptions" (i.e., assumptions that are implied by the existence of pseudo-random generators). The reader is referred to a recent survey paper on derandomization for more details [21].

All the above mentioned Hardness vs. Randomness tradeoffs require a *uniformly computable* function that is hard against a *non-uniform* class of circuits [1]. In our current state of knowledge such a non-uniform lower bound seems beyond our reach. In contrast, Impagliazzo and Wigderson [19] (see also [34]) showed derandomization under a uniform assumption. They proved a "gap theorem" for BPP: Loosely speaking, they showed that either BPP = EXP (i.e., BPP is as strong as possible) or else every language in BPP has a subexponential time deterministic algorithm in an average-case sense. The latter means that it is infeasible to come up with inputs on which the algorithm fails. We now formalize this notion in more detail.

Following [20] we say that two languages L_1, L_2 are $\mathcal{C}$-different if there exists an algorithm $REF \in \mathcal{C}$, called a *refuter*, that produces instances in the symmetric difference of L_1 and L_2. $\mathcal{C}$ may be an arbitrary uniform class, and can be, e.g., a deterministic, probabilistic or a non-deterministic class. For two complexity classes $\mathcal{C}, \mathcal{C}'$ the complexity class [io-pseudo($\mathcal{C}'$)] $\mathcal{C}$ denotes all languages which are $\mathcal{C}'$-similar (i.e., not $\mathcal{C}'$-different) to some language in $\mathcal{C}$ [2]. Notice that the stronger the refuter is, the smaller the pseudo-class is. Precise definitions, as well as some other related notions are given in Section 2. With this terminology, the result of Impagliazzo and Wigderson [19] is: if BPP $\neq$ EXP then BPP $\subseteq$ [io-pseudo(BPP)] SUBEXP [3].

1.1 Our results

Our goal in this paper is to prove an analogue of the [19] result for AM. While [19] works only for the low-end setting (see [34]), our result works only for the high-end setting. We prove:

Theorem 1 *If $E \not\subseteq AMTIME(2^{\beta n})$ for some constant β, then for every $c \geq 1$, $AM \subseteq$ [io-pseudo($NTIME(n^c)$)] NP.*

Precise definitions appear in Section 2. In words our result can be stated as follows: Either Arthur-Merlin protocols are very strong and everything in E can be proved to a sub-exponential time verifier. Or else Arthur-Merlin protocols are very weak and Merlin can prove nothing that cannot be proven in the uniform average-case setting by standard NP proofs.

Our second result concerns the class AM $\cap$ *co*AM which we believe is of special interest as it contains the well studied class SZK (Statistical Zero-Knowledge) [1], and therefore contains some very natural problems that are not known to be in NP, e.g., Graph non-isomorphism [12], approximations of shortest and closest vectors in a lattice [11] and Statistical Difference [29]. For AM $\cap$ *co*AM we can completely get rid of the [io-pseudo] quantifier and show:

Theorem 2

1. *If $E \not\subseteq AMTIME(2^{\beta n})$ for some constant β, then $AM \cap coAM \subseteq$ [io] $NP \cap$ [io] $coNP$.*
2. *If $E \not\subseteq$ [io] $AMTIME(2^{\beta n})$ for some constant β, then $AM \cap coAM = NP \cap coNP$.*

We mention that the fact that we have both *infinitely often* and *almost everywhere* versions for the case of AM$\cap$*co*AM is not trivial and indeed some technical work is needed for showing that. The Impagliazzo Wigderson construction [19] as well as our Theorem 1 do not have both versions.

1.2 Comparison to previous work .

- Nonuniform hardness vs. randomness tradeoffs were given in [23, 25, 32]. Miltersen and Vinodchandran [25] prove that if $NE \cap coNE \not\subseteq$ NTIME($2^{\beta n}$)/$2^{\beta n}$ for some constant β then AM = NP. This high-end result was extended in [32, 35] to the low-end setting.
- Using the easy witness method of Kabanets, [20] Lu [24] showed a derandomization of AM under a uniform assumption. Specifically, he showed that if NP $\not\subseteq$ [io-pseudo(NP)] DTIME($2^{n^{\epsilon}}$) for some $\epsilon > 0$ then AM = NP. Impagliazzo, Kabanets and Wigderson [15] were able to remove the [io-pseudo] quantifier from [24] and obtain the same conclusion using an assumption on exponential classes,

[1] We remark that in some cases these nonuniform assumptions follow from assumptions that involve only uniform classes. For example, [5] prove a stronger version of the Karp-Lipton theorem [22] and show that the uniform assumption EXP $\neq$ MA implies the non-uniform assumption EXP $\not\subseteq$ P/*poly*, which in turn can be used to derandomize BPP and MA.

[2] Here, as in [20] the "io" stands for infinitely often. One can also define "almost everywhere" versions of "pseudo". We remark that if the refuters have unlimited computational power, then this definition coincides with the standard notion of [io] $\mathcal{C}$ which involves all languages that are infinitely often equal to some language in $\mathcal{C}$.

[3] [19] use different notations and their statement is actually slightly stronger than the one we give here. The average-case notion of [19] is close to the standard notion of average-case complexity. We use the terminology of [20] which extends this notion. This extension drifts away from the standard notion of average-case complexity. However, we still use the term average-case for historical reasons. See remark 9 for more details.

namely, if NE$\cap co$NE $\not\subseteq$ [io] DTIME($2^{2^{\epsilon n}}$) for some $\epsilon > 0$, then AM $=$ NP.

Our result is a uniform version of the non-uniform result of Miltersen and Vinodchandran [25] and uses their construction. In Section 1.3 we explain our technique and the technical difficulty in this transition. The results of [24] and [15] are incomparable to ours and use different techniques. We would like to stress that our results give an analogue for AM of the derandomization result of Impagliazzo and Wigderson [19], while previous results do not.

Previous results do not give better results for AM $\cap$ *co*AM than for AM. Our Theorem 2 manages to remove the [io-pseudo] quantifier for AM $\cap$ *co*AM by exploiting the trivial fact that AM$\cap co$AM is closed under complement. We are not aware of a previous result in this context which exploits properties of the class being derandomized [4].

1.3 The technique

In this section we attempt to explain our technique on an "intuitive level". It is helpful to be imprecise with respect to "infinitely often". The presentation is in some cases oversimplified and the reader is referred to the technical sections for precise details.

The case of MA. Let us first start with the MA analogue of Theorem 1. The analogue is correct both in the high-end and the low-end setting, and for notational simplicity we discuss the low-end result, i.e., that EXP $\neq$ MA implies MA $\subseteq$ NSUBEXP. The proof follows from [6, 3] and relies on two claims. One is that if MA $\not\subseteq$ NSUBEXP then some generator fails and EXP $\subseteq$ P/*Poly*. The second is that the non-uniform assertion EXP $\subseteq$ P/*Poly* implies the uniform assertion EXP $=$ MA [5].

We now present this second argument in the terminology of *instance checkers* [8]. An instance checker is a probabilistic polynomial-time oracle machine that when given oracle access to f and input x outputs $f(x)$, and when given oracle access to any $f' \neq f$ either outputs $f(x)$ or rejects with probability almost one. By [5], every EXP-complete language f has an instance checker. Let f be an EXP-complete function and assume EXP $\subseteq$ P/*Poly*. To show that $f \in$ MA consider the following proof system: On input x, Merlin sends Arthur a polynomial size circuit for $f \cap \{0,1\}^{|x|}$. Arthur simulates the instance checker on the given input x, using the circuit as the oracle. It is easy to check that the protocol is sound and complete. The crucial fact in the proof is so simple that it can be missed: by sending the circuit Merlin *commits* himself to a specific function.

Trying to use this argument for AM. We would like to apply a similar argument for AM. While for derandomizing MA it is sufficient to get hold of a uniformly computable function that is hard against deterministic Boolean circuits, for AM, known results rely on a function that is hard against nondeterministic *single-valued* circuits [25, 32]. Informally, a single valued circuit is a non-deterministic circuit in which each computation path can take one value from $\{0, 1, quit\}$ such that all non-quitting paths take the *same* value (which we define to be the "output" of the circuit).

We would like to show that if EXP-complete functions have small nondeterministic single-valued circuits then EXP $=$ AM. Let us examine the approach of [5]. Let f be an EXP-complete function. We ask Merlin to send a nondeterministic single-valued circuit that computes f. We would like Arthur to run the instance checker using the circuit as the oracle. As the circuit is nondeterministic, Arthur cannot "run" the circuit by himself. Therefore, each time Arthur wants to compute the function on some input, he asks Merlin to provide him with a non-quitting computation path for that input, and to save on the number of rounds, he sends all his random bits together. This argument fails because not every nondeterministic circuit is necessarily single-valued. A dishonest prover may supply a circuit that is not single valued. Consider a circuit that for every input has a non-quitting path that evaluates to 1 and another that evaluates to 0. The prover can evaluate the circuit to any value he wishes, and is not committed to any specific function. Indeed, we do not know how to prove that if EXP-complete functions have small nondeterministic single-valued circuits then EXP $=$ AM.

Reconstruction algorithms. Following [19] (and later [34]) we use the fact that the proofs of nonuniform Hardness vs. Randomness results give *reconstruction algorithms*. Let f be a hard function on which the generator is based. If the generator is not pseudorandom then there exists a circuit D that distinguishes the "pseudo-random bits" from uniformly distributed bits. A reconstruction algorithm R gets the circuit D and a short "advice string" a (that may depend on f and D) and outputs a small circuit $C = R(D, a)$ that computes the function f. Proofs of hardness vs. randomness results work by specifying such algorithms. Indeed, these algorithms establish the correctness of

[4] Usually, the class being derandomized is defined in terms of "feasible algorithms" (or protocols). Additional properties of these class are intuitively "given up" when these algorithms are converted into circuits.

the generator as it follows that if f is hard then the generator is pseudo-random. The key idea of [19] is to exploit specific properties of a particular reconstruction algorithm of [28, 6]. For starters, if R is efficiently computable, then the uniform algorithm (or in our case AM protocol) can run it. We use this idea in a different way and rely on different properties of a particular reconstruction algorithm. This property (called *resiliency*) is explained in detail below.

For derandomizing BPP (or MA), D and C are deterministic circuits. In the case of AM, the generators and reconstruction algorithms that are given in [25, 32] have D as a co-nondeterministic circuit and C as a nondeterministic *single-valued* circuit for f.

Our solution. We get our results for AM by integrating reconstruction algorithms with additional properties into the argument of [5]. Let f be an EXP-complete function. We start with a "non-uniform" pseudo-random generator for AM, that has a reconstruction algorithm R, and is based on the hard function f. We need to show that if this generator fails to derandomize AM then f has an AM protocol. If the generator fails we ask Merlin to send a distinguisher circuit D and the advice a which allows Arthur to construct $C = R(D, a)$ on his own. If Merlin is honest all goes fine just as before. If Merlin is dishonest, he can cheat by either sending a co-nondeterministic circuit D that is not distinguishing or by sending a wrong advice a. The first problem is removed by switching to the uniform average-case setting. Arthur can now run the refuter to generate D himself. (We remark that this is the only place where we exploit the uniform average case setting). We are left with handling incorrect advice strings a. For this purpose we suggest the notion of *resilient reconstruction algorithms.*

Resilient reconstructions. The new approach suggested in this paper is to study the behavior of reconstruction algorithms when given an "incorrect" advice string a. We cannot hope that the reconstruction algorithm R outputs a circuit C that computes f when given a wrong advice a. We can, however, hope that the circuit C is a nondeterministic *single valued* circuit for some function f'. We say that a reconstruction algorithm R is *resilient*, if it outputs a nondeterministic single-valued circuit given *any* (possibly wrong) advice a.

If R is a resilient reconstruction algorithm then even if Merlin is dishonest and sends an incorrect string a, the constructed circuit $C = R(D, a)$ is single-valued. Thus, even a dishonest Merlin has to commit himself to some function f' (the one defined by the single-valued circuit C). With that we can continue with the argument of [5], and Arthur can use the instance checker to validate his result.

We remark that [19] use a different approach: They use additional properties of the hard function f (e.g., downwards self reducibility and random self reducibility) as well as a specific "learning" property of the reconstruction algorithm of [6], to *efficiently find* the "correct" advice string a and reconstruct the circuit C. In contrast, we get the advice string a from Merlin and have to deal with the behavior of the reconstruction algorithm R on incorrect advice strings.

The Miltersen-Vinodchandran generator. In order to complete the argument, we have to show a construction of a generator that has a resilient reconstruction algorithm. We show that the Miltersen-Vinodchandran hitting-set generator [25] has such a *probabilistic* reconstruction algorithm. A weakness of the MV generator is that it only works for the high-end, and this is why we only get a high-end result.

In the argument above, we needed the resiliency of R to mean the following: if D is a distinguisher then for every a, $R(D, a)$ is single-valued. Interestingly, the reconstruction algorithm for the MV-generator is even more resilient than that and, in fact, whenever D rejects few inputs (regardless of being a distinguisher or not), for every a, $R(D, a)$ is single-valued. This added resiliency is the basis for our improved result for AM $\cap$ *co*AM. The closure property of this class allows Merlin to prove that D rejects only few inputs, which in turn guarantees that C is single-valued. Thus, Arthur no longer need the refuter, and we get rid of the refuter and the average-case clause.

2 Preliminaries

For a set $S \subseteq \{0,1\}^n$, the density of S, denoted $\rho(S)$, is $\rho(S) \stackrel{def}{=} \frac{|S|}{2^n}$. For a circuit D over Boolean inputs of length m we define $\rho(D) = \rho(\{y \in \{0,1\}^m \mid D(y) = 1\})$. For a language L and an input x, $L(x)$ is one if the input is in the language and zero otherwise. For a language L and an input length n, we define $L_n = L \cap \{0,1\}^n$. In this notation $\rho(L_n) = \Pr_{x \in \{0,1\}^n}[L_n(x) = 1]$.

For a class of languages C we define the class

$$[io]C = \{L : \exists M \in C \text{ s.t. for infinitely many } n,\, L_n = M_n\}$$

The notation $z \leftarrow U_m$ denotes picking z uniformly at random from $\{0,1\}^m$.

2.1 Complexity Classes

We denote EXP $=$ DTIME$(2^{poly(n)})$, E $=$ DTIME$(2^{O(n)})$ and SUBEXP $= \bigcap_{\epsilon>0}$ DTIME$(2^{n^{\epsilon}})$. We let NEXP, NE and NSUBEXP be their nondeterministic analogs respectively. Next, we define classes of languages that have Arthur-Merlin games (or protocols).

Definition 3 *(AM) A language L belongs to $AM_{\epsilon(n)}(TIME = t(n), COINS = m(n))$ if there exists a deterministic machine $M(x;y,z)$ such that for every $x \in \{0,1\}^n$:*

$$\begin{aligned} x \in L &\implies \forall_{y\in\{0,1\}^{m(n)}} \exists_z M(x;y,z) \\ x \notin L &\implies \Pr_{y\in\{0,1\}^{m(n)}}[\exists_z M(x;y,z)] \leq \epsilon(n) \end{aligned}$$

the machine uses at most $m(n)$ random coins and runs in time at most $t(n)$. If we are not interested in the number of coins we abbreviate the notation to $AMTIME_{\epsilon(n)}(t(n))$. If the ϵ is omitted then its default value is $1/2$. The class AM denotes the class $\bigcup_{c>0} AMTIME_{\frac{1}{2}}(n^c)$.

Note that when $x \in L$, with probability one (over the y's) there exists z such that $M(x;y,z)$. That is, Arthur-Merlin protocols are defined to have one-sided error. The original definition of [7] has two-sided error, but it was shown in [10] that this is equivalent to the one-sided version. By the results of [7] and [14], a language has a constant-round interactive proof (see [13] for the definition) of complexity $t(n)$, if and only if it is in AMTIME$(poly(t(n)))$.

Finally, following [25], we need the AM protocols to have extremely small error probability (not only small with respect to the input length, but small with respect to the number of random coins). The following Lemma shows that we can reduce the error probability to the required level. The proof uses explicit constructions of a primitive called a *disperser*, see surveys [27, 26, 31].

Lemma 4 *(Implicit in [25]) There exists some constant $\Delta > 1$, such that for every $0 < \delta < 1$, $AMTIME_{1/2}(n^c) \subseteq AM_{2^{-m+m^{\delta}}}(TIME = m^{2\Delta}, COINS = m = O(n^{c/\delta}))$.*

2.2 Nondeterministic circuits

Definition 5 *A nondeterministic Boolean circuit $C(x,y)$ gets x as an input and y as a witness. We say that $C(x) = 1$, if there exists a witness y such that $C(x,y) = 1$. Otherwise $C(x) = 0$. A co-nondeterministic circuit is defined similarly, where $C(x) = 0$ if there exists a witness y such that $C(x,y) = 0$. Otherwise $C(x) = 1$.*

Definition 6 *A nondeterministic SV (single-valued) circuit is a circuit as above with three possible outputs: 1,0 and* quit *and such that for every input $x \in \{0,1\}^n$, either $\forall_y C(x,y) \in \{0, quit\}$ or $\forall_y C(x,y) \in \{1, quit\}$, i.e., all non-quit paths are consistent. We say that $C(x) = 1$ if there exists at least one path outputting 1, $C(x) = 0$ if there exists at least one path outputting 0, and $C(x) = quit$ otherwise. We say that C is a nondeterministic TSV (*total *single-valued) circuit, if for all $x \in \{0,1\}^n$, $C(x) \neq quit$, i.e., C defines a total function on $\{0,1\}^n$. Otherwise, we say that it is a nondeterministic PSV (*partial *single-valued) circuit.*

It is easy to see that a Boolean function f, has a nondeterministic TSV circuit of size $O(s(n))$, if and only if f has both a nondeterministic and a co-nondeterministic circuits of size $O(s(n))$.

2.3 Generators

A generator is a function $G : \{0,1\}^k \to \{0,1\}^m$ for $m > k$. We think of G as "stretching" k bits into m bits. We say a generator G is:

- ϵ-pseudo-random for a class $\mathcal{A}$, if for every function $h : \{0,1\}^m \to \{0,1\}$ in $\mathcal{A}$, it holds that $|\Pr_{y\leftarrow U_k}[h(G(y)) = 1] - \Pr_{z\leftarrow U_m}[h(z) = 1]| < \epsilon$.
- ϵ-hitting for $\mathcal{A}$ if for every function $h : \{0,1\}^m \to \{0,1\}$ in $\mathcal{A}$ such that $\Pr_{z\leftarrow U_m}[h(z) = 1] > \epsilon$ there exists a $y \in \{0,1\}^k$ such that $h(G(y)) = 1$.

Note that every ϵ-pseudo-random generator is also ϵ-hitting.

We will be interested in $\mathcal{A}$'s such as functions computed by deterministic circuits of some size $t(m)$, nondeterministic circuits, co-nondeterministic circuits, etc. When G is not pseudo-random (hitting) for $\mathcal{A}$ we call a function $h \in \mathcal{A}$ that violates the condition above an ϵ-distinguisher for G.

2.4 Pseudo classes

In this section we define the notion of *uniform* indistinguishability in the average-case sense. For the purposes of this paper, we define only indistinguishability with respect to nondeterministic observers. Indistinguishability with respect to other observers (e.g. deterministic, probabilistic) can be similarly defined. The following definitions and notations are adopted from [20].

Definition 7 *We say that two languages $L, M \subseteq \{0,1\}^*$ are $NTIME(t(n))$-distinguishable a.e. (almost* *everywhere), if there exists a nondeterministic length-preserving procedure REF (which we call a* refuter*), that runs in time $t(n)$, such that for all but finitely many n's, R on input 1^n, has at least one accepting computation path, and on every accepting path it outputs an instance x, such that $x \in L \triangle M$ (where $L \triangle M$ is the symmetric difference between L and M). If this holds only for infinitely many n's, we say that L and M are $NTIME(t(n))$-distinguishable i.o. (infinitely often).*

If L and M are not $\mathrm{NTIME}(t(n))$-distinguishable a.e. (resp. i.o.), we say that they are $\mathrm{NTIME}(t(n))$-indistinguishable i.o. (resp. a.e.).

Definition 8 *Given a complexity class $\mathcal{C}$ of languages over $\{0,1\}^*$ we define the complexity classes:*
$[pseudo(NTIME(t(n)))]\ \mathcal{C} = \{L : \exists M \in \mathcal{C}$ s.t. L and M are $NTIME(t(n)) -$ indistinguishable a.e.$\}$
$[io\text{-}pseudo(NTIME(t(n)))]\ \mathcal{C} = \{L : \exists M \in \mathcal{C}$ s.t. L and M are $NTIME(t(n)) -$ indistinguishable i.o.$\}$

Remark 9 *In contrast to [19, 20] the refuters we use are nondeterministic. Thus, the notion of uniform average case setting used in this paper is stronger than that of [20] (which discusses probabilistic zero-error refuters) and not precisely comparable to that of [19] which (discusses probabilistic two-sided error refuters). We remark that the notion of [19] seems strong as the refuter does not have to know whether it produced a "correct" counterexample. A refuter which chooses its output at random is valid even if it only supplies a counterexample with probability $1/n$. Note that standard amplification is not applicable in this case as there's no way to check if a given input is a counterexample. Nevertheless, in the setup of [19] it is possible to amplify this success probability and get a refuter which outputs a counterexample with probability close to 1.*

2.5 Instance checking

Blum and Kannan [8] introduced the notion of instance checkers. We give a slight variation on their definition.

Definition 10 *An instance checker for a language L, is a probabilistic polynomial-time oracle machine IC, whose output is $0, 1$ or fail and such that,*

- *For every input x, $Pr[IC^L(x) = L(x)] = 1$.*
- *For every input $x \in \{0,1\}^\ell$, and every oracle L', $Pr[IC^{L'}(x) \notin \{L(x), fail\}] \leq 2^{-\ell}$.*

It follows from [5, 2] that:

Theorem 11 *There exists a complete problem (under linear-time reductions) in E that has an instance checker that makes queries of length exactly $c\ell$ on inputs of length ℓ, for some constant c.*

From this we can deduce that:

Corollary 12 *Every problem in E that is complete under linear-time reductions, has an instance checker that makes queries of length exactly $c\ell$ on inputs of length ℓ, for some constant c.*

3 Uniform hardness vs. randomness for AM

In this section we prove Theorem 1. We start with the description of the Miltersen-Vinodchandran hitting set generator on which our proof is heavily based.

3.1 The Miltersen-Vinodchandran generator

Let $f : \{0,1\}^\ell \to \{0,1\}$. We look at f as a d-variate polynomial $f : H^d \to \{0,1\}$ where $|H| = h = 2^{\ell/d}$. For a field F with $q \geq 2h$ elements and $H \subseteq F$, let $\hat{f}$ be the low degree extension of f [4]. That is, we think of f as a function $f : H^d \to \{0,1\}$ and $\hat{f}$ is the unique multivariate polynomial $\hat{f} : F^d \to F$ that extends f and has degree at most $h-1$ in each variable. We let $e_i = (0,\ldots,0,1,0,\ldots,0)$ be the i'th basis vector of F^d, with one in the i'th coordinate and zeros everywhere else. For $w \in F^d$, the points $\{w + ae_i \mid a \in F\}$ lie on an axis-parallel line that passes through w with direction i. The restriction of the multi-variate polynomial $\hat{f}$ to that line is a univariate polynomial of degree at most $h-1$, and we denote it by $\hat{f}|_{w+Fe_i}$. The generator $\mathrm{MV}_f = \mathrm{MV}_{f,\ell,m,d,h,q} : [d] \times F^d \to F^h$ is defined as:

$$\mathrm{MV}_{f,\ell,m,d,h,q}(i,w) \quad = \quad \hat{f}|_{w+Fe_i}$$

We often omit some (or all) of the subscripts. We see that $\mathrm{MV}_f : \{0,1\}^k \to \{0,1\}^m$, with

- $k = \log(d) + \log(q^d)$,
- $m = h\log(q)$ (since $2^m = q^h$), and,
- if $f \in \mathrm{DTIME}(2^{O(\ell)})$, then $\mathrm{MV}_{f,\ell,d} \in \mathrm{DTIME}(2^{O(\ell)})$.

From now on we choose $q = 2h$. Under this choice $k \leq \log(d) + d(\frac{\ell}{d} + 1) < 2\ell$ and $m \leq h^2$. Miltersen and Vinodchandran prove:

Theorem 13 ([25]) *For every $\delta > 0$, $c > 0$, and $\ell > \frac{1}{\delta^2}$, set $d = \frac{1}{\delta}$. Let $f : \{0,1\}^\ell \to \{0,1\}$. If f requires non-deterministic TSV circuits of size $\Omega(2^{4(c+2)\delta\ell})$, then $MV_{f,d} : \{0,1\}^k \to \{0,1\}^m$ is ϵ-hitting for co-nondeterministic circuits of size m^c, where $k = 2\ell$, $m = \Omega(2^{\delta\ell})$, and $\epsilon = 1 - 2^{m^\delta - m}$.*

The statement of Theorem 13 involves many parameters. However, the precise parameters can be ignored by the reader. Intuitively, the theorem says that if f is sufficiently hard (hard for TSV circuits of size $2^{\Omega(\ell)}$) then the MV generator stretches $O(\ell)$ bits into $2^{\Omega(\ell)}$ bits and fools co-nondeterministic circuits of size $2^{\Omega(\ell)}$. This is the way parameters are chosen in high-end results. When using the generator to derandomize, we choose ℓ to be logarithmic in the length of the input of the AM problem and terms of the form $2^{\Omega(\ell)}$ become polynomials in the length of this input.

The emergence of co-nondeterministic circuits, and the large error of the generator MV_f might look bewildering to the reader. However, these details are of purely technical nature, and naturally arise in the process of derandomization of AM. To see that, consider an AM language L, as in Definition 3, that is solvable with a predicate $M(x;y,z)$. If $x \notin L$, then for almost all y, there is no z such that $M(x;y,z)$. Equivalently, for $1-\epsilon$ of the y we have that $\forall_z M(x,y;z) = 0$. Now, in the derandomization process we try all y in the generator's image. In particular, we hope that for at least one y in the generator's image, $\forall_z M(x,y;z) = 0$. We therefore see that what we need is a hitting set generator against co-nondeterministic circuits, and we can allow very large errors.

The novelty of this hardness vs. randomness result lies in the fact that the hardness assumption is against a class of total, single-valued circuits that seems weaker than the class fooled by the generator, i.e., co-nondeterministic circuits.[5] It also turns out that such hitting generators suffice to completely derandomize the one-sided error class AM, that is, to prove that AM = NP.

As with other hardness vs. randomness tradeoffs, the proof of Theorem 13 works by showing a reconstruction algorithm that from a co-nondeterministic distinguisher for MV_f generates a small TSV circuit for f. We show this algorithm (as well as some useful properties it has) in Section 3.3.

[5] It should be noted however that although single-valued circuits seem weaker than nondeterministic or co-nondeterministic circuits, the assumption that there is a function in EXP (or E) that is hard for single-valued circuits implies the assumption that there is a function that is hard for nondeterministic (or co-nondeterministic) circuit). This is because EXP is closed under complement. Previous results by [23] relied on hardness for circuits with NP oracle.

3.2 Resilient reconstructions

Typically, the correctness of pseudo-random generators (and hitting-set generators) is proved by showing a reconstruction algorithm that from a distinguisher for the generator and some additional advice, generates a small circuit that computes the (supposedly hard) function on which the generator is based. Let us start by defining the notion of reconstruction algorithms for generators that fools nondeterministic (or co-nondeterministic) circuits. In the following two definitions, G_f is a PRG (or a HSG), that is based on the hardness of the function f, and supposedly fools co-nondeterministic circuits.

Definition 14 *A deterministic machine $R(\cdot,\cdot,\cdot)$ is called a $(t,\epsilon) \to T$ reconstruction algorithm for G_f with success probability p, if*

- *For every function $f : \{0,1\}^\ell \to \{0,1\}$, and every size t co-nondeterministic circuit D that is an ϵ-distinguisher for G_f,*

with probability at least p over r there exists a such that,

- *$R(D,a,r)$ outputs a nondeterministic TSV circuit C of size at most $T = T(t,\epsilon,\ell)$ that computes f.*

We also say in short that R is a $(t,\epsilon) \to_p T$ reconstruction algorithm. A reconstruction algorithm R is called efficient *if it runs in time polynomial in its output length T.*

We are interested in the behavior of the reconstruction algorithm R when given an "incorrect" advice, i.e, when D isn't a distinguisher or when a isn't the correct string. Clearly, we can not expect R to output the correct circuit given an incorrect advice. We can, however, hope that R outputs an SV circuit even when given an incorrect (or malicious) advice. We call such a reconstruction algorithm *resilient*. The formal definition follows.

Definition 15 *A reconstruction algorithm $R(D,a;r)$ is p-*resilient *against a co-nondeterministic circuit D if $\Pr_r[\forall a\ R(D,a;r)$ is SV$] \geq p$. The reconstruction algorithm is p-resilient against a family K of co-nondeterministic circuits if it is p-resilient against any circuit $D \in K$.*

3.3 A resilient reconstruction algorithm for MV_f

Our main observation is that the reconstruction algorithm for MV_f that is given in [25] (with slightly different parameters) is resilient against the family of all co-nondeterministic circuits that almost always accept. (That is, almost always answer one). More precisely, let K_ϵ be the family of all co-nondeterministic circuits D with $\rho(D) > \epsilon$, then,

Lemma 16 *Let $\delta > 0$ and $\ell \in N$ such that $\ell > \frac{1}{\delta^2}$. Set $d = \frac{1}{\delta}$. There exists an efficient $(t, \epsilon) \to_p T$ reconstruction algorithm R for $MV_{f,\ell,d}$, that is p-resilient against K_ϵ, where $\epsilon = 1 - 2^{m^\delta - m}$, $T = O(2^{12\delta\ell}t^2)$ and $p = 1 - 2^{m^\delta}$.*

Proof: We basically repeat the Miltersen-Vinodchandran proof that a reconstruction algorithm exists, and we note that the reconstruction is resilient. Suppose D is a co-nondeterministic circuit that is an ϵ–distinguisher for $\mathrm{MV}_{f,\ell,d}$. That is, if we denote

$$\begin{aligned} G &= \mathrm{IMAGE}(\mathrm{MV}_f) = \{v \in \{0,1\}^m \,|\exists_{i,w} \mathrm{MV}_f(i,w) = v\} \\ Z &= \mathrm{ZEROS}(D) = \{y \in \{0,1\}^m \,| D(y) = 0\} \end{aligned}$$

then $G \subseteq Z$ (because the generator does not hit any string v that D accepts) and $|Z| < (1-\epsilon)2^m = 2^{m^\delta - m}$. $2^m = 2^{m^\delta}$ (because $\rho(D) \geq \epsilon$). Denote, $\mathrm{SIZE}(D) = t(m)$.

We let F_{h-1} denote the set of all degree $h-1$ univariate polynomials over F. The generator MV_f outputs elements from F_{h-1}. We also associate $z \in Z$ with elements of F_{h-1} (and recall that $\{0,1\}^m = F^h$). For $q \in F_{h-1}$ and $S = \{x_1, \ldots, x_s\} \subseteq F$, let $q|_S$ be the restriction of q to the set S, i.e., the vector $(q(x_1), \ldots, q(x_s))$. We say that $S \subseteq F$ "splits" Z if for every $q_1 \neq q_2 \in Z$ it holds that $q_1|_S \neq q_2|_S$. The following claim says that a large enough randomly chosen S splits Z with high probability (in particular, $|S| \geq 3\log|Z|$ will do).

Claim 17 *For a uniformly chosen $S \subseteq F$,* $\Pr(S \textit{ does not split } Z) < |Z|^2 2^{-|S|}$.

Proof: Let $s = |S|$. Fix $q_1 \neq q_2 \in F_{h-1}$. As q_1 and q_2 are different univariate polynomials of degree at most $h-1$, the probability that q_1 and q_2 agree on s randomly chosen elements in the field is at most $(\frac{h-1}{q})^s \leq 2^{-s}$ since we chose q to be $2h$ (this probability is even smaller when S is chosen without repetitions). Taking the union bound over all pairs in Z, the probability that there exists such a bad pair q_1, q_2 is smaller than $\binom{|Z|}{2}2^{-s} < |Z|^2 2^{-s}$. □

We are now ready to describe the reconstruction algorithm $R(\cdot,\cdot,\cdot)$. The inputs to R are interpreted as:

- The distinguisher D given as a co-nondeterministic circuit.
- r is a uniformly chosen $S \subseteq F$, where $|S| = 3m^\delta \geq 3\log|Z|$.
- $a = \hat{f}(S^d)$, i.e., the value $\hat{f}(v)$ for every element $v \in S^d$.

R outputs a nondeterministic circuit C. We now describe the computation of C. The input to C is $y = (y_1, \ldots, y_d) \in F^d$, and its output is $\hat{f}(y_1, \ldots, y_d)$. C successively learns the values $\hat{f}(A_i)$ for $A_i = \{(y_1, \ldots, y_i, s_{i+1}, \ldots, s_d) \mid s_j \in S\}$. $A_0 = S^d$ and so we have $\hat{f}(A_0)$ as an input to R, and we can hardwire it into C. $A_d = \{(y_1, \ldots, y_d)\}$, so after d iterations C can output $\hat{f}(A_d) = \hat{f}(y_1, \ldots, y_d)$.

Say we already have the values $\hat{f}(A_i)$, we show how C computes $\hat{f}(A_{i+1})$. For every $s_{i+2}, \ldots, s_d \in S$, C does the following guesses:

- C guesses $q \in F_{h-1}$
- C guesses z such that $D(q; z)$ rejects. (Recall that C is co-nondeterministic).

We remark (and we will soon show) that at this point (1) the only non-rejecting paths are those that guessed q such that $D(q) = 0$ and a proof z for that.

The algorithm proceeds to check that for every $j \in S$, $q(j) = \hat{f}(y_1, \ldots, y_i, j, s_{i+2}, \ldots, s_d)$. Recall that for $j \in S$, $(y_1, \ldots, y_i, j, s_{i+2}, \ldots, s_d) \in A_i$ and so we already know $\hat{f}(y_1, \ldots, y_i, j, s_{i+2}, \ldots, s_d)$.

We remark (and we will soon show) that at this point (2), the only non-rejecting paths are those who guessed the polynomial $q(j) = \hat{f}(y_1, \ldots, y_i, j, s_{i+2}, \ldots, s_d)$. In particular, $\hat{f}(y_1, \ldots, y_i, y_{i+1}, s_{i+2}, \ldots, s_d) = q(y_{i+1})$. After doing that for every $s_{i+2}, \ldots, s_d \in S$ we know all the values in $\hat{f}(A_{i+1})$.

Lemma 18 *The above algorithm is a resilient reconstruction algorithm for MV_f with parameters as stated in the lemma.*

Proof:

Correctness : To see that R is a reconstruction algorithm with success probability p, we have to show that with this probability, when D is a distinguisher, our conclusions in stages (1) and (2) are correct in every iteration.

Since D is a distinguisher, it must hold that $\rho(D) > \epsilon$ and hence $|Z| = |ZEROS(D)| \leq$

$(1-\epsilon)2^m = 2^{m^\delta}$. Therefore, by Claim 17, with probability at least p, S splits Z. Suppose that S is indeed splitting.

If C guessed a polynomial q for which $D(q) = 1$, then for all z, $D(q;z)$ accepts and C rejects. If, on the other hand, C guessed a polynomial q for which $D(q) = 0$, then for some z, $D(q;z)$ rejects. Thus, at stage (1) the surviving paths are exactly those who guessed $q \in Z$ and a witness z for that.

Next, C checks that q and $q'(j) \stackrel{def}{=} \hat{f}(y_1,\ldots,y_i,j,s_{i+2},\ldots,s_d)$ agree on S. Notice that $q' \in G \subseteq Z$. However, as both q and q' are in Z, and both agree on S, it must be that $q = q'$ (because S splits Z). We therefore conclude that the correct path, guessing $q = q'$ survives, and furthermore, every surviving path guessed q'. It follows that every non-rejecting path computes the value $\hat{f}(A_{i+1})$ correctly. Hence, the algorithm is TSV.

Complexity : The algorithm of the circuit C makes d iterations. In each iteration C guesses at most $|S|^d$ polynomials (strings of length m) on which it evaluates the circuit D, and each evaluation takes at most $t(m) = \mathrm{SIZE}(D)$ time. Thus, the total running time is: $t(m) \cdot O(d \cdot |S|^d) = t(m) \cdot O(d(3m^\delta)^d) \leq t(m) \cdot O(d3^d m^{\delta d}) \leq t(m) \cdot O(2^{4d} m)$. However, $d = \frac{1}{\delta} \leq \sqrt{\ell} \leq \delta\ell$, and so $2^{4d} \leq 2^{4\delta\ell}$. We also have, $m \leq h^2 = 2^{2\ell/d}$. Altogether, the running time is at most $t(m) \cdot O(2^{6\delta\ell})$. The circuit size is at most the square of this.

Resiliency : Finally, we show that the reconstruction algorithm is p-resilient against K_ϵ. First note that Claim 17 is correct when $\rho(D) > \epsilon$, even if D is not a distinguisher. This is because $|Z| = |ZEROS(D)| \leq (1-\epsilon)2^m = 2^{m^\delta}$ and this is all that is needed in the proof of Claim 17. So S splits Z with probability at least p.

Now Suppose for contradiction that $D \in K_\epsilon$, $r = S$ splits Z, and there is some (incorrect) advice a and some input $y \in F^d$ to C, such that $C = R(D,a,r)$ has two different accepting paths on y that result in different values for $\hat{f}(y)$. It must hold then that at some iteration C chooses in stage (1) q_1 in the first path and q_2 in the other, and $q_1 \neq q_2$. Let us look at the first time this happens, and suppose it is during the computation of $val(A_i)$, and when the last values are fixed to some $s_{i+1},\ldots,s_d \in F$. Since the two paths are accepting, it follows that for every $j \in S$, $val(y_1,\ldots,y_{i-1},j,s_{i+1},\ldots,s_d) = q_1(j) = q_2(j)$ (note that $val(y_1,\ldots,y_{i-1},j,s_{i+1},\ldots,s_d)$ does not necessarily equal $\hat{f}(y_1,\ldots,y_{i-1},j,s_{i+1},\ldots,s_d)$ because the advice $val(A_0)$ may be incorrect and is not necessarily the restriction of $\hat{f}$ to S^d). However, as before, since S splits Z, and $q_1, q_2 \in Z$ agree on S, it must hold that $q_1 = q_2$ contradicting our assumption. Thus, whenever S splits Z, $R(D,a,r)$ is SV for every possible (correct or incorrect) advice a.

□ □

3.4 The proof of Theorem 1

We are now ready to prove Theorem 1.

Proof: Assume not. Then $\mathrm{E} \not\subseteq \mathrm{AMTIME}(2^{\beta n})$ for some $\beta > 0$, and $\mathrm{AM} \not\subseteq$ [io-pseudo(NTIME($n^{O(1)}$))] NP. Let $\delta > 0$ be a constant. We will choose δ later, and for the time being express other parameters in terms of δ.

Let f be the E-complete language from Theorem 11. Let IC be the instance checker for f, and suppose that IC makes queries of length exactly $c\ell$ on inputs of length ℓ, for some constant c. As f is E-complete under linear-time reductions, it must be that for some $\beta' > 0$, $f \notin \mathrm{AMTIME}(2^{\beta'\ell})$.

Claim 19 *If AM $\not\subseteq$ [io-pseudo(NTIME($n^{O(1)}$))] NP, then there is a polynomial-time nondeterministic procedure P, that for all but finitely many ℓ, on input 1^n, has at least one accepting computation path, and on every accepting path it outputs a co-nondeterministic circuit D_m, such that D_m is an ϵ–distinguisher for $MV_{f,\ell,m,d}$. Where $n = 2^{\theta(\delta^2\ell)}$, $m = n^{O(1/\delta)}$, $d = \frac{1}{\delta}$, and $\epsilon = 1 - 2^{m^\delta - m}$.*

Proof: By the hypothesis of the claim, It follows that there is a language $L \in \mathrm{AMTIME}_{1/2}(n^{O(1)})$ such that $L \notin$ [io-pseudo(NTIME($n^{O(1)}$))] NP. By Lemma 4, $L \in \mathrm{AM}_{2^{m^\delta - m}}(\mathrm{TIME} = m^{O(1)}, \mathrm{COINS} = m = O(n^{O(1/\delta)}))$. That is, there is a polynomial-time machine $M(x;y,z)$ such that:

$$x \in L_n \implies \Pr_{y \in \{0,1\}^m}[\exists_z \ M(x;y,z) = 1\ ] = 1$$

$$x \notin L_n \implies \Pr_{y \in \{0,1\}^m}[\exists_z \ M(x;y,z) = 1\ ] \leq 2^{m^\delta - m}$$

We define the co-nondeterministic machine $\bar{M}$ as follows:

$$\bar{M}(x;y) = \begin{cases} 0 & \exists_z M(x,y,z) = 1 \\ 1 & \forall_z M(x,y,z) = 0 \end{cases}$$

In this notation L is characterized as:

$$x \in L_n \implies \Pr_{y\in\{0,1\}^m}[\bar{M}(x;y)=1\,]=0$$
$$x \notin L_n \implies \Pr_{y\in\{0,1\}^m}[\bar{M}(x;y)=1\,]\geq 1-2^{m^\delta-m}$$

We now define a language L', that is the output of the AM protocol when the truly random coins $y \in \{0,1\}^m$ are replaced with all the possible outputs in $\{0,1\}^m$ of the $\mathrm{MV}_{f,\ell,d}$ generator. Note that the output length of $\mathrm{MV}_{f,\ell,d}$ is at least $h=2^{\delta\ell}\geq m$. Define L' to be:

$$L'(x) = \begin{cases} 1 & \forall_{y\in \mathrm{IMAGE}(\mathrm{MV}_{f,d})}\exists_z M(x;y,z)=1 \\ 0 & \text{Otherwise.}\end{cases}$$

In terms of $\bar{M}$, L' is defined as follows:

$$L'(x) = \begin{cases} 1 & \forall_{y\in \mathrm{IMAGE}(\mathrm{MV}_{f,d})}\bar{M}(x;y)=0 \\ 0 & \exists_{y\in \mathrm{IMAGE}(\mathrm{MV}_{f,d})}\bar{M}(x;y)=1\end{cases}$$

We analyze now the running time of L'. L' calls the non-deterministic machine M on $|\mathrm{IMAGE}(\mathrm{MV}_{f,d})|$ different inputs. We see that $|\mathrm{IMAGE}(\mathrm{MV}_{f,d})| \leq 2^{3\ell} \leq 2^{O(\log(m)/\delta)}$ and $m = n^{O(1/\delta)}$, so altogether $|\mathrm{IMAGE}(\mathrm{MV}_{f,d})| \leq 2^{O(\log(n)/\delta^2)}$. Also, since $f \in \mathrm{DTIME}(2^{O(\ell)})$, each element in $\mathrm{IMAGE}(\mathrm{MV}_{f,d})$ is generated in time $2^{O(\log(m)/\delta)} = 2^{O(\log(n)/\delta^2)}$. Each execution of M takes $poly(m) = poly(n^{1/\delta})$ non-deterministic time. Altogether, $L' \in \mathrm{NTIME}(n^{O(1/\delta^2)})$, and since δ is a constant, $L' \in \mathrm{NP}$.

On the other hand, we know that $L \notin$ [io-pseudo(NTIME($n^{O(1)}$))] NP. It therefore follows that there exists a refuter $REF \in \mathrm{NTIME}(n^{O(1)})$ and an input length K, such that for all $k \geq K$, $REF(1^k) \in L \triangle L'$. Furthermore, $L \subseteq L'$ and so $REF(1^k) \in L' \setminus L$. W.l.o.g. $\ell \geq K$. We can describe now the nondeterministic procedure P, that outputs distinguishers for the generator. On input 1^n, P runs (the nondeterministic procedure) $REF(1^n)$. Let x_n be the output of $REF(1^n)$ on an accepting path, and recall that there is at least one such path. P generates the co-nondeterministic circuit D_m, with inputs from $\{0,1\}^m$, obtained from the computation $\bar{M}(x_n;\cdot)$ (that is, we hardwire x_n into the circuit which now takes as input elements from $\{0,1\}^m$). We see that:

- $\rho(D_m) \geq 1-2^{m^\delta-m}$ (because $REF(1^n) \notin L$).
- for every $y \in \mathrm{IMAGE}(\mathrm{MV}_{f,\ell,d})$, $D_m(y)=0$ (because $REF(1^n) \in L'$).
- $SIZE(D_m) \leq m^{4\Delta}$, where Δ is the constant from Lemma 4 (because $M \in \mathrm{AM}_{\epsilon=2^{m^\delta-m}}(\mathrm{TIME} = m^{2\Delta}, \mathrm{COINS}=m)$, and the translation to a circuit is at most quadratic in size).

By definition, D_m is a $(1-2^{m^\delta-m})$–distinguisher for $\mathrm{MV}_{f,\ell,d}$. P runs REF on 1^n, this takes time $n^{O(1)}$, it then creates the circuit D_m whose size is polynomial in the running-time of the machine $\bar{M}$ on input of length n, which is also $n^{O(1)}$. □

From Claim 19 and Lemma 16 we can conclude that f has small *non-uniform* TSV circuits. However, to reach a contradiction we need to show that f has an efficient (uniform) Arthur-Merlin protocol. We prove that by redoing the proof of Theorem 13 in a uniform (interactive) way. Consider the following $\mathrm{AMTIME}(2^{\beta'\ell})$ protocol for f. The input to the protocol is $y \in \{0,1\}^\ell$, and the output is $f(y)$. Set $\ell' = c\ell$, $m = 2^{\delta\ell'}$ and $n = m^{\Theta(\delta)}$.

The protocol :

1. Merlin: Sends a co-nondeterministic circuit D_m, and an accepting computation path of $P(1^n)$ whose output is D_m (P is the procedure from Claim 19).
2. Arthur:
   - Verifies that indeed D_m is an output of an accepting path of $P(1^n)$.
   - Sends a randomly chosen set $S \subseteq F$ of size $3m^\delta$.
3. Merlin: Sends $\hat{f}(S^d)$.
4. Arthur: Sends a randomly chosen $r \in \{0,1\}^{\mathrm{poly}(\ell')}$.
5. Let C be the non-deterministic circuit $C = R(D_m, S, \hat{f}(S^d))$ (R is the reconstruction algorithm from Lemma 16).
   Merlin:
   - Runs $\mathrm{IC}^C(y,r)$. I.e., it runs the instance checker on the input y, with the random coins r and using the non-deterministic circuit C as an oracle.
   - Sends the queries and the answers of the oracle C during the execution of $\mathrm{IC}^C(y,r)$, and,
   - For each pair of query q and answer a, gives a witness w such that $C(q;w)=a$.
6. Arthur: Runs $\mathrm{IC}^C(y,r)$ verifying the queries and the answers (using the witnesses).

Output : $\mathrm{IC}^C(y,r)$.

Lemma 20 *If we take δ to be small enough then the protocol above is an $AMTIME_{1/2}(2^{\beta'\ell})$ protocol computing f.*

Proof:

Completeness: We show that the interaction with an honest prover will always compute the right value (with probability 1). By Claim 19, for all but finitely many n's, $P(1^n)$ outputs on accepting paths a co-nondeterministic circuit D_m which is a $(1-2^{m^\delta-m})$–distinguisher for $\mathrm{MV}_{f,\ell',d}$. Arthur can be sure that D_m is indeed such a distinguisher because Merlin shows him that it is an output of an accepting path. Merlin also sends the correct values of $\hat{f}$ and the right answers of the instance checker IC. So by Lemma 16, C computes $\hat{f}$ (and hence f) correctly on inputs of length ℓ' (here we don't even need S to be splitting since the prover is honest and will lead Arthur through the correct computation path of C). By Theorem 11, we conclude that IC^C computes f correctly (on inputs of length ℓ) with probability 1.

Soundness : Let the prover be arbitrary. As before, D_m is a $(1-2^{m^\delta-m})$–distinguisher for $\mathrm{MV}_{f,\ell',d}$. This implies that $\rho(D_m) \geq 1-2^{m^\delta-m}$ and hence $|\mathrm{ZEROS}(D_m)| \leq 2^{m^\delta}$. It follows from Lemma 16 that except for some small probability, say $\frac{1}{10}$, C is a PSV circuit, regardless of what the prover chooses to give as the value of $\hat{f}(S)$. Because the circuit is PSV, it defines a (partial) function $g : \{0,1\}^{\ell'} \to \{0,1\}$. We now run the instance checker of f over the input y with an oracle access to g. By Theorem 11 we get the right answer, or reject, with probability at least, say, $\frac{9}{10}$. Altogether the probability we get the wrong answer is at most $\frac{1}{5}$.

Running time:
- Steps 1 and 2: By Claim 19, the running time of $P(1^n)$ is $n^{O(1)} = m^{O(1)}$, Therefore the size of the circuit D_m is also $m^{O(1)}$.
- Steps 3 and 4: The size of Merlin's message $\hat{f}(S^d)$, is about $|S|^d = 3^d m^{\delta d} \leq 2^{2\delta\ell} m = 2^{O(\delta\ell)}$ (for large enough ℓ).
  Sending r takes $poly(\ell') = poly(\ell)$ bits.
- Steps 5 and 6: There are poly(ℓ) steps of the instance checker IC, and each such step may involve a computation of the circuit C. By Lemma 16, $|C| = 2^{O(\delta\ell')} \cdot |D_m|^2 = m^{O(\delta\ell')}$. As $\ell' = c\ell$ and c is some fixed constant independent of δ, the running time of IC (including the evaluations of C) is $2^{O(\delta\ell)}$.

We see that the total running time is $2^{O(\delta\ell)}$ with the constant in the O notation independent of δ. Taking δ small enough this becomes smaller than $2^{\beta'\ell}$.

□

From Lemma 20 we get the desired contradiction.

□

4 Derandomizing $\mathrm{AM} \cap co\mathrm{AM}$

The proof of Theorem 2 is almost the same as that of Theorem 1. The main difference is that now we ask Merlin to send an instance x on which the generator MV fails, and to prove that x is *not* in the language being derandomized. This can be done since the complement of that language is also in AM. Such instances give rise to circuits D_m with $\rho(D_m) > 1-2^{m^\delta-m}$. The punch line is that Lemma 16 says that the reconstruction algorithm R is resilient against such circuits. We now give the formal proofs.

4.1 Proof of Theorem 2 part 1

Proof: Assume not. Then both $\mathrm{E} \not\subseteq \mathrm{AMTIME}(2^{\beta\ell})$ for some $\beta < 1$, and $\mathrm{AM} \cap co\mathrm{AM} \not\subseteq [\mathrm{io}]\ \mathrm{NP} \cap [\mathrm{io}]\ co\mathrm{NP}$. Let $\delta > 0$ be a constant that will be determined later. Let f be as in the proof of Theorem 1.

$\mathrm{AM} \cap co\mathrm{AM} \not\subseteq [\mathrm{io}]\ \mathrm{NP} \cap [\mathrm{io}]\ co\mathrm{NP}$. It follows that $\mathrm{AM} \cap co\mathrm{AM} \not\subseteq [\mathrm{io}]\ \mathrm{NP}$ and therefore there is a language $L \in \mathrm{AMTIME}_{1/2}(n^{O(1)}) \cap co\mathrm{AMTIME}_{1/2}(n^{O(1)})$ such that $L \notin [\mathrm{io}]\ \mathrm{NP}$. By Lemma 4, $L \in \mathrm{AM}_{2^{m^\delta-m}}(m^{O(1)}, m = n^{O(1/\delta)}) \cap co\mathrm{AM}_{\frac{1}{10}}(n^{O(1)}, n^{O(1)})$ [6]. We define M, $\bar{M}$ and L' as in the proof of Claim 19.

As before, $L' \in \mathrm{NTIME}(n^{O(1/\delta^2)})$. However, $L \notin [\mathrm{io}]\ \mathrm{NP}$. It follows that for all input lengths, except finitely many, there exists an input $x_n \in L \triangle L' = L' \setminus L$. We let D_m be the co-nondeterministic circuit with inputs from $\{0,1\}^m$, obtained from the computation $\bar{M}(x_n; \cdot)$.

We look at the same protocol as in the proof of Theorem 1 except that instead of step 1, the prover sends $x_n \in \{0,1\}^n$ (x_n is supposed to be in $L' \setminus L$) and proves that $x_n \notin L$ using the AM protocol for $\bar{L}$. We now prove correctness:

Soundness : In the first step, the prover sends x_n. If $x_n \in L$, then the prover is caught cheating with

[6] Note that we need the strong amplification of the success probability (i.e. Lemma 4) only for the protocol for L but not for $\bar{L}$. For the latter we use the standard amplification technique of running many independent copies of the protocol in parallel and deciding by majority.

probability at least $\frac{9}{10}$ during the proof that $x_n \notin L$. Otherwise, $x_n \notin L$, which means that $\rho(D_m) \geq 1 - 2^{m^\delta - m}$ and $|\text{ZEROS}(D_m)| \leq 2^{m^\delta}$. Now the same proof as in Lemma 20 shows that the output of the protocol is correct with probability at least $\frac{4}{5}$. So altogether the protocol computes the correct value or rejects with probability at least $\frac{7}{10}$.

Completeness : For every n (except finitely many) the prover is able to choose an element $x_n \in L' \setminus L$ and the proof is identical to the proof of Lemma 20.

Running time : The running time of the protocol is essentially the same as in the proof of Lemma 20. The only difference is that we replace the procedure P with the AM protocol for $\bar{L}$. The running time of this protocol is $m^{O(1)}$. The rest of the time analysis, as well as the choice of δ, is as before.

□

4.2 Proof of Theorem 2 part 2

Proof:

Assume not.

- Let f be the language:

  $\{(M, x, c) : M$ is a (padded) description of a deterministic machine that accepts x in time at most $c\}$.

  By "padded description" we mean that M is a string with a (possibly empty) prefix of zeros, followed by a description of a machine that starts with the bit 1. It is easy to verify that this language is complete in E, and hence by Corollary 12 it has an instance checker that makes queries of length exactly $c\ell$ on inputs of length ℓ, for some constant c. A property of this language that will be useful for us is that it is easy to embed inputs of length n as inputs of length $n+1$ (and so the truth table for length $n+1$ contains the truth table of length n in it) by padding M.

  Claim 21 *The hypothesis of the theorem implies that $f \notin$ [io] AMTIME$(2^{\beta\ell})$ for some $\beta < 1$.*

  We prove Claim 21 later.

- AM $\cap$ *co*AM $\not\subseteq$ NP, so there is a language $L \in \text{AMTIME}_{1/2}(n^{O(1)}) \cap co\text{AMTIME}_{1/2}(n^{O(1)})$ such that $L \notin$ NP. By Lemma 4, $L \in \text{AM}_{2^{m^\delta - m}}(m^{O(1)}, m = O(n^{O(1/\delta)})) \cap co\text{AM}_{\frac{1}{10}}(n^{O(1)}, n^{O(1)})$.

We define $L, \bar{M}$ and L' as in the proof of Theorem 2 (1).

As before, $L' \in$ NP. However, $L \notin$ NP. It follows that there is an infinite set I of input lengths, such that for every $n \in I$ there exists an input $x_n \in L \triangle L' = L' \setminus L$. We let D_m be the co-nondeterministic circuit with input $y \in \{0,1\}^m$, obtained from the computation $\bar{M}(x_n; \cdot)$.

We look at the same protocol as in the proof of Theorem 2 (1) except that in step 1, the prover first chooses an input length n between $2^{\delta^2 c\ell}$ and $2^{\delta^2 c(\ell+1)}$, then an element $x_n \in \{0,1\}^n$ that is supposed to be in $L' \setminus L$. The prover proves that $x_n \notin L$ using the AM protocol for $\bar{L}$. We now prove correctness:

Soundness : As before, if $x_n \in L$, then the prover is caught cheating with probability at least $\frac{9}{10}$ during the proof that $x_n \notin L$. Otherwise, $x_n \notin L$. Now the proof continues as in Theorem 2 (1).

Completeness : There are infinitely many n's such that there exist $x_n \in \{0,1\}^n$ for which $x_n \in L' \setminus L$. For such n, let $\ell = \lfloor \frac{\log(n)}{\delta^2 c} \rfloor$. So, $2^{\delta^2 c\ell} \leq n \leq 2^{\delta^2 c(\ell+1)}$. As in the proof of Theorem 1, from x_n we can obtain a distinguisher for $MV_{f,\ell,d}$ and from there by the reconstruction algorithm, a TSV circuit that computes correctly the function f on inputs of length ℓ (we remind the reader that the dependence between $n = n(\ell)$ and ℓ is about $n = m^\delta = 2^{\delta^2 c\ell}$). So for every such ℓ, the prover is able to choose an element $x_n \in L' \setminus L$, and when he does so the protocol runs smoothly with the same proof as in Theorem 2 (1).

Running time : The running time of the protocol is essentially the same as in the proof of Theorem 2(1).

□

We conclude the section with the proof of Claim 21:

Proof: We would like to use a padding argument. However, padding arguments usually don't go well with [io] classes (because you don't know which padded length is a good one). To solve that we use the padding argument together with the instance checker for f. Details follow:

Suppose that $f \in \bigcap_{\beta<1}$[io] $AMTIME(2^{\beta\ell})$. We would like to show that for every $g \in$ E, $g \in \bigcap_{\beta<1}$[io] $AMTIME(2^{\beta\ell})$, which contradicts the hypothesis of the theorem. Let $\eta > 0$ be an arbitrary constant. Suppose that the linear-time reduction ϕ, from g to f, converts inputs of length ℓ to inputs of length $c_2\ell$. Let P be the AMTIME$(2^{\beta\ell})$ protocol for f that is correct on infinitely many input lengths, for

$\beta = \frac{\eta}{2cc_2}$, and say errors are exponentially small. The protocol for g works as follows:

Input : $x \in \{0,1\}^{\ell}$.

Protocol :

- Compute $\phi(x) = y \in \{0,1\}^{c_2\ell}$ such that $g(x) = f(y)$.
- For each $0 \leq i < cc_2$ do the following:
  - Run $\mathrm{IC}^P(y)$ where the oracle calls to P are with queries of length $cc_2\ell$, and where queries of length $cc_2\ell$ are embedded in inputs of length $cc_2\ell + i$. Note that each oracle call to P means that Arthur and Merlin run the protocol P.

  If for some i we get an answer, that's our answer, otherwise we reject.

Now, we have cc_2 attempts, in each attempt the probability we get a wrong answer is small (either there is a query answered in the AM protocol incorrectly, or the instance checker fails). Thus, the probability we get a wrong answer is small. On the other hand, if for one of the i's, M computes f correctly on input length $cc_2n + i$, then the instance checker will succeed with high probability (there may be some error because the AM oracle may err). Altogether we get a two sided $\mathrm{AMTIME}(2^{\eta\ell})$ protocol for g which is infinitely often correct.

$\square$

5 Discussion and open problems

5.1 A low-end result

Our results are all in the "high-end" setting. That is we assume a hard function for $\mathrm{AMTIME}(2^{\Omega(n)})$ and conclude a "full derandomization" placing AM in NP. We do not know how to extend these results to the "low-end" setting. That is, assume a hard function for AM and conclude a "weak derandomization" (placing AM in NSUBEXP). This happens because the Miltersen-Vinodchandran generator [25] is only known to work in the "high-end" setting.

Recently, Shaltiel and Umans [32] gave a generator construction that improves that of Miltersen and Vinodchandran [25] and also works in the "low-end" setting. However, the known reconstruction algorithm for this construction does not seem to be resilient.

The gap theorem of [19] uses a different property of reconstruction algorithms: They show that the reconstruction algorithm of [28, 6] is *learning*: The "correct" advice string a can be efficiently computed with oracle access to the hard function. They use this property to efficiently find the "correct" advice string a when the hard function is "downwards self reducible" (which they can assume in their setting) [7]. We'd like to point out that the reconstruction algorithm for the Miltersen-Vinodchandran generator is learning. We also want to point out that the reconstruction algorithm for [32, 35] isn't learning [8].

5.2 A completely resilient probabilistic reconstruction

In Lemma 16 we show that the Miltersen-Vinodchandran generator has a reconstruction algorithm that is resilient against a large family K of co-nondeterministic circuits. An interesting open problem is to construct a generator that has a reconstruction algorithm which is resilient against *all* co-nondeterministic circuits. For concreteness let us phrase the open problem precisely.

Goal: Construct a generator $G_{f,\ell,m}$ with the following properties:

- There is a constant b such that for every ℓ and m, $G_{f,\ell,m}$ stretches ℓ^b bits into m bits.
- There exists an algorithm R that is an efficient p-resilient $(t,\epsilon) \rightarrow_p m^{O(1)}$ reconstruction algorithm against *all* co-nondeterministic circuits, for $p > 1/2$ and $\epsilon = 1 - 2^{m^{\delta}-m}$.

The reader should note that this goal is different than the conclusion of Lemma 16 in two respects:

- The reconstruction algorithm R is required to output a PSV circuit for every circuit D, whereas Lemma 16 only works when D rejects very few inputs. This change allows to remove the [io-pseudo()] clause from Theorem 1. Roughly speaking, this is because we can now trust Merlin to send the distinguisher.
- The reconstruction for the generator $MV_{f,\ell,m}$ from Lemma 16 is only shown to work for $m >> 2^{\ell/d} \geq 2^{\sqrt{\ell}}$. Such parameters are the *high end* setting. The goal is phrased for the *low-end*. In the following conclusions we will require a generator which

[7] In [34] it was pointed out that this assumption is problematic when trying to extend their result to the "high-end" setting.

[8] The construction of [32, 35] works for derandomizing both BPP and AM. The reconstruction algorithm given there for BPP is "learning". However, the one given for AM isn't and requires additional advice which we do not know how to compute even with oracle access to f on inputs of length ℓ.

works whenever m is a polynomial in ℓ. We are also more liberal in the length of the seed. Whereas the Miltersen-Vinodchandran generator uses a seed of length $O(\ell)$, we allow G to use a seed of size ℓ^b.

The two changes above will give an improved version of our main theorem: If EXP $\neq$ AM then AM $\subseteq \cap_{\delta>0}[io]\mathrm{NTIME}(2^{n^\delta})$.

5.3 Placing AM in Σ_2^{P}

One of our motivations for studying this problem is the attempt to prove that AM $\subseteq \Sigma_2^{\mathrm{P}}$. We remark that AM is known to be in Π_2^{P} (and therefore in Σ_3^{P}), and that [30] constructs an oracle relative to which AM isn't in Σ_2^{P}.

Intuitively, a "gap-theorem" is helpful as it says that AM is either weak (say AM $\subseteq$ NP and then certainly AM $\subseteq \Sigma_2^{\mathrm{P}}$) or AM is strong (say AM = EXP). However, in the second case AM is closed under complement and thus AM $\subseteq \Sigma_2^{\mathrm{P}}$.

Following this line precisley gives that if the goal above is achieved then AM $\subseteq \cap_{\delta>0}[io]\Sigma_2^{\mathrm{TIME}(2^{n^\delta})}$. Thus, the goal implies some form of *unconditional* derandomization of AM.

Acknowledgements

We thank Oded Goldreich, Russell Impagliazzo, Valentine Kabanets, Alon Rosen and Chris Umans for helpful discussions. We thank Oded Goldreich and Rahul Santhanam for very thorough and very helpful comments on the paper.

References

[1] W. Aiello and J. Hastad. Statistical zero-knowledge languages can be recognized in two rounds. *Journal of Computer and System Sciences*, 42(3):327–345, 1991.

[2] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[3] V. Arvind and J. Kobler. On resource-bounded measure and pseudorandomness. *17th International Conference on Foundations of Software Technology and Theoretival Computer Science*, 1997.

[4] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *In Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 21–31, 1991.

[5] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[6] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[7] 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.

[8] M. Blum and S. Kannan. Designing programs that check their work. *Journal of the ACM*, 42(1):269–291, 1995.

[9] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, Nov. 1984.

[10] M. Furer, O. Goldriech, Y. Mansour, M. Sipser, and S. Zachos. On completeness and soundness in interactive proof systems. *S. Micali, editor, Advances in Computing Research 5: Randmness and Computation*, pages 429–442, 1989.

[11] O. Goldreich and S. Goldwasser. On the limits of nonapproximability of lattice problems. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98)*, pages 1–9, New York, May 23–26 1998. ACM Press.

[12] 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(1):691–729, 1991.

[13] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18(1):186–208, 1989.

[14] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. *volume 5 of Advances in Computing Research*, pages 73–90, 1989.

[15] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. In *Proceedings of the 16th Annual Conference on Computational Complexity (CCC-01)*, pages 2–12, 2001.

[16] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Near-optimal conversion of hardness into pseudorandomness. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1999.

[17] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and pseudo-randomn generators with optimal seed-length. In *Proceedings of the Thirty-second Annual ACM Symposium on the Theory of Computing*, 21–23 May 2000.

[18] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC '97)*, pages 220–229, 1997.

[19] R. Impagliazzo and A. Wigderson. Randomness vs. time: De-randomization under a uniform assumption.

In *39th Annual Symposium on Foundations of Computer Science*, pages 734–743, 1998.

[20] V. Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. In *Proceedings of the 15th Annual IEEE Conference on Computational Colmplexity (COCO-00)*, pages 150–157, 2000.

[21] V. Kabanets. Derandomization: a brief overview. *Bulletin of the European Association for Theoretical Computer Science*, 76:88–103, Feb. 2002. Columns: Computational Complexity.

[22] R. M. Karp and R. J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Twelfth Annual ACM Symposium on Theory of Computing (STOC '80)*, pages 302–309, New York, Apr. 1979. ACM.

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

[24] C. Lu. Derandomizing arthur-merlin games under uniform assumptions. *Computational Complexity*, 10(3):247–259, 2001.

[25] P. B. Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. In *40th Annual Symposium on Foundations of Computer Science (FOCS '99)*, pages 71–80, 1999.

[26] Nisan and Ta-Shma. Extracting randomness: A survey and new constructions. *JCSS: Journal of Computer and System Sciences*, 58, 1999.

[27] N. Nisan. Extracting randomness: How and why: A survey. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity (CCC-96)*, pages 44–58, Los Alamitos, May 24–27 1996. IEEE Computer Society.

[28] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, Oct. 1994.

[29] A. Sahai and S. Vadhan. A complete promise problem for statistical zero-knowledge. In *Proceedings of the 38th Annual Symposium on the Foundations of Computer Science*, pages 448–457. IEEE, 1997.

[30] M. Santha. Relativized arthur-merlin versus merlin-arthur games. *Information and Computation, 80(1):44-49*, 1989.

[31] R. Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the European Association for Theoretical Computer Science*, 77:67–, June 2002. Also available on http://www.wisdom.weizmann.ac.il/ ronens.

[32] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *42nd IEEE Symposium on Foundations of Computer Science (FOCS-01)*, pages 648–657, 2001.

[33] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, 1999.

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

[35] C. Umans. Pseudo-random generators for all hardnesses. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC-02)*, pages 627–634, 2002.

[36] A. C. Yao. Theory and applications of trapdoor functions. In *23rd annual symposium on foundations of computer science (Chicago, Ill., 1982)*, pages 80–91. IEEE, New York, 1982.