Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 57 (2022)

When Arthur has Neither Random Coins nor Time to Spare: Superfast Derandomization of Proof Systems

Lijie Chen [*] Roei Tell [†]

May 22, 2022

Abstract

What is the actual cost of derandomization? And can we get it for free? These questions were recently raised by Doron et. al (FOCS 2020) and have been attracting considerable interest. In this work we extend the study of these questions to the setting of **derandomizing interactive proofs systems**.

First, we show conditional derandomization of $\mathcal{MA}$ and of $\mathcal{AM}$ with **optimal runtime overhead**, where optimality is under the #NSETH assumption. Specifically, denote by $\mathcal{AMTIME}^{[\rightleftharpoons c]}[T]$ a protocol with c turns of interaction in which the verifier runs in polynomial time T. We prove that for every $\epsilon > 0$ there exists $\delta > 0$ such that:

$$\mathcal{MATIME}[T] \subseteq \mathcal{NTIME}[T^{2+\epsilon}]\,,$$
$$\mathcal{AMTIME}^{[\rightleftharpoons c]}[T] \subseteq \mathcal{NTIME}[n \cdot T^{\lceil c/2 \rceil + \epsilon}]\,,$$

assuming the existence of properties of Boolean functions that can be recognized quickly from the function's truth-table such that functions with the property are hard for proof systems that receive near-maximal amount of non-uniform advice.

To obtain faster derandomization, we introduce the notion of a **deterministic doubly efficient argument system**. This is a doubly efficient proof system in which the verifier is *deterministic*, and the soundness is relaxed to be computational, as follows: For every probabilistic polynomial-time adversary $\tilde{P}$, the probability that $\tilde{P}$ finds an input $x \notin L$ and misleading proof π such that $V(x, \pi) = 1$ is negligible.

Under strong hardness assumptions, we prove that **any constant-round doubly efficient proof system can be compiled into a deterministic doubly efficient argument system, with essentially no time overhead**. As one corollary, under strong hardness assumptions, for every $\epsilon > 0$ there is a deterministic verifier V that gets an n-bit formula Φ of size $2^{o(n)}$, runs in time $2^{\epsilon \cdot n}$, and satisfies the following: An honest prover running in time $2^{O(n)}$ can print, for every Φ, a proof π such that $V(\Phi, \pi)$ outputs the number of satisfying assignments for Φ; and for every adversary $\tilde{P}$ running in time $2^{O(n)}$, the probability that $\tilde{P}$ finds Φ and π such that $V(\Phi, \pi)$ outputs an incorrect count is $2^{-\omega(n)}$.

[*]Massachusetts Institute of Technology, MA. Email: wjmzbmr@gmail.com

[†]The Institute for Advanced Study at Princeton NJ and the DIMACS Center at Rutgers University, NJ. Email: roei.tell@gmail.com

ISSN 1433-8092

Contents

1 Introduction

We study the well-known problem of *eliminating randomness from interactive proof systems*. While we do not expect to be able to eliminate randomness from general proof systems (as that would imply that $\mathcal{NP} = \mathcal{IP} = \mathcal{PSPACE}$, by [Sha92]), a classical line of works suggests that for proof systems that only use constantly many rounds of interaction, this might be doable. In particular, assuming sufficiently strong circuit lower bounds,[1] we have that $\mathcal{AM} = \mathcal{NP}$ (for a subset of prominent works in this area, see e.g., [KM98; IKW02; MV05; SU05; GSTS03; SU07]).

Recently, Doron *et al.* [DMO+20] raised the fine-grained derandomization question of what is the *minimal time overhead* that we need to pay when eliminating randomness from probabilistic algorithms. In their paper and in two subsequent works [CT21b; CT21a], the following picture emerged: Under sufficiently strong lower bounds for algorithms that use non-uniformity, we can derandomize probabilistic algorithms that run in time T by deterministic algorithms that run in time $n \cdot T^{1+\epsilon}$, for an arbitrarily small $\epsilon > 0$ (see [CT21b] for details);[2] and furthermore, if we are willing to settle for derandomization that errs on a negligible fraction of inputs over any polynomial-time samplable distribution, then we can derandomize probabilistic time T in deterministic time $n^{\epsilon} \cdot T$ for an arbitrarily small $\epsilon > 0$ (see [CT21a]).

The current paper studies the question of *superfast derandomization of proof systems*: We ask what is the minimal time overhead that we need to pay when eliminating randomness from proof systems such as $\mathcal{MA}$ and $\mathcal{AM}$. While it is well-known that proof systems with constantly many rounds can be simulated by proof systems with two rounds (see [BM88]), in this work we care about *fine-grained time bounds*, and therefore we care about the precise number of rounds in any such system. We denote by $\mathcal{AMTIME}^{[\rightleftharpoons c]}[T]$ the set of problems solved by proof systems that use c turns of interaction, where in each turn the relevant party communicates $T(n)$ bits (recall that the verifier just sends random bits; see Definition 3.5 for precise details).

Our contributions, at a bird's eye. First, we construct *essentially optimal* derandomization algorithms for $\mathcal{MA}$*, and for* $\mathcal{AM}$ *protocols with constantly many rounds*. The derandomization algorithms are constructed under appealing hardness hypotheses, which compare favorably to hypotheses from previous works (and the claim that they are essentially optimal is under the assumption #NSETH). The results for $\mathcal{MA}$ and for $\mathcal{AM}$ are presented in Sections 1.1 and 1.2, respectively.

Secondly, we introduce the notion of *deterministic doubly efficient argument systems*, which are deterministic doubly efficient proof systems such that no polynomial-time adversary can find, with non-negligible probability, an input $x \notin L$ and a proof π such that the verifier accepts x given proof π. Under strong hardness assumptions, we compile every constant-round doubly efficient proof system into a deterministic doubly efficient argument system with essentially no time overhead.

As a special case of the latter result, under strong hardness assumptions, we construct a verifier that certifies the number of solutions for a given n-bit formula of size $2^{o(n)}$ in time $2^{\epsilon \cdot n}$, for any $\epsilon > 0$, such that no $2^{O(n)}$-time algorithm can find a formula and a proof on which this verifier errs. The general result mentioned in the preceding paragraph as well as the special case of #SAT are presented in Section 1.3.

[1]For example, it suffices to assume that for some $\epsilon > 0$ it holds that $\mathcal{E} = \mathcal{DTIME}[2^{O(n)}]$ is hard for SVN circuits of size $2^{\epsilon \cdot n}$ on all input lengths (see [MV05]).

[2]This is tight for worst-case derandomization, under the hypothesis #NSETH (see Assumption 3.19).

Useful properties. Many of our hardness hypotheses will rely on the existence of *useful properties*, in the sense of Razborov and Rudich [RR97]: Loosely speaking, these are algorithms verifying that a given string is the truth-table of a function that is hard for a certain class. Specifically, for two complexity classes $\mathcal{C}$ and $\mathcal{C}'$, we say that a property Π of truth-tables is $\mathcal{C}'$-constructive and useful against $\mathcal{C}$ if there is a $\mathcal{C}'$-algorithm that decides whether a truth-table is in Π, and if every truth-table in Π is hard for algorithms from $\mathcal{C}$ (see Definition 3.3 for precise details).[3]

The reason that we consider useful properties is that *they are necessary for any derandomization of proof systems, even just of $\mathcal{MA}$*. In fact, the useful properties that are necessary are constructive in *quasilinear time* (this follows using the well-known approach of Williams [Wil13; Wil16]; see Appendix A for a proof). Accordingly, we will consider useful properties in which truth-tables of length $N = 2^n$ can be recognized in non-deterministic polynomial time $N^k = 2^{k \cdot n}$ or even in non-deterministic near-linear time $N^{1+\epsilon} = 2^{(1+\epsilon)\cdot n}$, for a small constant $\epsilon > 0$.

Previous works deduced superfast derandomization from hard functions whose truth-tables can be printed quickly (i.e., they have bounded amortized complexity; see, e.g., [CT21b]). The assumptions above are more relaxed: We only require recognizing the truth-table (rather than printing it), we allow non-determinism (i.e., proof) in recognizing the truth-table, and we allow for a collection of hard truth-tables per input length, rather than just a single hard truth-table per input length.

1.1 Superfast derandomization of $\mathcal{MA}$

Our first result is a derandomization of $\mathcal{MA}$ that induces a (near-)quadratic time overhead. This derandomization is (conditionally) tight, and it relies on the existence of constructive properties that are useful against $\mathcal{NP} \cap co\mathcal{NP}$ machines that receive near-maximal non-uniform advice. That is:

Theorem 1.1 (quadratic derandomization of $\mathcal{MA}$ and useful properties)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists an $\mathcal{NTIME}[N^{2+\epsilon/4}]$-constructive property useful against $(\mathcal{N} \cap co\mathcal{N})\mathcal{TIME}[2^{(2-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$. Then,*

$$pr\mathcal{MATIME}[T] \subseteq pr\mathcal{NTIME}[T^{2+\epsilon}] \,.$$

We stress that, in contrast to derandomization of $pr\mathcal{BPP}$, for $\mathcal{MA}$ the quadratic time overhead above might be unavoidable. To see this, assume that #NSETH holds; that is, for every $\epsilon > 0$ there is no non-deterministic algorithm running in time $2^{(1-\epsilon)\cdot n}$ and *counting* the number of satisfying assignments for a given n-bit formula of size $2^{o(n)}$ (see Assumption 3.19). Then, for every $\epsilon > 0$ we have that $\mathcal{MATIME}[T] \not\subseteq \mathcal{NTIME}[T^{2-\epsilon}]$: The reason is that $\mathcal{MA}$ algorithms may simulate a sumcheck protocol to count the number of satisfying assignments of a given formula in time $\tilde{O}(2^{n/2})$ (see [Wil16]). See Theorem 6.1 for a more general statement.

The technical result underlying Theorem 1.1 is actually stronger: Under the hypothesis we deduce that $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{NTIME}[T^{2+\epsilon}]$ (see Theorem 4.2), and if we assume that a *deterministic* algorithm can quickly print the hard truth-tables, we can deduce that $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[T^{2+\epsilon}]$ (see Theorem 4.3). The latter extension to $pr\mathcal{BPTIME}$ compares favorably to assumptions needed to get the same conclusion in previous works [DMO+20; CT21b]; we elaborate on this in Section 1.4.

[3]We also assume non-triviality, i.e. for every $n \in \mathbb{N}$ the property Π contains functions on n input bits.

1.2 Superfast derandomization of $\mathcal{AM}$

Our next step is to consider derandomization of $\mathcal{AM}$ protocols, and for simplicity of presentation we focus on protocols with perfect completeness.[4] Our second main result is a derandomization of $\mathcal{AMTIME}^{[\rightleftharpoons c]}[T]$, for any constant number $c \in \mathbb{N}$ of turns, that is (conditionally) tight: Loosely speaking, we prove that any such protocol can be simulated in non-deterministic time $\approx T^{c/2}$, assuming the existence of constructive properties that are useful against $\mathcal{MAM}$ protocols that receive near-maximal non-uniform advice. That is:

Theorem 1.2 (superfast derandomization of $\mathcal{AM}$)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that for every $k \geq 1$ there exists an $\mathcal{NTIME}[N^{k+\epsilon/3}]$-constructive property useful against $\mathcal{MAM}[2^{(1-\delta)\cdot k\cdot n}]/2^{(1-\delta)\cdot n}$. Then, for every polynomial T it holds that*

$$pr\mathcal{AMTIME}[T] \subseteq pr\mathcal{NTIME}[n \cdot T^{1+\epsilon}] ,$$

and furthermore for every constant $c \in \mathbb{N}$ it holds that

$$pr\mathcal{AMTIME}^{[\rightleftharpoons c]}[T] \subseteq pr\mathcal{NTIME}[n \cdot T^{\lceil c/2 \rceil + \epsilon}] .$$

The conclusions in Theorem 1.2 are essentially tight, assuming #NSETH; see Theorem 6.2 for details. Moreover, the hardness assumption is quite mild compared to what one might expect; let us explain why we claim so, comparing our assumption to ones in previous results. Recall that in classical results, to derandomize $\mathcal{AM}$ we assume lower bounds for the non-uniform analogue of $\mathcal{NP}$ (see, e.g., [SU05]). The assumption above is for a stronger class, namely the non-uniform analogue of $\mathcal{MAM}$. However, interestingly, this assumption still suffices to *optimally* derandomize $\mathcal{AM}$ with *any* constant number of turns; that is, the derandomization overhead is optimal for any number of rounds given hardness only for (non-uniform) $\mathcal{MAM}$.

Now, observe that our assumption refers to hardness for time bounds $2^{kn} \gg 2^n$ (i.e., upper bounds of 2^{kn} and lower bounds of $2^{(1-\delta)\cdot kn}$, for every constant $k \geq 1$). This is quantitatively reminiscent of a hardness assumption from [CT21b] that was used to deduce that $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[n \cdot T^{1+\epsilon}]$. However, the latter result also needed an additional hardness assumption, namely the existence of *one-way functions*; in contrast, in Theorem 1.2 *we do not rely on any cryptographic assumption*. In fact, similarly to Section 1.1, the techniques underlying Theorem 1.2 also extend to the setting of derandomization of $pr\mathcal{BPP}$, and allow us to deduce a conclusion as in [CT21b] without cryptographic assumptions. For further details see Section 1.4.

A strengthening. We further strengthen Theorem 1.2 by relaxing its hypothesis. Specifically, recall that in Theorem 1.2 we assumed that for every $k \geq 1$ there is a corresponding useful property, and let us denote it by $\mathcal{L}_k$. We prove a stronger version that relies on the same hardness hypothesis for $\mathcal{L}_1$, but for every $k > 1$ only requires $\mathcal{L}_k$ to be hard for $\mathcal{NTIME}[2^{(1-\delta)\cdot k\cdot n}]/2^{(1-\delta)\cdot n}$ (rather than for $\mathcal{MAM}$ protocols with the same complexity). See Section 5.3 and Theorem 5.18 for details.

[4]Our results extend to the case of protocols with imperfect completeness, at the cost of strengthening the hardness assumptions (see, e.g., Remarks 5.4 and 7.6).

Uniform tradeoffs for $\mathcal{AM} \cap co\mathcal{AM}$. The results above relied on hardness for protocols that use a large number of non-uniform advice bits. Classical results were able to deduce derandomization of the more restricted class $\mathcal{AM} \cap co\mathcal{AM}$ relying only on hardness assumptions for *uniform* protocols (see, e.g., [GSTS03; SU07]).

Indeed, we are able to show an "extreme high-end" analogue of these results too. Assuming that there exists a function whose truth-tables can be recognized in near-linear time but that is hard for 7-round protocols running in time $2^{(1-\delta)\cdot n}$, we show that $\mathcal{AM} \cap co\mathcal{AM}$ can be derandomized with only a quadratic time overhead:

Theorem 1.3 (uniform tradeoffs for superfast derandomization)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists $L \notin \mathtt{i.o.}(\mathcal{MA} \cap co\mathcal{MA})\mathcal{TIME}_2^{[\rightleftharpoons 7]}[2^{(1-\delta)\cdot n}]$ such that truth-tables of L of length $N = 2^n$ can be recognized in non-deterministic time $N^{1+\epsilon/3}$.* [5] *Then, for every time-computable T it holds that*

$$(\mathcal{AM} \cap co\mathcal{AM})\mathcal{TIME}[T] \subseteq (\mathcal{N} \cap co\mathcal{N})\mathcal{TIME}[T^{2+\epsilon}].$$

Needless to say, hardness for protocols with constantly many rounds did not appear in previous results [GSTS03; SU07], and this is because previous results did not consider fine-grained time bounds (in which case constantly many rounds can be simulated by two rounds [BM88]).

1.3 A lunch that looks free: Deterministic doubly efficient arguments

In two previous works [CT21b; CT21a], the way to bypass the #NSETH barrier and obtain faster derandomization results was to consider derandomization *that is allowed to err on a tiny fraction of inputs*, and to construct derandomization algorithms that use new *non-black-box techniques*. In the current work we follow in the same vein.

Let us first state a striking special case of the general results that we prove. Under a very strong hardness assumption (we will be formal about the assumption when we state the general case), we prove that for every $\epsilon > 0$ there is a verifier that *counts the number of satisfying assignments for a given n-bit formula in time $2^{\epsilon\cdot n}$*, with one caveat: The soundness of the verifier is only *computational*, rather than information-theoretic.

Theorem 1.4 (effectively certifying #SAT in deterministic time $2^{\epsilon\cdot n}$; informal)**.** *Assume that a certain lower bound for probabilistic machines with oracle access to proof systems holds (see Theorem 7.9). Then, for every $\epsilon > 0$ there is a deterministic verifier V that gets as input an n-bit formula Φ of size at most $2^{o(n)}$, runs in time $2^{\epsilon\cdot n}$, and satisfies the following:*

1. **(Completeness.)** *There is an algorithm that, given any input formula Φ as above, runs in time $2^{O(n)}$ and outputs a proof π such that $V(\Phi, \pi) = \#\mathsf{SAT}(\Phi)$.* [6]
2. **(Computational soundness.)** *For every probabilistic algorithm $\tilde{P}$ running in time $2^{O(n)}$, the probability that $\tilde{P}(1^n)$ prints an n-bit formula Φ of size $2^{o(n)}$ and proof π such that $V(\Phi, \pi) \notin \{\perp, \#\mathsf{SAT}(\Phi)\}$ is $2^{-\omega(n)}$.*

We remind the reader that constructing a verifier as above without the relaxation of computational hardness (i.e., a verifier that is sound for all inputs and with any proof) is believed to be impossible; this is known as the #NETH assumption, which is weaker than the "strong" version #NSETH mentioned above. To the best of our knowledge, the existence of a verifier such as the one in Theorem 1.4 was not conjectured before.

[5] That is, we have that $\mathrm{tt}(L) \in \mathcal{NTIME}[N^{1+\epsilon/3}]$, where $\mathrm{tt}(L) = \{f_n \in \{0,1\}^{N=2^n}\}_{n\in\mathbb{N}}$ and f_n is the truth-table of L on n-bit inputs. Also, the subscript "2" in the notation $(\mathcal{MA} \cap co\mathcal{MA})\mathcal{TIME}_2^{[\rightleftharpoons 7]}$ refers to $\mathcal{MA} \cap co\mathcal{MA}$ protocols with imperfect completeness (say, 2/3).

[6] We denote by $\#\mathsf{SAT}(\Phi)$ the number of assignments that satisfy the formula Φ.

1.3.1 Deterministic doubly efficient argument systems

More generally, consider the notion of doubly efficient proof systems, as introduced by Goldwasser, Kalai, and Rothblum [GKR15] (for a survey, see [Gol18]). These are interactive proof systems in which the verifier is very fast (say, runs in quasilinear time) and the honest prover is slower, but still efficient (say, runs in polynomial time). We denote by $\mathsf{de}\mathcal{IP}^{[=c]}[T]$ the class of problems solvable by doubly efficient proof systems with c turns of interaction, a verifier that runs in time T, and an honest prover that runs in time $\text{poly}(T)$ (see Definition 3.6).[7]

We initiate a study of deterministic doubly efficient argument systems. Recall that argument systems are a standard relaxation of proof systems in which soundness is guaranteed only against computationally bounded provers (see, e.g., [Tha22] for an exposition, and [Gol18, Section 1.5] for a discussion of probabilistic doubly efficient argument systems). In our notion of deterministic doubly efficient argument systems *we require the verifier to be deterministic* (i.e., an $\mathcal{NP}$-type verifier), and the honest prover to be efficient, but we further relax the soundness requirement such that it is guaranteed only against inputs that can be found by computationally bounded adversaries.

Definition 1.5 (deterministic doubly efficient argument system)**.** *We say that $L \subseteq \{0,1\}^*$ is in $\mathsf{de}\mathcal{DARG}[T]$ if there exists a deterministic T-time verifier V such that the following holds:*

1. *There exists a deterministic algorithm P that, when given $x \in L$, runs in time $\text{poly}(T)$ and outputs π such that $V(x, \pi) = 1$.*

2. *For every polynomial p, and every probabilistic algorithm $\tilde{P}$ running in time $p(T)$, and every sufficiently large $n \in \mathbb{N}$, the probability that $\tilde{P}(1^n)$ prints $x \notin L$ of length n and $\pi \in \{0,1\}^{T(n)}$ such that $V(x, \pi) = 1$ is $T(n)^{-\omega(1)}$.*

Intuitively, the meaning of the soundness condition in Definition 1.5 is that no efficient adversary can find a false claim $x \notin L$ and then convince the verifier that the false claim is true, except with negligible probability.

One may wonder whether we can relax the condition that the honest prover is efficient. We remark that such a definition would be too relaxed, and potentially allow the resulting class to contain essentially all possible problems (even uncomputable ones); we elaborate and further discuss our definitional choices in Section 3.2.2.

1.3.2 Derandomizing doubly efficient proofs into deterministic doubly efficient arguments

Our main result in this context is that, under strong hardness hypotheses, *every constant round doubly efficient proof system* can be simulated by a deterministic doubly efficient argument system *with essentially no time overhead*. We stress that even if we start with a system that has many rounds, after this simulation we still end up with an $\mathcal{NP}$-type verifier that has essentially the same time complexity. This opens the door to first using a protocol with many rounds to solve a problem faster, and then simulating this fast protocol by a deterministic doubly efficient argument system of roughly the same complexity. (This is the approach that underlies Theorem 1.4.)

[7]The original definition of doubly efficient proof systems uses the fixed time bound $T = \tilde{O}(n)$ (see, e.g., [GKR15; Gol18]). In this work we also consider more general time bounds.

As a first step, consider any $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T]$ protocol in which the verifier uses only $n^{o(1)}$ random coins. We simulate it by a deterministic doubly efficient argument system, under the following assumption: There exists $f\colon \{0,1\}^{O(T)} \to \{0,1\}^{n^{o(1)}}$ such that each output bit of f can be computed in time $\bar{T}$, but no probabilistic machine with oracle access to $pr\mathcal{AMTIME}^{[\rightleftharpoons c]}[n]$ can print (an approximate version) of the entire string $f(x)$ in time slightly larger $\bar{T}$; and this hardness holds with high probability when choosing x from any efficiently samplable distribution.

Assumption 1.6 (non-batch-computability assumption; see Assumption 7.4)**.** *The* $(N \mapsto K, K', \eta)$-non-batch-computability *assumption for time $\bar{T}$ with oracle access to $\mathcal{O}$ is the following. There exists a function $f\colon \{0,1\}^N \to \{0,1\}^K$ such that:*

1. *There exists a deterministic algorithm that gets input $(z,i) \in \{0,1\}^N \times [K]$ and outputs the i^{th} bit of $f(z)$ in time $\bar{T}$.*
2. *For every oracle machine M running in time $\bar{T} \cdot K'$ with oracle access to $\mathcal{O}$, and every distribution $\mathbf{z}$ over $\{0,1\}^N$ that is samplable in time polynomial in $\bar{T}$, with probability at least $1-(\bar{T})^{-\omega(1)}$ over choice of $z \sim \mathbf{z}$ it holds that $\Pr[M^{\mathcal{O}}(z)_i \neq f(z)_i] \geq \eta$, where the probability is over $i \in [K]$ and the random coins of M.* [8]

Indeed, the name "non-batch-computability" in Assumption 1.6 refers to the fact that each individual output bit of f is computable in time $\bar{T}$, whereas printing (an approximate version) of the entire K-bit string is hard for time $\bar{T} \cdot K' > \bar{T}$ (even with oracle access to $\mathcal{O}$). Our first general result is then the following:

Theorem 1.7 (simulating doubly efficient proof systems with few coins by deterministic doubly efficient argument systems; informal, see Theorem 7.5)**.** *Let $L \in \mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T]$ such that the verifier for L uses only $R(n) = n^{o(1)}$ random coins. Assume that for some $\alpha, \beta \in (0,1)$ the $(N \mapsto K, K^{\beta}, \alpha)$-non-batch-computability assumption holds for time $\bar{T}$ with oracle queries of length $O(T)$ to $pr\mathcal{AMTIME}_2^{[\rightleftharpoons c]}[n]$, where*

$$\begin{aligned} N(n) &= n + c \cdot T \\ K(n) &= \mathrm{poly}(R(n)) \\ \bar{T}(n) &= T \cdot K\,. \end{aligned}$$

Then $L \in \mathsf{de}\mathcal{DARG}[T \cdot n^{o(1)}]$.

The hardness hypothesis in Theorem 1.7 is very strong, but the corresponding derandomization conclusion is also exceptionally strong. We discuss the hardness hypothesis in detail in Section 7.2.1, and in particular compare it to known results about batch-computing functions by interactive protocols, by Reingold, Rothblum and Rothblum [RRR21; RRR18]. Indeed, even an assumption considerably stronger than the one in Theorem 1.7 still seems reasonable, in particular an assumption in which the oracle is a linear-space machine (rather than a linear-time verifier in an interactive protocol); see Section 7.2.1 for details. We also comment that "non-batch-computability" hardness against probabilistic machines (without oracles) is necessary for derandomization with no overhead in limited special cases (see [CT21a, Section 6.3]).

Theorem 1.7 implies Theorem 1.4 as a special case, because the underlying probabilistic proof system for #SAT (i.e., a constant-round sumcheck protocol) uses a number of random coins that is logarithmic in the running time. Moreover, in the special

[8]In all our results we will use this assumption with $\bar{T}(N) = \mathrm{poly}(N)$, in which case the error bound $(\bar{T})^{-\omega(1)}$ is just any negligible function in the input length.

case of Theorem 1.4 we are able to *considerably relax the hypothesis*, relying on certain properties of the proof system for #SAT; see Theorems 3.17, 7.8 and 7.9 for details.

The general case. Turning to the general case of simulating doubly efficient proof systems (without restricting the number of random coins) by deterministic doubly efficient argument systems, we can do so under one additional hypothesis:

Theorem 1.8 (simulating general doubly efficient proof systems by deterministic doubly efficient argument systems; informal, see Corollary 7.12)**.** *For every $\alpha, \beta, \epsilon \in (0,1)$ there exists $\eta, \delta > 0$ such that for every polynomial $T(n)$ and constant $c \in \mathbb{N}$ the following holds. Assume that:*

1. *There exists $L^{\mathsf{hard}} \notin \mathcal{MATIME}^{[\rightleftharpoons c+1]}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ such that given $n \in \mathbb{N}$, the truth-table of L^{hard} of n-bit inputs can be printed in time $2^{(1+\epsilon/3)\cdot n}$.*

2. *The $(N \mapsto K, K^{\beta}, \alpha)$-non-batch-computability assumption holds for time $\bar{T}$ with oracle queries of length $O(T)$ to $pr\mathcal{AMTIME}_2^{[\rightleftharpoons c]}[n]$, where $N = n + c \cdot T^{1+\epsilon/2}$ and $K = \mathrm{polylog}(T)$ and $\bar{T} = T^{1+\epsilon/2} \cdot K$.*

Then, $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{de}\mathcal{DARG}[T^{1+\epsilon}]$.

The additional hypothesis in Theorem 1.8 (i.e., the one in Item (1)) is a strengthening of the one in Theorem 1.2 that refers to the value of $k = 1$. The strengthening is because now we consider hardness for protocols with $c + 1$ turns rather than 3 turns, we assume a single hard problem rather than a useful property, and the upper-bound on the complexity of truth-tables is deterministic rather than non-deterministic.

1.3.3 Deterministic argument systems for $\mathcal{NP}$-relations

In Definition 1.5 the honest prover runs in time $\bar{T} = \mathrm{poly}(T)$, and therefore such argument systems exist only for problems in $\mathcal{DTIME}[\bar{T}]$. An alternative definition, which would allow constructing deterministic argument systems for $\mathcal{NTIME}[\bar{T}]$-relations, is to give the honest prover a witness for the corresponding relation as auxiliary input (while still insisting that it runs in time $\bar{T}$). Our proofs extend to this setting, allowing to simulate $\mathcal{AMTIME}^{[\rightleftharpoons c]}[T]$ protocols for $\mathcal{NTIME}[\bar{T}]$-relations, in which the honest prover is efficient given a witness for the relation, into deterministic argument systems running in time $T^{1+\epsilon}$. For further details see Remark 7.7.

1.4 Implications for derandomization of $pr\mathcal{BPP}$

Our proof techniques also extend to the setting of derandomization of $pr\mathcal{BPP}$ (rather than just of proof systems), and in this setting they yield results that compare favorably to previous works. For derandomization in this setting we replace the assumptions about useful properties with assumptions that truth-tables of hard functions can be efficiently printed. (This is since our derandomization algorithm cannot guess-and-verify a hard truth-table, but needs to efficiently print it.) It is useful to think of an algorithm that prints a truth-table of a function as "batch-computing" the function.

The techniques underlying Theorem 1.1 extend to show a quadratic-time derandomization of $pr\mathcal{BPP}$, assuming that a function whose truth-tables can be printed in time $2^{(2+\epsilon/3)\cdot n}$ is hard for $\mathcal{NTIME}[2^{(2-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ (see Theorem 4.3).[9] This compares favorably to two relevant results in the previous works [DMO+20; CT21b], as follows. (In the table below, the upper-bound is for a deterministic algorithm that prints the truth-table of the hard function on n bits.)

Upper bound	**Lower bound**	**Time overhead**	
$2^{(2+\Theta(\epsilon))\cdot n}$	$\mathcal{NTIME}[2^{(2-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$	$T^{2+\epsilon}$	Theorem 4.3
$2^{(2+\Theta(\epsilon))\cdot n}$	$\mathcal{MATIME}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$	$T^{2+\epsilon}$	[DMO+20]
$2^{(3/2)\cdot n}$	$\mathcal{NTIME}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$	$T^{3+\epsilon}$	[CT21b, Thm 1.8]

While the three results above are formally incomparable, in the current work we are able to obtain derandomization with quadratic time overhead from hardness for $\mathcal{NTIME}$ machines with advice, whereas previous works either assumed hardness for $\mathcal{MATIME}$ machines with advice, or paid a cubic time overhead.

Moreover, the techniques underlying Theorem 1.2 extend to yield the conclusion $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[n \cdot T^{1+\epsilon}]$, which is optimal under #NSETH, under an assumption that compares favorably to the one used to obtain derandomization with such overhead in our previous work [CT21b]. The previous work obtained this conclusion using a cryptographic assumption (the existence of one-way functions) as well as a hardness assumption that is necessary for obtaining the conclusion using PRGs. In the current work we are able to replace the cryptographic assumption with a hardness assumption for $\mathcal{MA}$ protocols with advice (see Theorem 5.5 for precise details).

2 Technical overview

Most of the proofs in this work sequentially build on the ideas of each other, and therefore we encourage readers read the current section sequentially.

In Section 2.1 we describe the simplest proof, which is of Theorem 1.1. Then in Section 2.2 we build on this proof to prove Theorem 1.2. In Section 2.3 we describe the more involved proofs in this paper, which are for the results Section 1.3; these are inherently different from other proofs, and use non-black-box techniques while also relying on the previous proofs. We conclude by explaining the proof of Theorem 1.3 in Section 2.4 (this proof does not rely on Section 2.3).

Basic building-block: A simple and efficient PRG. The starting point for our results is a very simple PRG, which was recently used for superfast derandomization in [DMO+20; CT21b] (its ideas date back to [Sip88] and variations on them have been used for derandomization of proof systems, e.g. [MV05]). Recall that a reconstructive PRG is a pair of efficient algorithms: A generator Gen maps a truth-table f to a list G^f of strings; and a reconstruction Rec maps every distinguisher D for G^f to an efficient procedure Rec^D that computes f. [10] Indeed, if f is hard for algorithms with complexity as that of Rec^D, then it follows that G^f is pseudorandom.

[9] For simplicity, in this section we assert lower bounds for $\mathcal{NTIME}$ and $\mathcal{MATIME}$. The results that we mention only require hardness for $(\mathcal{N} \cap co\mathcal{N})\mathcal{TIME}$ and $(\mathcal{MA} \cap co\mathcal{MA})\mathcal{TIME}$, respectively.

[10] When we say that D is a distinguisher for a list $G^f \subseteq \{0,1\}^n$, we mean that D distinguishes the uniform distribution over the list from a truly uniform n-bit string.

The simple reconstructive PRG works as follows. Given f, the generator Gen encodes it to $\bar{f} = \mathsf{Enc}(f)$ by an error-correcting code that is encodable in near-linear time and locally list-decodable, partitions $\bar{f}$ into consecutive substrings of size $N = |f|^{.99}$, and outputs the list of $|\bar{f}|/N \approx N^{.01}$ substrings. Indeed, this generator Gen runs in near-linear time, and is thus particularly suitable for constructing superfast derandomization algorithms. In contrast, the reconstruction Rec is inefficient, and in addition only works when the distinguisher $D\colon \{0,1\}^N \to \{0,1\}$ is extremely biased; for example, when D accepts all but $2^{N^{.99}}$ strings and yet rejects 1% of the strings in G^f.[11]

Note that for any such D, a random pairwise-independent hash function $h\colon \{0,1\}^N \to \{0,1\}^{N^{.999}}$ will not have collisions in the set $D^{-1}(0)$. The reconstruction Rec gets as advice a description of such h and the hash values of the N-bit substrings of $\bar{f}$ that G^f outputs. Given input $z \in [|f|]$, it runs the local list-decoder for Enc, and whenever the decoder queries $q \in [|\bar{f}|]$, the reconstruction *non-deterministically de-hashes* the corresponding substring in $\bar{f}$ to obtain $\bar{f}_q$. That is, denoting by $\bar{f}^{(i)}$ the substring of $\bar{f}$ that contains the index q, the reconstruction Rec guesses $z \in \{0,1\}^N$, verifies that $h(z) = h(\bar{f}^{(i)})$ using the stored hash value, verifies that $z \in D^{-1}(0)$ by querying D, and if the verifications passed then $z = \bar{f}^{(i)}$ (since there are no collisions in $D^{-1}(0)$) and Rec outputs the bit in z corresponding to index q.

We refer the reader to Proposition 4.1 for a clean statement that outlines the foregoing basic version of this reconstructive PRG.

Preliminary observations: Using this approach for derandomizing proof systems. Note that the reconstruction Rec is non-deterministic, and thus when using $(\mathsf{Gen}, \mathsf{Rec})$ above we need to assume hardness for non-deterministic procedures. Such assumptions are natural in the current context of derandomizing proof systems. In addition, the natural way to use Gen for derandomizing proof systems, which is indeed the one that we will use for many of our results, is to receive a truth-table f from a prover, verify that f is indeed hard (using an assumption that there is a constructive and useful property), and then instantiate the PRG Gen^f.

2.1 Warm-up: Proof of Theorem 1.1

The main bottleneck in the previous proofs [DMO+20; CT21b] that used $(\mathsf{Gen}, \mathsf{Rec})$ came from the fact that Gen only fools extremely biased distinguishers, whereas our goal is to construct a PRG that fools all distinguishers. In previous works this was bridged by straightforward error-reduction: They considered a procedure $\bar{D}(z)$ that uses a randomness-efficient sampler Samp and verifies that z satisfies

$$\Pr_{i \in [N^{1.01}]}[D(\mathsf{Samp}(z,i)) = 1] \approx \Pr_{r \in \{0,1\}^N}[D(r) = 1] \,,$$

and noted that if Gen^f fools $\bar{D}$, then $\mathsf{S} \circ \mathsf{Gen}^f = \left\{\mathsf{Samp}(\mathsf{Gen}^f, s)\right\}_{s \in [N^{1.01}]}$ fools D. (See [CT21b, Section 5.2] for a detailed explanation.)

To see why this is a bottleneck, note that the resulting $\bar{D}$ is of size $O(N^{2.01})$ (because Samp uses $N^{1.01}$ seeds s). Thus, if we use f that is hard for non-deterministic circuits of such size, we need $|f| > N^{2.01}$ and the number of pseudorandom strings is $(\bar{f}/N) \cdot$

[11]Thus, the reconstructive PRG $(\mathsf{Gen}, \mathsf{Rec})$ is particularly suitable for the task of *quantified derandomization*; see [GW14; DMO+20; CT21b; Tel21] for further details about this application.

$N^{1.01} > N^{2.01}$. Evaluating D on each of the strings, this yields derandomization in near-cubic time.[12]

The observation leading the way to Theorem 1.1 is simple: We do not really need f to be hard for non-deterministic circuits of size $N^{2.01}$, but only for *non-deterministic algorithms* that run in time $N^{2.01}$ and use $|f|^{.99} + N$ bits of advice. There can indeed be such truth-tables of size $|f| = N^{1.01}$, as in the hypothesis of Theorem 1.1.

To elaborate, let us sketch the proof, and in fact let us show how to derandomize $pr\mathcal{BPP}$ in near-quadratic non-deterministic time. We are given $D\colon \{0,1\}^N \to \{0,1\}$ of linear size, and we want to approximate its acceptance probability up to a small additive error. We guess f of size $|f| \approx N^{1.01}$ and verify in time $N^{2.02}$ that f is hard for non-deterministic algorithms running in time $N^{2.01}$ and using $O(N)$ bits of advice. Our generator is $\mathsf{S} \circ \mathsf{Gen}^f$, yielding $N^{1.01}$ pseudorandom strings and derandomization in quadratic time. The reconstruction Rec gets D and the hash values as advice; whenever the list-decoder queries $\bar{D}$, then Rec computes $\bar{D}$ with queries to D in time $N^{2.01}$. See Section 4 for further details.

2.2 Proof of Theorem 1.2

Turning to $\mathcal{AM}$ protocols, let V be an $\mathcal{AMTIME}[T]$ verifier, and recall that we want to derandomize V in $\mathcal{NTIME}[n \cdot T^{1.01}]$. The approach above can still be used, but it yields derandomization in quadratic time rather than in time $n \cdot T^{1.01}$.

The main idea in the proof is to *compose the generator* $\mathsf{S} \circ \mathsf{Gen}$ *with itself*, using different hard truth-tables, to get a PRG with only $n^{1.01}$ seeds rather than $T^{1.01}$ seeds. In high-level, we first use $\mathsf{S} \circ \mathsf{Gen}^{f_1}$ with a long truth-table (of size $|f_1| = T^{1.01}$) to transform V into a verifier V' with running time $T^{1.01}$ that uses only $O(\log T)$ random coins, and then use $\mathsf{S} \circ \mathsf{Gen}^{f_2}$ with a short truth-table (of size $|f_2| = n^{1.01}$) to fully derandomize V', using a seed of length $(1.01) \cdot \log(n)$. [13] Details follow.

As a first step, we will need a refined version of $(\mathsf{Gen}, \mathsf{Rec})$. Recall that in Theorem 1.2 we are assuming hardness for probabilistic protocols. Following an idea from [DMO+20], the reconstruction can now compute $\bar{D}$ probabilistically rather than deterministically, and thus use only a small number queries to D rather than $N^{1.01}$; this reduces the running time to be close to T rather than to T^2. Furthermore, we observe that *the reconstruction* Rec *does not actually use the full power of its oracle access to* D: Loosely speaking, there is $\alpha \in (0,1)$ such that for a good non-deterministic guess, it suffices to "prove" to Rec that an α-fraction of Rec's queries are accepted by D; and simultaneously, for a bad non-deterministic guess, less than α-fraction of Rec's queries will be accepted by D. See Proposition 5.2 for precise details and a proof.

Now, when derandomizing $\mathcal{AM}$ with perfect completeness, the distinguisher D is a non-deterministic procedure testing whether there exists a satisfying witness for a given input and random coins.[14] A naive instantiation of Rec would thus yield an

[12]An alternative approach from these works is to allow $\bar{D}$ to use randomness, in which case it is only of size $O(N)$ and the derandomization runs in quadratic time. However, this requires assuming that f is hard for non-uniform $\mathcal{MA}$ circuits, an assumption we are trying to avoid.

[13]This approach follows an idea from [CT21b], wherein superfast derandomization was achieved by composing two PRGs in a similar way. However, in [CT21b] one of the PRGs relied on cryptographic assumptions, and the other was the Nisan-Wigderson [NW94] PRG. In contrast, in this work there are no cryptographic assumptions, and we simply compose two instantiations of $(\mathsf{Gen}, \mathsf{Rec})$.

[14]In works concerning derandomization of $\mathcal{AM}$ the output of D is often negated, and it is then thought of as co-nondeterministic circuit. This is done when considering hitting-set generators for D, for derandomizing protocols with perfect completeness. We avoid doing so, and as mentioned in Section 1 our results easily generalize to derandomization of protocols with imperfect completeness (see Remark 5.4).

$\mathcal{MA}^{\mathcal{NP}}$ machine, whereas we want to assume hardness only for $\mathcal{MAM}$. To do so we rely on the observations about Rec above: Our reconstruction algorithm will first receive a witness for Rec, then it will use randomness to choose a small number of queries (thereby reducing the running time to near-linear), and finally it will send the queries to the prover to obtain non-deterministic witnesses for D on these queries. This yields an $\mathcal{MAM}$ protocol, and we show that it indeed suffices for the reconstruction to work (for details see Proposition 5.2 and the proof of Proposition 5.3).

Now, denote the running time of the protocol V that we want to derandomize by $T(n) = n^k$. Recall that our derandomization will compose $\mathsf{S} \circ \mathsf{Gen}$ using a truth-table f_1 of length $T^{1.01}$ and another truth-table f_2 of length $n^{1.01}$. We assume that f_1 can be verified in near-linear time $T^{1.01}$ and is hard for $\mathcal{MAM}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$, and that f_2 can be verified in time $n^{k+.01}$ and is hard for $\mathcal{MAM}[2^{(1-\delta)k\cdot n}/2^{(1-\delta)\cdot n}]$. Thus, we can verify both in time $T^{1.01}$, and using the reconstruction argument above, both of them are pseudorandom for V, which runs in time T. Our derandomization enumerates over the output strings of $\mathsf{S} \circ \mathsf{Gen}^{f_2}$, and uses each string as a seed for $\mathsf{S} \circ \mathsf{Gen}^{f_1}$.[15] The resulting pseudorandom set has $n^{1.01}$ strings and can be computed in time $n^{1.01} \cdot T^{1.01} < n \cdot T^{1.02}$, which suffices for our derandomization of V.

To extend this result to $\mathcal{AM}$ protocols with constantly many rounds, we first simulate any constant-round protocol by a two-round protocol, and then apply the result above as a black-box. Indeed, the key observation is that the simulation overhead when using the classical result of [BM88] allows us to obtain tight results (under #NSETH), while only assuming hardness for $\mathcal{MAM}$ with advice; see Section 5.2.2 for details.

A relaxation of the hypothesis. As mentioned after the statement of Theorem 1.2, we further relax its hypothesis, by requiring that for $k > 1$, the property $\mathcal{L}_k$ will be useful only against $\mathcal{NTIME}$ machines with the specified time and advice complexity (rather than against $\mathcal{MAM}$ protocols of this complexity). Our approach for doing so is to replace the inner PRG $\mathsf{S} \circ \mathsf{Gen}^{f_2}$ with the *Nisan-Wigderson* generator [NW94]. This can be useful for us, because the NW generator is suitable for our superfast parameter setting when it is instantiated for a small output length (i.e., $|f|^\eta$ where $|f|$ is the truth-table length and $\eta \ll \epsilon$ is a sufficiently small constant; see Theorem 5.16), and $\mathsf{S} \circ \mathsf{Gen}^{f_1}$ reduces the number of random coins to $O(\log(T)) = O(\log(n))$.

The main obstacle towards applying the NW generator in this setting is that it requires hardness against machines with advice and (non-adaptive) *oracle access* to $\mathcal{NTIME}$ (i.e., to the distinguisher, which in this case is an $\mathcal{NTIME}$ machine), whereas we only assume hardness against $\mathcal{NTIME}$ *machines* with advice. To bridge this gap, we use an idea of Shaltiel and Umans [SU06] (following [FF93; SU05]), which allows to transform truth-tables that are hard for $\mathcal{NTIME}$ machines with advice into truth-tables that are hard for machines with advice and non-adaptive oracle access to $\mathcal{NTIME}$. We use their transformation while analyzing it in a more careful way, which allows us to bound the overheads in the hardness of the truth-table incurred by the transformation. (See Section 5.3.1 for further details.)

2.3 Proofs of the results from Section 1.3

We simulate probabilistic doubly efficient proof systems by deterministic doubly efficient argument systems using *non-black-box derandomization algorithms*, rather than

[15]Indeed, the length of output strings of $\mathsf{S} \circ \mathsf{Gen}^{f_2}$ is an "overkill", since they are of length close to n but we only use the first $O(\log(n))$ bits in them as a seed for $\mathsf{S} \circ \mathsf{Gen}^{f_1}$.

PRGs. Specifically, we will use a targeted pseudorandom generator, which takes an input z and prints a list of strings that looks pseudorandom to efficient algorithms that also have access to the same z.

Our targeted PRG will be reconstructive (i.e., based on a hard function), and will thus yield an "instance-wise" hardness-vs-randomness tradeoff: If the underlying function is hard to compute on a set S of inputs of density $1-\mu$, then the targeted PRG is pseudorandom on each and every input in S.[16] The specific generator that we will use, from [CT21a], is denoted G_f and relies on a function $f \colon \{0,1\}^* \to \{0,1\}^*$ with *multiple output bits* such that:

1. **(Upper bound.)** Individual output bits of the string $f(z)$ can be computed in time T'.

2. **(Lower bound.)** The entire string $f(z)$ cannot be approximately printed in time $T' \cdot |f(x)|^\beta$, for a small constant $\beta > 0$.[17]

Note that the obvious algorithm prints $f(z)$ in time $T' \cdot |f(z)|$, but f is *"non-batch-computable"*, in the sense that printing (an approximate version of) the entire string cannot be done in time close to that of computing a single bit (i.e., in time $T' \cdot |f(x)|^\beta$).

As a first step we will consider derandomizing verifiers that use a small number of coins, say $n^{o(1)}$. We instantiate G_f accordingly with $n^{o(1)}$ output bits, and with f and a time bound T' that is slightly larger than the running time of the verifier. In this setting G_f runs in time $T' \cdot n^{o(1)}$ and prints $n^{o(1)}$ random strings, and thus reduces the number of random coins (to $o(\log(n))$) without increasing the time complexity. To deduce that G_f is pseudorandom for an algorithm from a class $\mathcal{C}$ on input z, it suffices to assume that f is hard to approximately print by algorithms running in time $T' \cdot n^{o(1)}$ with oracle access to $\mathcal{C}$. (See Theorem 7.1 for a precise statement.)

Our goal will be to use G_f in order to replace the random coins of the verifier by pseudorandom coins. If we are able to reduce the number of coins to (say) $o(\log(n))$, then the verifier can just ask the prover in advance to send all possible $n^{o(1)}$ transcripts of interaction, and then check for consistency and compute the probability that it would have accepted (see, e.g., the proof of Theorem 7.5 for details). However, a naive application of G_f to the common input x to the protocol does not seem to suffice for this purpose: This is because in any round of interaction, when we replace the verifier's random coins by pseudorandom coins, the verifier's behavior depends not only on x but also on the messages sent by the prover. [18]

The main idea: Using transcripts as a source of hardness. The main idea that underlies our constructions is using the *transcript of the interaction in each round as a source of hardness* for G_f. That is, in each round of interaction we apply the targeted generator G_f with the current *transcript as input*, and obtain coins that the verifier can use in that specific round, given the previous interaction.

[16] To be more accurate, for every potential distinguisher M there exists a "reconstruction" algorithm F_M such that the following holds: On every z on which F_M fails to compute the hard function, the targeted PRG with input z is pseudorandom for M on input z.

[17] The meaning of "approximately print" here is as in Assumption 1.6: A probabilistic algorithm M approximately prints $f(x)$ with error δ if $\Pr[M(x)_i = f(x_i)] \geq 1-\delta$, where the probability is over the random coins of M and an output index. For simplicity, we think of δ as a small constant for now.

[18] Another way to see this is to think of a worst-case verifier that *tries* to distinguish the pseudorandom strings from random ones. In this case, the prover's messages can be thought of as supplying this worst-case verifier with non-uniform advice.

Recall, however, that we do not assume that f is hard on all inputs, and thus an all-powerful prover could potentially find transcripts on which f is easy (in which case G_f is not guaranteed to be pseudorandom). This is where our relaxation of the soundness condition comes in: Since we are only interested in soundness with respect to polynomial-time provers and polynomial-time samplable inputs, *we can think of the transcript as an input (to G_f) sampled from a polynomial-time samplable distribution*. By our assumption f cannot be computed with non-neglibile probability over inputs chosen from *any* polynomial-time samplable distribution, and thus for all but a negligible fraction of transcripts G_f will be pseudorandom.

We stress that the function f is hard for algorithms running in fixed polynomial time $T' \cdot n^{o(1)}$ (this models the verifier), but its hardness holds over inputs (i.e., transcripts) chosen according to any polynomial-time samplable distribution, where the latter polynomial may be arbitrarily large (this models an input chosen from a polynomial-time samplable distribution and the prover's corresponding messages).

Materializing this approach turns out to be considerably more subtle than it might seem. We thus include a self-contained proof for an easy "warm-up" case, namely that of derandomizing doubly efficient proof systems with one prover message (similar to $\mathcal{MA}$). In this setting, if we are willing to assume that one-way functions exist, then we can reduce the number of random coins to be n^{ϵ} (for an arbitrarily small $\epsilon > 0$) and carry out the strategy above quite easily. See Section 7.1 for details.

The key step: Proof of Theorem 1.7. Let us start with Theorem 1.7, in which we derandomize protocols with c rounds in which the verifier uses $n^{o(1)}$ random coins, under the assumption that f is hard to approximately print for algorithms with oracle access to $\mathcal{AM}$ protocols with c rounds. Recall that we are interested in algorithms with perfect completeness, and thus when replacing random coins by pseudorandom ones we only need to argue that soundness is maintained. (For protocols with few random coins this can be assumed without loss of generality; see Remark 7.6.)

In each round we feed the current transcript to G_f and thus reduce the number of coins to $o(\log(n))$. Naturally, we want to use a hybrid argument, and claim that if in any round the residual acceptance probability of the verifier (when considering the continuation of the interaction after this round) significantly increased,[19] then we can compute the hard function f with the transcript at that round as input. Note that our "distinguisher" for G_f in this case is the function that computes the acceptance probability of the residual verifier after fixing the current transcript.

When this interaction happens with an all-powerful prover, the distinguisher for G_f can be computed by an $\mathcal{AM}$ protocol with at most c turns, and we can indeed use our hardness hypothesis. However, when we consider the acceptance probability with respect to efficient provers, then the distinguisher will be an argument system (rather than an $\mathcal{AM}$ protocol). This is a challenge for us (rather than an advantage), because we want to use this distinguisher to contradict the hardness of f – but the class of argument systems is broader than $\mathcal{AM}$, so this will necessitate using a stronger hardness hypothesis (i.e., against algorithms with oracle access to argument systems).

To handle this challenge we use a careful hybrid argument: In each round we replace not only random coins by pseudorandom ones, but also gradually replace all-powerful prover strategies by efficient prover strategies; that is, in the i^{th} hybrid we

[19]By "acceptance probability" here we mean the maximal probability that the verifier accepts, when considering interaction with a prover (in the relevant class of provers) that maximizes this probability. In some sources this is referred to as the value of the game/protocol.

replace the random coins in the i^{th} round by pseudorandom coins, and replace the all-powerful prover strategy in the i^{th} round by an efficient strategy. Assuming that the common input x is a NO instance, the first hybrid (with random coins and an all-powerful prover) has low acceptance probability, and we are interested in analyzing the case where the last hybrid (with pseudorandom coins and an efficient prover) has higher acceptance probability, in which case there is a noticeable increase in the acceptance probability between a pair of hybrids, say in the i^{th} hybrid.

In our analysis, we now further replace the efficient prover strategy in the i^{th} round by an all-powerful prover strategy; crucially, this only causes an *increase* in the acceptance probability gap.[20] At this point the "residual protocol" in both distributions that were obtained in the i^{th} hybrid uses an all-powerful prover, and (with some work) we can show that there exists a distinguisher for G_f (with the transcript as input) that is an $\mathcal{AM}$ protocol with c rounds. (For further details see the part titled "Obtaining an $\mathcal{AMTIME}^{[\rightleftharpoons c]}$ distinguisher" in the proof of Claim 7.5.2 for details.)

The general case: Proof of Theorem 1.8. The argument above works under the assumption that the protocol uses $n^{o(1)}$ coins, and we now want to extend it to general protocols. To do so we use an additional assumption, namely that there exist functions whose truth-tables can be verified in near-linear time, but that are hard for $\mathcal{AM}$ protocols with $c+1$ rounds that run in time $2^{(1-\delta)\cdot n}$ and use $2^{(1-\delta)\cdot n}$ bits of non-uniform advice. We will use this additional assumption to reduce the number of random coins in each round from $\text{poly}(n)$ to $O(\log(n))$, using the reconstructive PRG $(\mathsf{Gen}, \mathsf{Rec})$, and then invoke Theorem 1.7 as a black-box.

The argument here follows in the same spirit as the one above, first replacing the coins in all rounds simultaneously and then using a careful hybrid argument (and a reconstruction argument) to obtain a contradiction. To support the setting that is obtained via the hybrid argument, we refine the reconstructive PRG $(\mathsf{Gen}, \mathsf{Rec})$ yet again, this time showing that it works when the distinguisher is any function that agrees with a certain promise problem, where the advice to the reconstruction algorithm depends only on the promise problem rather than on the function. (See Section 7.3.1.)

2.4 Proof of Theorem 1.3

Finally, we briefly explain the ideas in the proof of Theorem 1.3. The most important change, compared to the proofs of Theorem 1.1 and Theorem 1.2, is that since now we are only assuming hardness against *uniform* protocols (our hardness assumption is that $L \notin \mathtt{i.o.}(\mathcal{MA} \cap co\mathcal{MA})\mathcal{TIME}^{[\rightleftharpoons 7]}[2^{(1-\delta)\cdot n}]$), we need to make the reconstruction algorithm uniform as well.

We will use yet another refinement of $(\mathsf{Gen}, \mathsf{Rec})$ above. Recall that Rec (after our last refinement) gets the "right" advice $\mathtt{adv}$, and can then compute the function f if the distinguisher "proves" to the verifier that sufficiently many queries are 1-instances. Following [MV05; GSTS03; SU07], we strengthen Rec so that it meets a $\mathsf{resiliency}$ condition: Namely, using a small number of rounds of interaction, Rec is able to get any prover to send advice $\mathtt{adv}$ and *commit* to a single truth-table $f_{\mathtt{adv}}$ specified by the advice.[21] That is, given $\mathtt{adv}$ and a few rounds of interaction, there is a single $f_{\mathtt{adv}}$ such

[20]This is the place in the proof where we use the fact that the protocol has perfect completeness (i.e., we are only arguing that soundness is maintained given pseudorandom coins).

[21]To see the challenge, assume that $\mathtt{adv}$ is a non-deterministic circuit that supposedly computes the truth-table. In this case, different non-deterministic guesses could yield different truth-tables.

that Rec answers according to it. Working carefully, we are able to make Rec resilient without significant time overhead (see Definition 5.1 and Proposition 5.2).

If our reconstruction could be convinced that $f_{\mathtt{adv}}$ to which the prover committed is the "right" truth-table (i.e., $f_{\mathtt{adv}} = f$), then on query x it could simply output $f_{\mathtt{adv}}(x)$. But what if the prover committed to a truth-table different than f? To resolve this issue we initially encode f using a *highly efficient PCP of proximity* (i.e., the one of [BGH+05]), which then allows us to locally test the encoded string without incurring significant time overheads and deduce that $f_{\mathtt{adv}}$ is close to f, otherwise we reject (see Theorem 3.16); we combine this with local error correction, to compute f given any truth-table that is close to f. The price that we pay for using the PCPP is that the truth-table (which is now a PCPP witness of the original truth-table) is necessarily of size $T^{1.01}$ (i.e., slightly larger than the time complexity of the function), and thus our PRG uses $(1.01) \cdot \log(T)$ seeds and we get derandomization in quadratic time $T^{2.01}$ rather than in time $n \cdot T^{1.01}$.

3 Preliminaries

Throughout the paper, we will typically denote random variables by boldface, and will denote the uniform distribution over $\{0,1\}^n$ by $\mathbf{u}_n$ and the uniform distribution over a set $[n]$ by $\mathbf{u}_{[n]}$. Recall the following standard definition of a distinguisher for a distribution $\mathbf{w}$, by which we (implicitly) mean a distinguisher between $\mathbf{w}$ and the uniform distribution.

Definition 3.1 (distinguisher). *We say that a function $D \colon \{0,1\}^n \to \{0,1\}$ is an ϵ-distinguisher for a distribution $\mathbf{w}$ over $\{0,1\}^n$ if $\Pr[D(\mathbf{w}) = 1] \notin \Pr[D(\mathbf{u}_n) = 1] \pm \epsilon$. We say that D is an (α, β)-distinguisher if $\Pr[D(\mathbf{w}) = 1] \geq \alpha$ and $\Pr[D(\mathbf{u}_n) = 1] \leq \beta$.*

We also fix a standard notion of "nice" time bounds for complexity classes, where we are only concerned of time bounds that are not sub-linear.

Definition 3.2 (time bound). *We say that $T \colon \mathbb{N} \to \mathbb{N}$ is a time bound if T is time-computable and non-decreasing, and for every $n \in \mathbb{N}$ we have that $T(n) \geq n$.*

Recall that $pr\mathcal{MATIME}[T]$ denotes the class of *promise problems* (rather than languages) that can be solved by $\mathcal{MA}$ protocols with a verifier running in time T. Derandomization of $pr\mathcal{MATIME}[T]$ as in the conclusions of Theorems 1.1 and 1.2 is stronger than derandomization of $\mathcal{MATIME}[T]$.

3.1 Useful properties

Following Razborov and Rudich [RR97], we now define useful properties, which are sets of truth-tables that can be efficiently recognized and that describe functions hard to compute in a certain class $\mathcal{C}$. Our definition is a bit more careful than usual, since we are interested in the case where $\mathcal{C}$ is a class decidable by uniform machines that gets non-uniform advice (rather than a class decidable by non-uniform circuits).

Definition 3.3 (useful property). *Let $\mathcal{L} \subseteq \{0,1\}^*$ be a collection of strings such that every $f \in \mathcal{L}$ is of length that is a power of two, and let $\mathcal{C}$ be a class of languages. We say that $\mathcal{L}$ is a $\mathcal{C}'$-constructive property useful against $\mathcal{C}$ if the following two conditions hold:*

1. *(Non-triviality.) For every $N = 2^n$ it holds that $\mathcal{L}_n = \mathcal{L} \cap \{0,1\}^N \neq \emptyset$.*

2. *(Constructivity.)* $\mathcal{L} \in \mathcal{C}'$.

3. *(Usefulness.) For every $L \in \mathcal{C}$ and every sufficiently large $n \in \mathbb{N}$ it holds that $L_n \notin \mathcal{L}_n$, where $L_n \in \{0,1\}^{2^n}$ is the truth-table of L on n-bit inputs.*

To clarify the meaning of the "usefulness" condition above, let us consider the case where $\mathcal{C}$ is a class of Turing machines with advice of bounded length. In this case, the condition asserts that for every fixed machine M and infinite sequence `adv` of advice strings, and every sufficiently large $n \in \mathbb{N}$, the machine M with advice `adv` fails to compute any truth-table in $\mathcal{L}_n$.

Since each string in $\mathcal{L}$ is of length 2^n for some $n \in \mathbb{N}$, and we think of it as a truth-table of a function over n bits, we will usually denote the input length to $\mathcal{L}$ as $N = 2^n$. For example, when we refer to an $\mathcal{NTIME}[N^2]$-constructive property useful against $\mathcal{NTIME}[2^{1.99 \cdot n}]$ we mean that 2^n-length truth-tables in $\mathcal{L}$ can be recognized in non-deterministic time 2^{2n}, but that the corresponding n-bit functions cannot be computed in non-deterministic time $2^{1.99 \cdot n}$.

3.2 Proof systems

The following definition refers to non-deterministic unambiguous computation, which captures non-deterministic computation of both the language and its complement:

Definition 3.4 (non-deterministic unambiguous computation)**.** *We say that a machine M is* non-deterministic and unambiguous *if for every $x \in \{0,1\}^*$ there exists a value $L(x) \in \{0,1\}$ such that the following holds:*

1. *There exists a non-deterministic guess π such that $M(x,\pi) = L(x)$.*

2. *For every non-deterministic guess π' it holds that $M(x,\pi') \in \{L(x), \perp\}$.*

3.2.1 Arthur-Merlin proof systems

Let us now recall the definition of Arthur-Merlin proof systems (i.e., of $\mathcal{AM}$). Since we will be concerned with precise time bounds, we also specify the precise structure of the interaction, as follows.

Definition 3.5 (Arthur-Merlin proof systems)**.** *We say that $L \in \mathcal{AMTIME}^{[\rightleftharpoons c]}[T]$ if there is a proof system in which on a shared input $x \in \{0,1\}^*$, a verifier interacts with a prover, taking* turns *in sending each other information, such that the following holds.*

- **Public coins:** *Whenever the verifier sends information to the prover, that information is just uniformly chosen bits.*

- **Structure of the interaction:** *The number of turns is c, and we always assume that the first turn is the verifier sending random bits to the prover.*

- **Running time:** *The number of bits that are sent in each turn is exactly $T(|x|)$, and in the end the verifier performs a deterministic linear-time computation on the transcript (which is of length $c \cdot T(|x|) = O(T(|x|))$ and outputs a single bit.*

- **Completeness and soundness:** *For every $x \in L$ there exists a prover such that the verifier accepts with probability 1; for every $x \notin L$ and every prover, the verifier rejects, with probability at least 2/3.*

Furthermore, if the verifier sends at most $R(n)$ random coins in each round, then we say that $L \in \mathcal{AMTIME}^{[\rightleftharpoons c]}[T, R]$. When L can be decided by an interaction as above in which the prover *takes the first turn, we say that $L \in \mathcal{MATIME}^{[\rightleftharpoons c]}[T]$. In all the definitions above, when omitting the number of messages c, we mean that $c = 2$.*

Following standard conventions, we will sometimes refer to the verifier as Arthur and to the prover as Merlin. Note that when c is odd (meaning that the last turn is the verifier's), in the last turn the verifier does not need to send any random bits to the verifier, but may run a randomized linear-time computation on the transcript rather than a deterministic one.[22]

When we want to refer to proof systems with *imperfect completeness* (i.e., for any $x \in L$ there is a prover such that the verifier accepts with probability at least $2/3$), we explicitly add a subscript "$_2$", for example $\mathcal{AMTIME}_2^{[\rightleftharpoons c]}[T]$ or $(\mathcal{MA} \cap co\mathcal{MA})\mathcal{TIME}_2^{[\rightleftharpoons c]}[T]$.

3.2.2 Doubly efficient proof systems and deterministic argument systems

As mentioned in Section 1.3, our derandomization results will apply to proof systems in which the honest prover is efficient. We define this notion as follows:

Definition 3.6 (doubly efficient proof systems)**.** *We say that $L \in \mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T]$ if it meets all the conditions in Definition 3.5, and in addition meets the following condition: There exists a deterministic algorithm P running in time polynomial in T such that for every $x \in L$, when the verifier interacts with P on common input x, it accepts with probability* 1. *We define $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T, R]$ analogously.*

Note that Definition 3.6 focuses on doubly efficient proof systems with *public coins*; the known constructions of doubly efficient proof systems all use public coins (see [GKR15; RRR21; GR18; Gol18]). In addition, one could use a broader definition that allows the honest prover to run in time $\bar{T} \gg T$ that is not necessarily polynomial in T; in this work the narrower definition will suffice for us.

We will also be interested in a natural subclass of doubly efficient proof systems, which we now define. Loosely speaking, we say that a system has an efficient universal prover if for any partial transcript, the maximum acceptance probability of the residual protocol across all provers is (approximately) attained by an *efficient* prover. That is:

Definition 3.7 (universal provers)**.** *Let $L \in \{0,1\}^*$ and let V be a verifier in a proof system for L. For $\mu \colon \mathbb{N} \to [0, 1)$, we say that the proof system has a μ-*approximate universal prover with running time *$\bar{T}$ if there exists an algorithm P that on any input x and π, where π is a partial transcript for the proof system, runs in time $\bar{T}(|x|)$, and satisfies that*

$$\Pr\left[\langle V, P\rangle(x, \pi) = 1\right] > \max_{\bar{P}} \left\{\Pr\left[\langle V, \bar{P}\rangle(x, \pi) = 1\right]\right\} - \mu(|x|),$$

where the notation $\langle V, P\rangle(x, \pi)$ denotes the outcome of interaction of V and P on input x and when the first part of the transcript is fixed to π, and the maximum on the RHS is over all prover strategies (regardless of their efficiency).

Note that given $x \in L$, the universal prover acts as an honest prover that convinces the verifier to accept with probability at least $2/3 - \mu$ (or $1 - \mu$, if the protocol admits perfect completeness). In particular, a proof system with an efficient universal prover

[22]This is equivalent to the definition above, in which the verifier sends random coins in the last turn then runs a deterministic linear-time computation on the transcript (which includes these random coins).

is a doubly efficient proof system. However, we do not think of the universal prover as an honest prover, since given $x \notin L$ or a dishonest partial transcript, the universal prover still tries to maximize the acceptance probability of the verifier.

A well-known doubly efficient proof system that has a universal prover is the *sumcheck protocol*; see Theorem 3.17 for details.

Let us now recall Definition 1.5 of deterministic doubly efficient argument systems and discuss a few definitional issues.

Definition 3.8 (deterministic doubly efficient argument system; Definition 1.5, restated). *We say that $L \subseteq \{0,1\}^*$ is in $\mathsf{de}\mathcal{DARG}[T]$ if there exists a deterministic T-time verifier V such that the following holds:*

1. *There exists a deterministic algorithm P that, when given $x \in L$, runs in time $\mathrm{poly}(T)$ and outputs π such that $V(x, \pi) = 1$.*

2. *For every polynomial p, and every probabilistic algorithm $\tilde{P}$ running in time $p(T)$, and every sufficiently large $n \in \mathbb{N}$, the probability that $\tilde{P}(1^n)$ prints $x \notin L$ and $\pi \in \{0,1\}^T$ such that $V(x, \pi) = 1$ is $T(n)^{-\omega(1)}$.*

Observe that $\mathsf{de}\mathcal{DARG}[T] \subseteq \mathcal{DTIME}[\mathrm{poly}(T)]$, since the honest prover runs in time $\mathrm{poly}(T)$. Removing the efficiency restriction on the honest prover makes the definition too broad to be meaningful: Under a plausible hardness assumption, *every* language (regardless of is complexity) has a proof system as above in which the honest prover P runs in time larger than that of the adversaries $\tilde{P}$. [23]

Thus, for the definition to be meaningful we need to have $T = T_V < T_P < T_{\tilde{P}}$, where the three latter notations represent the running times of the verifier V, of the honest prover P, and of the adversaries $\tilde{P}$, respectively. While Definition 3.8 couples these bounds so that they are all polynomially related, a broader definition in which the gaps are super-polynomial still makes sense. We use the narrower definition only because it suffices for our purposes in the current work.

3.3 Error-correcting codes

We recall the definition of locally list-decodable codes, and state a standard construction that we will use.

Definition 3.9 (locally list decodable codes). *We say that $\mathsf{Enc}\colon \Sigma^N \to \Sigma^M$ is* locally list-decodable from agreement ρ in time t and with output-list size L *if there exists a randomized oracle machine $\mathsf{Dec}\colon [N] \times [L] \to \Sigma$ running in time t such that the following holds. For every $z \in \Sigma^M$ that satisfies $\Pr_{i \in [M]}[z_i = \mathsf{Enc}(x)_i] \geq \rho$ for some $x \in \Sigma^N$ there exists $a \in [L]$ such that for every $i \in [N]$ we have that $\Pr[\mathsf{Dec}^z(i, a) = x_i] \geq 2/3$, where the probability is over the internal randomness of Dec.*

Theorem 3.10 (a locally list-decodable code, see [STV01]). *For every constant $\eta > 0$ there exists a constant $\eta' > 0$ such that the following holds. For every $m \in \mathbb{N}$ and $\rho = \rho(m)$ there exists a code $\mathsf{Enc}\colon \{0,1\}^m \to \Sigma^{\bar{m}}$, where $|\Sigma| = O(m^{\eta'}/\rho^2)$ and $\bar{m} = O_{\eta'}(m/\rho^{2/\eta'})$, such that:*

1. *The code is computable in time $\tilde{O}(\bar{m} \cdot \log(|\Sigma|)) = \tilde{O}(m/\rho^{2/\eta'})$.*

[23]To see this, assume that there exists a relation $R = \{(x, \pi)\}$ that can be decided in time T, but every probabilistic algorithm $\tilde{P}$ getting input x and running in time $\mathrm{poly}(T)$ fails to find π such that $(x, \pi) \in R$, except with negligible probability (over x and over the random coins of $\tilde{P}$). Then, for every L, define V that accepts (x, π) iff $(x, \pi) \in R$, and observe that this verifier meets the relaxed definition of $\mathsf{de}\mathcal{DARG}$.

2. *The code is locally list-decodable from agreement ρ in time $m^{\eta} \cdot (1/\rho)^{1/\eta'}$ and with output list size $O(1/\rho)$. Furthermore, the local decoder issues its queries in parallel, as a function of the randomness and the input.*

We also need the following two uniquely decodable codes:

Theorem 3.11 (uniquely decodable code, see e.g., [GLR+91] and [AB09, Section 19.4])**.** *For every constant $\eta > 0$ the following holds. For every $m \in \mathbb{N}$ there exists a code $\mathsf{Enc}\colon \{0,1\}^m \to \{0,1\}^{\bar{m}}$, where $\bar{m} = \tilde{O}(m)$, such that:*

1. *The code is computable in time $\tilde{O}(\bar{m}) = \tilde{O}(m)$.*

2. *The code is locally decodable from agreement 0.9 with decoding circuit size m^{η}. Furthermore, the decoding circuit issues its queries in parallel and outputs correctly with probability at least $1 - 1/m$.*

Lemma 3.12 (unique decoding for low-degree univariate polynomials; see [WB86])**.** *Let q be a prime power. Given t pairs (x_i, y_i) of elements of $\mathbb{F}_q$, there is at most one polynomial $g\colon \mathbb{F}_q \to \mathbb{F}_q$ of degree at most u for which $g(x_i) = y_i$ for more than $(t+u)/2$ pairs. Furthermore, there is a polynomial time algorithm that finds g or report that g does not exist.*

3.4 Near-linear time constructions: Extractors, hash functions, cryptographic PRGs

In this section we state several known algorithms that we will use in our proofs and that run in near-linear time. The first is a pairwise-independent hash function based on convolution hashing [MNT93].

Theorem 3.13 (a quasilinear-time computable pairwise independent hash function; for proof see, e.g., [CT21b, Theorem 3.12])**.** *For every $m, m' \in \mathbb{N}$ there exists a family $\mathcal{H} \subseteq \left\{\{0,1\}^m \to \{0,1\}^{m'}\right\}$ of quasilinear-time computable functions such that for every distinct $x, x' \in \{0,1\}^m$ it holds that $\Pr_{h \in \mathcal{H}}[h(x) = h(x')] \leq 2^{-m'}$.*

The second construction is of a seeded randomness extractor that runs in linear time, which was presented by Doron *et al.* [DMO+20] following [TSZS06, Theorem 5].

Theorem 3.14 (a linear-time computable extractor, see [DMO+20])**.** *There exists $c \geq 1$ such that for every $\gamma < 1/2$ the following holds. There exists a strong oblivious (δ, ϵ)-sampler $\mathsf{Samp}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ for $\delta = 2^{n^{1-\gamma} - n}$ and $\epsilon \geq c \cdot n^{-1/2+\gamma}$ and $d \leq (1 + c \cdot \gamma) \cdot \log(n) + c \cdot \log(1/\epsilon)$ and $m = \frac{1}{c} \cdot n^{1-2\gamma}$ that is computable in linear time.*

The third construction is that of a cryptographic PRG that works in near-linear time. Such a PRG can be obtained assuming standard one-way functions, by starting with a fast PRG that has small stretch and then applying standard techniques for extending the output length.

Theorem 3.15 (OWFs yield PRGs with near-linear running time; see [CT21a, Theorem 3.4])**.** *If there exists a polynomial-time computable one-way function secure against polynomial-time algorithms, then for every $\epsilon > 0$ there exists a PRG that has seed length $\ell(n) = n^{\epsilon}$, is computable in time $n^{1+\epsilon}$, and fools every polynomial-time algorithm with negligible error. Moreover, if the one-way function is secure against polynomial-sized circuits, then the PRG fools every polynomial-sized circuit.*

3.5 Pair languages and PCPPs

A pair language L is a subset of $\{0,1\}^* \times \{0,1\}^*$. We say that $L \in \mathcal{NTIME}[T]$ if $L(x,y)$ can be computed by a nondeterministic algorithm in time $T(|x|+|y|)$. We also say L has stretch K for a function $K\colon \mathbb{N} \to \mathbb{N}$, if for all $(x,y) \in L$, it holds that $|y| = K(|x|)$. We will use the following PCP of proximity for pair languages by Ben-Sasson *et al.* [BGH+05]:

Theorem 3.16 (PCPP with short witnesses [BGH+05]). *Let $K\colon \mathbb{N} \to \mathbb{N}$ such that $K(n) \geq n$ for all $n \in \mathbb{N}$. Suppose that L is a pair language in $\mathcal{DTIME}[T]$ for some non-decreasing function $T\colon \mathbb{N} \to \mathbb{N}$ such that L has stretch K. There is a verifier V and an algorithm A such that for every $x \in \{0,1\}^n$, denoting $T = T(n+K(n))$ and $K = K(n)$:*

1. ***(Efficiency.)** When V is given input x and oracle access to $y \in \{0,1\}^K$ and to a proof $\pi \in \{0,1\}^{2^r}$, where $r \leq \log(T) + O(\log\log(T))$, the verifier V uses r bits of randomness, makes at most* $\mathrm{polylog}(T)$ non-adaptive *queries to both y and π, and runs in time* $\mathrm{poly}(n, \log(K), \log(T))$.

2. ***(Completeness.)** Let M be an $O(T)$-time nondeterministic machine that decides L. That is, $(x,y) \in L$ if and only if there exists $w \in \{0,1\}^{O(T)}$ such that $M((x,y),w) = 1$. For every $y \in \{0,1\}^K$ and $w \in \{0,1\}^{O(T)}$ such that $M((x,y),w) = 1$, the algorithm $A(x,y,w)$ runs in time $\tilde{O}(T)$ and outputs a proof $\pi \in \{0,1\}^{2^r}$ such that*

$$\Pr\left[V^{y,\pi}(x, \mathbf{u}_r) = 1\right] = 1\,.$$

3. ***(Soundness.)** Let $Z = \left\{z \in \{0,1\}^K : (x,z) \in L\right\}$. Then, for every $y \in \{0,1\}^K$ that has Hamming distance at least $K/\log(T)$ from every $z \in Z$ and every $\pi \in \{0,1\}^{2^r}$,*

$$\Pr\left[V^{y,\pi}(x, \mathbf{u}_r) = 1\right] \leq 1/3\,.$$

3.6 Constant-round sumcheck and #NSETH

The following result is an "asymmetric" version of Williams' [Wil16] adaptation of the sumcheck protocol [LFK+92] into a constant-round protocol for counting the number of satisfying assignments of a given formula. Specifically, while in [Wil16] the n variables are partitioned into symmetric subsets and each sumcheck round is a summation over one of the subsets, here we partition the variables such that the first set is smaller, and in the corresponding sumcheck protocl the first round is shorter.

Theorem 3.17 (a constant-round protocol for counting satisfying assignments of a formula). *Let $k \in \mathbb{N}$, $\delta \in (0,1)$, and $\gamma = \frac{1-\delta}{k}$ be constants. For any $s(n) \leq 2^{o(n)}$, there is an $\mathcal{MATIME}^{[\rightleftharpoons 2k]}[2^{\max(\delta,\gamma)\cdot n + o(n)}]$ protocol Π that gets as input a formula $C\colon \{0,1\}^n \to \{0,1\}$ of size $s(n)$ and (with probability 1) outputs the number of satisfying assignments of C, such that the first prover message has length at most $2^{\delta n + o(n)}$, and the rest $k-1$ prover messages have length at most $2^{\gamma n + o(n)}$.*

Moreover, the protocol has the following two properties:

1. *After the prover sends its first message, the maximum acceptance probability of the subsequent protocol is either 1 or at most $1/3$.*

2. *There is a $1/n$-approximate universal prover running in $2^{O(n)}$ for Π.*[24]

[24]For the definition of a universal prover, see Definition 3.7.

Proof. The protocol is essentially identical to that of [Wil16, Theorem 3.4]. The only difference is that in [Wil16, Theorem 3.4] the n input variables are partitioned into $k+1$ blocks, each of length $n/(k+1)$, whereas we will partition them into a single block of length $\delta \cdot n$ and k other blocks of length $\gamma \cdot n$. Due to this fact, and since we are also claiming additional properties in the "moreover" part that were not stated in [Wil16], we include a complete proof.

Let $s(n) \leq 2^{o(n)}$, and let $C\colon \{0,1\}^n \to \{0,1\}$ be a formula of size $s(n)$. The prover and verifier will work with a prime $p \in (2^n, 2^{n+1}]$.[25] Let $P\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be the arithmetic circuit constructed by the standard arithmetization of the Boolean circuit C (see the proof of [Wil16, Theorem 3.3] for details) such that P has size $\mathrm{poly}(s(n)) \leq 2^{o(n)}$ and degree at most $\mathrm{poly}(s(n)) \leq 2^{o(n)}$.

For simplicity, we assume that $\delta \cdot n$ and $\gamma \cdot n$ are integers. We partition the n variables $x_1, \ldots, x_n$ into $k+1$ blocks $S_1, \ldots, S_{k+1}$, such that $|S_1| = \delta \cdot n$ and $|S_i| = \gamma \cdot n$ for every $i \geq 2$. For each $i \in [k+1]$, via interpolation similar to the proof of [Wil16, Theorem 3.4], we define a degree-$2^{|S_i|}$ polynomial $\Phi_i\colon \mathbb{F}_q \to \mathbb{F}_q^{|S_i|}$ such that for every $j \in \{0,1,\ldots,2^{|S_i|}-1\}$, $(\Phi_i(j))_\ell$ is the ℓ-th bit in the $|S_i|$-bit binary representation of j . Using a fast interpolation algorithm (see [Wil16, Theorem 2.2]), the polynomial Φ_i (i.e., the list of the coefficients of $|S_i|$ univariate polynomials, each corresponding to one of Φ_i's output values) can be constructed in $2^{|S_i|} \cdot \mathrm{poly}(n)$ time.

The protocol Π is specified as follows:

- In the first round, the honest prover computes the coefficients of the polynomial

$$Q_1(y) = \sum_{\substack{j_2,\ldots,j_{k+1} \\ \in\{0,1,\ldots,2^{\gamma \cdot n}-1\}}} P(\Phi_1(y), \Phi_2(j_2), \ldots, \Phi_{k+1}(j_k)) \tag{3.1}$$

  via interpolation, and sends Q_1 to the verifier (Note that Q_1 has degree $2^{o(n)+\delta \cdot n}$, so this message has size $2^{o(n)+\delta\cdot n}$). The verifier then chooses $r_1 \in \mathbb{F}_q$ uniformly at random and sends it to the prover.

- In the t-th round for $t \in \{2,3,\ldots,k\}$, the honest prover sends the coefficients of the degree-$2^{o(n)+\gamma\cdot n}$ polynomial

$$Q_t(y) = \sum_{\substack{j_{t+1},\ldots,j_{k+1} \\ \in\{0,1,\ldots,2^{\gamma \cdot n}-1\}}} P(\Phi_1(r_1), \ldots, \Phi_{t-1}(r_{t-1}), \Phi_t(y), \Phi_{t+1}(j_{t+1}), \ldots, \Phi_{k+1}(j_{k+1})) \tag{3.2}$$

  to the verifier. The verifier first checks if

$$\sum_{j_t \in \{0,1,\ldots,2^{\gamma\cdot n}-1\}} Q_t(j_t) = Q_{t-1}(r_{t-1}), \tag{3.3}$$

  and rejects immediately if the equality does not hold. The verifier then picks $r_t \in \mathbb{F}_q$ uniformly at random. The verifier then picks $r_\tau \in \mathbb{F}_q$ uniformly at random, and sends it to the prover if $\tau < k$

[25]The honest prover can find the smallest prime p that is larger than 2^n, which is at most 2^{n+1} by Bertrand's postulate, in time $2^{O(n)}$ and send p to the verifier. The verifier checks that the received number lies in $(2^n, 2^{n+1}]$ and is a prime, in deterministic time $\mathrm{poly}(n)$ (e.g., using [AKS04; AKS19]).

- Finally, at the end of the k-th round, the verifier checks if

$$Q_k(r_k) = \sum_{j_{k+1} \in \{0,1,\ldots,2^{\gamma \cdot n}-1\}} P((\Phi_1(r_1), \ldots, \Phi_k(r_k), \Phi_{k+1}(j_{k+1})),$$

and rejects immediately if the equality does not hold. Otherwise, it outputs

$$\sum_{j_1 \in \{0,1,\ldots,2^{\delta \cdot n}-1\}} Q_1(j_1) \ .$$

The upper bound on message lengths and the running time of the verifier can be verified directly from the protocol above. The analysis establishing the completeness and soundness of this protocol, as well as the upper bound on the complexity of the honest prover, follow a standard analysis of the sumcheck protocol; see [Wil16, Theorem 3.4]. We therefore focus on establishing the moreover part.

To see the "moreover" part, for every $i \in [k]$, let D_i be the degree of the polynomial Q_i. Fix a partial transcript π for all the interaction before the t-th prover message, and without loss of generality assume that $\pi = (\tilde{Q}_1, r_1, \ldots, \tilde{Q}_{t-1}, r_{t-1})$ (if $t = 1$ then π is empty). We prove the following claim.

Claim 3.18. *Given a partial transcript $\pi = (\tilde{Q}_1, r_1, \ldots, \tilde{Q}_{t-1}, r_{t-1})$, if $t = 1$ or $\tilde{Q}_{t-1}(r_{t-1}) = Q_{t-1}(r_{t-1})$, then the maximum acceptance probability of the subsequent protocol is* 1*, and can be achieved by a $2^{O(n)}$-time prover. Otherwise, it is at most $(k+1-t)/n^2$.*

Proof. We prove the claim by an induction on t, starting from $k+1$ and moving downward to 1. In the base case $t = k+1$ all messages have been sent, and the verifier accepts if and only $\tilde{Q}_{t-1}(r_{t-1}) = Q_{t-1}(r_{t-1})$. Hence the claim holds immediately.

Now, for $t \in [k]$, assuming the claim holds for $t+1$. If $t = 1$ or $\tilde{Q}_{t-1}(r_{t-1}) = Q_{t-1}(r_{t-1})$, then the prover simply sends the correct polynomial Q_t defined by (3.1) or (3.2) and proceeds as the honest prover (note that the check (3.3) will pass).

Otherwise, we have that $t \geq 2$ and $\tilde{Q}_{t-1}(r_{t-1}) \neq Q_{t-1}(r_{t-1})$. In this case, in order to not be rejected immediately, the prover has to send a polynomial $\tilde{Q}_t$ such that $\sum_{j_t \in \{0,1,\ldots,2^{\gamma \cdot n}-1\}} \tilde{Q}_t(j_t) = \tilde{Q}_{t-1}(r_{t-1})$. In particular, it means that $\tilde{Q}_t \neq Q_t$, where Q_t is defined by (3.2). Therefore, with probability at most $D_t/q \leq 1/n^2$ over the choice of r_t, it holds that $\tilde{Q}_t(r_t) = Q_t(r_t)$. By the induction hypothesis, we know that the maximum acceptance probability is at most $1/n^2 + (k-t)/n^2 \leq (k+1-t)/n^2$. □

The existence of a $1/n$-approximate universal prover with running $2^{O(n)}$ (the second item of the "moreover" part) follows immediately from Claim 3.18.

To see the first item, we note that if $\tilde{Q}_1 = Q_1$, where Q_1 is defined by (3.1), then the maximum acceptance probability is 1, by Claim 3.18. If $\tilde{Q}_1 \neq Q_1$, then since both of them have degree at most D_1, the probability of $\tilde{Q}_1(r_1) = Q_1(r_1)$ is at most $D_1/q \leq 1/n^2$, hence the overall acceptance probability is at most $1/n$, using the same argument as in the proof of Claim 3.18. ■

The "savings" in running time above (i.e., the improvements over the brute-force algorithm that runs in time $2^{(1+o(1))\cdot n}$) were achieved via a probabilistic interactive protocol. The following assumption, denoted #NSETH, asserts that without randomness it is impossible to achieve running time $2^{(1-\epsilon)\cdot n}$, for any constant $\epsilon > 0$.

Assumption 3.19 (#NSETH)**.** *There does not exist a constant $\epsilon > 0$ and a non-deterministic machine M that gets as input a formula Φ over n variables of size $2^{o(n)}$, runs in time $2^{(1-\epsilon)\cdot n}$, and satisfies the following:*

1. *There exists non-deterministic choices such that M outputs the number of satisfying assignments for Φ.*

2. *For all non-deterministic choices, M either outputs the number of satisfying assignments for Φ or outputs $\perp$.*

Recall that the standard strong exponential-time hypotheses SETH asserts that for every $\epsilon > 0$ it is hard to solve k-SAT with n variables in time $2^{(1-\epsilon)\cdot n}$, where $k = k_\epsilon$ is sufficiently large. The assumption #NSETH is incomparable to SETH: On the one hand, in #NSETH we assume hardness with respect to a larger class of formulas (i.e., n-bit formulas of size $2^{o(n)}$), and also assume hardness of the *counting* problem; but on the other hand, the hardness in #NSETH is for non-deterministic machines rather than just for deterministic algorithms.

4 Superfast derandomization of $\mathcal{MA}$

The following is the "basic version" of the highly efficient reconstructive PRG, which was mentioned in the beginning of Section 2. We will further refine this version later on in Propositions 5.2 and 7.10.

Proposition 4.1 (a reconstructive PRG with unambiguous non-deterministic reconstruction)**.** *For every $\epsilon_0 > 0$ there exists $\delta_0 > 0$ and a pair of algorithms that for any $N \in \mathbb{N}$ and $f \in \{0,1\}^{N^{1+\epsilon_0/3}}$ satisfy the following:*

1. **(Generator.)** *The generator G gets input 1^N, oracle access to f, and a random seed of length $(1+\epsilon_0)\cdot\log(N)$, and outputs an N-bit string in time $N^{1+\epsilon_0}$.*

2. **(Reconstruction.)** *For any $D\colon \{0,1\}^N \to \{0,1\}$ such that $\Pr_{s\in[N^{1+\epsilon_0}]}[D(G^f(1^N,s)) = 1] > \Pr_{r\in\{0,1\}^N}[D(r)=1] + 1/10$ there exists a string $\texttt{adv}$ of length $|f|^{1-\delta_0}$ such that the following holds. When the reconstruction R gets input $x \in [|f|]$ and oracle access to D and non-uniform advice $\texttt{adv}$, it runs in non-deterministic time $|f|^{1-\delta_0}$, issues queries in parallel, and unambiguously computes f_x.*

Proof. For a sufficiently small $\gamma = \gamma(\epsilon_0)$ to be determined later, let $\mathsf{Samp}\colon \{0,1\}^{\bar{N}} \times [\bar{L}] \to \{0,1\}^N$ be the sampler from Theorem 3.14, instantiated with parameter γ, with a a sufficiently small constant error, and with $\bar{N} = N^{1+O(\gamma)}$ and $\bar{L} = \bar{N}^{1+O(\gamma)}$.

The generator G. The generator encodes f to $\bar{f} = \mathsf{Enc}(f)$ using the code Enc in Theorem 3.10, instantiated with parameters $m = N$ and ρ, η that are sufficiently small constants. We think of $\bar{f}$ as a binary string (by naively encoding each symbol using $\log(|\Sigma|)$ bits, in which case $|\bar{f}| = \bar{m}\cdot\log(|\Sigma|) = \tilde{O}(N^{1+\epsilon_0/3})$. The generator then partitions $\bar{f}$ into $L = |\bar{f}|/\bar{N}$ consecutive substrings $\bar{f}_1, \ldots, \bar{f}_L$, and given seed $(i,j) \in [L]\times[\bar{L}]$ it outputs the N-bit string $\mathsf{Samp}(\bar{f}_i, j)$. The number of strings in the set is

$$L\cdot\bar{L} = (|\bar{f}|/\bar{N})\cdot(\bar{N}^{1+O(\gamma)}) = N^{1+\epsilon_0/3+O(\gamma)} < N^{1+\epsilon_0}\ ,$$

and the running time of the generator is $\tilde{O}(N^{1+\epsilon_0/3}) < N^{1+\epsilon_0}$.

The reconstruction R. Fix a $(1/10)$-distinguisher $D\colon \{0,1\}^N \to \{0,1\}$. Denoting the uniform distribution over the output-set of G^f by $\mathbf{G}$, our assumption is that $\Pr[D(\mathbf{G}) = 1] > \Pr_{r\in\{0,1\}^N}[D(r) = 1] + 1/10$.

Let $\bar{D}\colon \{0,1\}^{\bar{N}} \to \{0,1\}$ be the function

$$\bar{D}(z) = 1 \iff \Pr_{j\in[\bar{L}]}[D(\mathsf{Samp}(z,j)) = 1] \le \Pr_{r\in\{0,1\}^N}[D(r) = 1] + .01\,, \tag{4.1}$$

and let $S = \bar{D}^{-1}(1)$ and $T = \bar{S} = \bar{D}^{-1}(0)$. [26] Note that $|\bar{S}| \le 2^{\bar{N}^{1-\gamma}}$, by the properties of Samp. Then, by the definition of T we have that

$$\begin{aligned}\Pr[D(\mathbf{G})) = 1] &= \Pr_{i\in[L],j\in[\bar{L}]}[D(\mathsf{Samp}(\bar{f}_i,j))] \\ &\le \Pr_i[\bar{f}_i \in T] + \Pr_{i,j}[D(\mathsf{Samp}(\bar{f}_i,j)) = 1 | \bar{f}_i \notin T] \\ &\le \Pr_i[\bar{f}_i \in T] + \left(\Pr_{r\in\{0,1\}^N}[D(r) = 1] + .01\right)\,. \end{aligned} \tag{4.2}$$

Since $\Pr[D(\mathbf{G}) = 1] - \Pr_r[D(r) = 1] > (1/10)$, we deduce that $\Pr_i[\bar{f}_i \in T] > (1/10) - .01 > \rho$, where the last inequality is by a sufficiently small choice of ρ.

Computing a "corrupted" version of $\bar{f}$. We first construct a machine M that computes a "corrupted" version of $\bar{f}$, as follows. Let $\mathcal{H}$ be the hash family in Theorem 3.13, using parameters $m = \bar{N}$ and $m' = \bar{N}^{1-\gamma/2}$. We argue that:

Fact 4.1.1. *With probability at least $1 - 2^{-\bar{N}^{1-\gamma}}$ over $h \sim \mathcal{H}$, for every distinct $z, z' \in \bar{S}$ it holds that $h(z) \ne h(z')$.*

Proof. For every distinct $z, z' \in \bar{S}$, the probability over $h \sim \mathcal{H}$ that $h(z) = h(z')$ is at most $2^{-\bar{N}^{1-\gamma/2}}$. By a union bound over $|\bar{S}|^2 \le 2^{2\bar{N}^{1-\gamma}}$ pairs, with probability at least $1 - 2^{-\bar{N}^{1-\gamma/2}} \cdot 2^{2\bar{N}^{1-\gamma}} > 1 - 2^{-\bar{N}^{1-\gamma}}$ there does not exist a colliding pair in $\bar{S}$. □

Let $I = \{i \in [L] : \bar{f}_i \in T\}$. The machine M gets as advice the foregoing h, the set $\{(i, h(\bar{f}_i)) : i \in I\}$, and the value $\Pr_{r\in\{0,1\}^N}[D(r) = 1]$. Given $x \in [\bar{f}]$, the machine computes $i \in [L]$ such that the index x belongs to the i^{th} substring of $\bar{f}$, and if $i \notin I$ it outputs zero. Otherwise, the machine:

1. Non-deterministically guesses a preimage $z \in \{0,1\}^{\bar{N}}$ for $\bar{f}_i$ under h.
2. Verifies that $h(z) = \bar{f}_i$ using the advice value $(i, h(\bar{f}_i))$.
3. Verifies that $z \notin S$, using the oracle access to D, the sampler Samp, and the acceptance probability of D (that is given as advice).
4. If either of the two verifications failed, the machine aborts. Otherwise, it outputs the bit in z corresponding to index x.

Note that the foregoing machine computes, in an unambiguous non-deterministic manner, a string $\tilde{f}$ such that $\Pr_x[\bar{f}_x = \tilde{f}_x] \ge \rho$. (This is because for every x belonging to a substring indexed by $i \notin I$ it holds that $\tilde{f}_x = 0$, and for every other x it holds that $\tilde{f}_x = \bar{f}_x$.) The number of advice bits that M uses is at most $\tilde{O}(\bar{N} + L \cdot \bar{N}^{1-\gamma/2} + N) \le$

[26] Denoting the complement of S both by $\bar{S}$ and by T might seem unnecessarily cumbersome. However, this formulation will generalize more easily later on when we prove the "furthermore" statement.

$N^{1+O(\gamma)}$, and its running time is dominated by computing the hash function once (which takes time $\tilde{O}(\bar{N})$) and computing $\bar{D}$ once.

Computing $\bar{f}$, and thus also f. Consider the execution of the local list-decoder Dec from Theorem 3.10 with agreement ρ, when it is given oracle access to the "corrupted" version of $\bar{f}$ computed by M (i.e., the version in which a block $\bar{f}_i$ indexed by $i \in I$ has the correct values of $\bar{f}$, and all other blocks are filled with zeroes). Fixing the "right" index $\eta \in [O(1/\rho)]$ of f in the corresponding list of codewords, we reduce the error probability of Dec to less than $1/N^2$ (by $O(\log(N))$ repetitions) and now consider its execution with a fixed random string.

The reconstruction procedure R gets as advice the non-uniform advice for M, the index η, and the fixed random string for Dec (we will see that the latter string is of length $N^{O(\gamma)}$). Given $x \in [|\bar{f}|]$, it runs Dec and answers its queries using M. If any of the queries to M was answered by $\perp$, we abort and output $\perp$, and otherwise we output the result of the list-decoder's computation. Note that Dec can be described by an oracle circuit of size $\mathrm{poly}(1/\rho) \cdot N^{\eta} + \log(1/\rho) + O(1) \leq N^{2\eta} \leq N^{O(\gamma)}$, where we relied on a sufficiently small choice of η. It issues its queries in parallel, since both Dec and the machine M issue their queries in parallel.

We thus obtained a procedure for $\bar{f}$ that is non-deterministic and unambiguous, and uses at most $N^{1+O(\gamma)} < |f|^{1-\delta_0}$ advice bits, where the inequality relies on sufficiently small choices of γ and of δ_0. Denoting the time for verifying that $z \in T$ by K, the running time of the procedure for $\bar{f}$ is at most

$$N^{O(\gamma)} \cdot \left(\tilde{O}(\bar{N}) + K\right) = N^{1+O(\gamma)} + N^{O(\gamma)} \cdot K\,,$$

and using the naive algorithm for $\bar{D}$ (which enumerates over $i \in [\bar{L}]$) this is at most $N^{1+O(\gamma)} < |f|^{1-\delta_0}$. ■

Given the reconstructive PRG in Proposition 4.1, we can now prove Theorem 1.1:

Theorem 4.2 (derandomization with quadratic overhead from useful properties against SVN circuits)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists an $\mathcal{NTIME}[N^{2+\epsilon/4}]$-constructive property $\mathcal{L}$ useful against $(\mathcal{N} \cap co\mathcal{N})\mathcal{TIME}[2^{(2-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$. Then, for any time bound T we have that $pr\mathcal{BPTIME}[T] \subseteq pr(\mathcal{N} \cap co\mathcal{N})\mathcal{TIME}[T^{2+\epsilon}]$.*

Proof. Let A be a $pr\mathcal{BPTIME}[T]$ machine, fix a sufficiently large input length $|x|$, and let $N = T(|x|)$. Given input x, our derandomization algorithm guesses a string f_n of length $2^n = N^{1+\epsilon/3}$ and verifies that $f_n \in \mathcal{L}$ (if the verification fails, it aborts).[27] It then enumerates over the seeds of the generator from Proposition 4.1, when the latter is instantiated with $\epsilon_0 = \epsilon/2$ and with f_n as the oracle, to obtain $N^{1+\epsilon/2}$ pseudorandom strings $w_1, \ldots, w_{N^{1+\epsilon/2}} \in \{0,1\}^N$; and it outputs $\mathsf{MAJ}\left\{A(x, w_i)\right\}_{i \in [N^{1+\epsilon/2}]}$. This derandomization algorithm runs in time $O(N^{2+\epsilon}) = O(T(n)^{2+\epsilon})$.

Let $\delta \leq \delta_0/2$ be sufficiently small. For any $n \in \mathbb{N}$ and $N = 2^{n/(1+\epsilon/3)}$, assume that there is $x \in \{0,1\}^{T^{-1}(N)}$ and $f_n \in (\mathcal{L} \cap \{0,1\}^{2^n})$ such that $D_x(r) = A(x, r)$ is a $(1/10)$-distinguisher for the generator above, when the latter guesses the truth-table f_n. Let $\bar{D}_x$ be either the function computed by D_x (if D_x accepts a pseudorandom input with higher probability than it does a random input) or the negation of that function (otherwise). By Proposition 4.1, there is a reconstruction algorithm R for

[27]For simplicity we assume that $N^{1+\epsilon/3}$ is a power of two, since rounding issues do not meaningfully affect the proof.

f_n that runs in time $|f_n|^{1-2\delta}$ and uses oracle access to $\bar{D}_x$ and $|f_n|^{1-2\delta}$ bits of advice. To simulate the oracle for R, we supply R with additional advice x of length $|x| = T^{-1}(2^{n/(1+\epsilon/3)}) \leq |f_n|^{1-2\delta}$ and with an advice bit indicating whether or not to flip the output of D_x. Plugging in the time complexity of D_x as $N = |f_n|^{1/(1+\epsilon/3)}$, the running time of R is $|f_n|^{(1-2\delta)+1/(1+\epsilon/3)} < |f_n|^{2-\delta}$, and it non-deterministically and unambiguously computes the function whose truth-table is f_n.

Now, assume towards a contradiction that there are infinitely many $x \in \{0,1\}^*$ and f_n such that D_x is a $(1/10)$-distinguisher for the generator with f_n, and fix corresponding advice strings for R as above. (On input lengths for which there are no suitable x and f_n, the advice string indicates that R should compute the all-zero function.) This yields $L \in (\mathcal{N} \cap co\mathcal{N})\mathcal{TIME}[2^{(2-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ whose truth-tables are included, infinitely often, in $\mathcal{L}$, contradicting the usefulness of $\mathcal{L}$. ■

The proof of Theorem 4.2 also yields the following result, in which both the hypothesis and the conclusion are stronger:

Theorem 4.3 (derandomization with quadratic overhead from batch-computable truth-tables)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists $L \notin \texttt{i.o.}(\mathcal{N} \cap co\mathcal{N})\mathcal{TIME}[2^{(2-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ such that there is an algorithm that gets input 1^n and prints the truth-table of L on n-bit inputs in time $2^{(2+\epsilon/4)\cdot n}$. Then, for any time bound T we have that $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[T^{2+\epsilon}]$.*

We think of the algorithm in the hypothesis of Theorem 4.3 as "batch-computing" the hard function: It prints the entire truth-table in time slightly larger than 2^{2n}, whereas computing individual entries in the truth-table cannot be done in time slightly less than 2^{2n} (even using unambiguous non-determinism). The only difference between the proof of Theorem 4.2 and in the proof of Theorem 4.3 is that in the latter, instead of guessing and verifying the truth-table of a hard function, we explicitly compute the truth-table using the hypothesized algorithm.

5 Superfast derandomization of $\mathcal{AM}$

In this section we prove Theorems 1.2 and 1.3. Towards this purpose, in Section 5.1 we refine the reconstructive PRG from Proposition 4.1. Then, in Section 5.2.1, we conditionally construct two PRGs that rely on different hardness hypotheses and have different parameters, both of which use the foregoing refined reconstructive PRG. In Section 5.2.2 we compose the two PRGs in order to prove Theorem 1.2. Lastly, in Section 5.4 we use the reconstructive PRG in a different way to prove Theorem 1.3.

5.1 Refining the reconstructive PRG from Proposition 4.1

We now extend Proposition 4.1 in two ways. First, we argue that by allowing randomness in the reconstruction, we can reduce its query complexity (this is along the lines of ideas from [DMO+20; CT21b]). Secondly, we claim that the reconstruction does not need "full oracle access" to D; loosely speaking, the functionality of the reconstruction is maintained as long as we can guarantee that D answers zero on a sufficiently large fraction of the queries.

Definition 5.1 (α-indicative sequences)**.** *Let $t, s \in \mathbb{N}$ such that $s|t$, let $\alpha \in (0,1)$, and let $d \in \{0,1\}^t$. We say that d is (s,α)-*`valid` *if for every $i \in [t/s]$ it holds that $\sum_{j\in[s]} d_{(i-1)+j} \geq$*

*$\alpha \cdot s$. We say that a sequence $q \in \{0,1\}^t$ is (s,α)-*indicative of d *if for every $i \in [t/s]$ it holds that $\sum_{j\in[s]}(q_{(i-1)+j} \wedge d_{(i-1)+j}) \geq \alpha \cdot s$. We say that a sequence $q \in \{0,1\}^t$ is (s,α)-*deficient *if there exists $i \in [t/s]$ such that $\sum_{j\in[s]} q_{(i-1)+j} < \alpha \cdot s$.*

Proposition 5.2 (an extension of the PRG from Proposition 4.1)**.** *In the reconstruction of Proposition 4.1, for any D satisfying the hypothesis and any input x and witness w for the reconstruction procedure, denote by $\bar{R}^D(x,w) \in \{f_x, \perp\}$ the output of R with oracle access to D and advice* `adv`*. If we allow R to use randomness, then we may assume that it makes only $\bar{t} = N^{\epsilon_0/10}$ parallel queries to D and satisfies $\Pr[R^D(x,w) = \bar{R}^D(x,w)] \geq 2/3$.*

Furthermore, denoting $\bar{N} = N^{1+\epsilon_0/3}$, there exists $s \in \mathbb{N}$ satisfying $s|\bar{t}$ and $\alpha \in (0,1)$ and a random variable $\mathbf{h}$ over $\{0,1\}^{\bar{N}^{1-2\delta_0}}$ that can be sampled in quasilinear time, such that for any $D\colon \{0,1\}^N \to \{0,1\}$ satisfying $\Pr_{r\in\{0,1\}^N}[D(r) = 1] \leq 1/3$, with probability at least $1 - 1/N$ over $h \sim \mathbf{h}$ the following holds. For any input $x \in [\bar{N}]$ and witness w and random coins γ, denote by $d_{x,w,\gamma} \in \{0,1\}^{\bar{t}}$ the evaluations of D on the queries made by R, and denote by $a_{x,w,\gamma} \in \{0,1\}^{\bar{t}}$ the answers that R received to these queries. Then,

1. **(Honest oracle.)** *For any $f \in \{0,1\}^{\bar{N}}$, assume that $\Pr_{s\in[N^{1+\epsilon_0}]}[D(G^f(s)) = 1] \geq 1/2$. Then, there exists* `adv` $\in \{0,1\}^{\bar{N}^{1-2\delta_0}}$ *such that when R gets advice $(h,$* `adv`*$)$ the following holds.*

   (a) *Completeness: For every x there exists w such that with probability $1 - 1/N$ over γ it holds that $d_{x,w,\gamma}$ is (s,α)-valid, and if $a_{x,w,\gamma}$ is (s,α)-indicative of $d_{x,w,\gamma}$ then $R(x,w)$ outputs f_x.*

   (b) *Soundness: For every (x,w), with probability at least $1 - 1/N$ over γ, if $a_{x,w,\gamma}$ is (s,α)-indicative of $d_{x,w,\gamma}$, then $R(x,w)$ outputs either f_x or $\perp$.*

2. **(Dishonest oracles.)** *For every* `adv` $\in \{0,1\}^{\bar{N}^{1-2\delta_0}}$ *there exists $g \in \{0,1\}^{\bar{N}}$ such that when R gets advice $(h,$* `adv`*$)$ the following holds. For every (x,w), with probability at least $1 - 1/N$ over γ, if $a_{x,w,\gamma}$ is (s,α)-indicative of $d_{x,w,\gamma}$ then $R(x,w)$ outputs either g_x or $\perp$.*

3. **(Deficient oracles.)** *For every advice $(h,$* `adv`*$)$ to R and any (x,w) and any choice of γ, if $a_{x,w,\gamma}$ is (s,α)-deficient then R outputs $\perp$.*

We stress that the "dishonest oracles" claim and the "deficient oracles" claim do not depend on any particular choice of f for the generator G. That is, the two claims refer only to the behavior of the reconstruction R given advice $(h,$ `adv`$)$ and oracle access to D such that $\Pr_{r\in\{0,1\}^N}[D(r) = 1] \leq 1/3$.

Proof of Proposition 5.2. We follow the same proof as for Proposition 4.1 and explain the necessary changes. Let us start with proving the bound of $N^{\epsilon/10}$ on the number of queries when R is allowed to use randomness.

We modify the definition of $\bar{D}$ in Eq. (4.1). Specifically, let $\nu = \Pr_{r\in\{0,1\}^N}[D(r) = 1]$, and let $\bar{D}$ be a probabilistic procedure that gets $z \in \{0,1\}^{\bar{N}}$, uniformly samples $s = O(\log(N))$ values $i_1, \ldots, i_s \in [\bar{L}]$ and outputs 1 if and only if $\Pr_{j\in[s]}[D(\mathsf{Samp}(z,i_j)) = 1] < \nu + .005$. We also modify the definitions of S and of T, as follows:

$$S = \left\{ z \in \{0,1\}^{\bar{N}} : \Pr_{j\in[\bar{L}]}[D(\mathsf{Samp}(z,j)) = 1] \leq \nu + .001 \right\} ,$$

and

$$T = \left\{ z \in \{0,1\}^{\bar{N}} : \Pr_{j \in [\bar{L}]} [D(\mathsf{Samp}(z,j)) = 1] > \nu + .01 \right\} .$$

Note that $T \subseteq \bar{S}$, that $|\bar{S}| \leq 2^{\bar{N}^{1-\gamma}}$ (by the properties of Samp, assuming we instantiate it with a sufficiently small error), and that $\Pr_i[\bar{f}_i \in T] > \rho$ (using the exact same calculation as in Eq. (4.2)). Also, $\bar{D}$ accepts every $z \in S$, with probability at least $1 - 1/N^2$, and rejects every $z \in T$, with probability at least $1 - 1/N^2$.

Given these properties, the rest of the proof continues with only one change. Specifically, in the third step of the execution of M, we compute $\bar{D}(z)$ (using randomness), and continue only if the output is zero. Since any $z \in S$ will be accepted with high probability, if we continue then we are confident that $z \in \bar{S}$, and thus (given that we verified $h(z) = h(\bar{f}_i)$) we are certain that $z = \bar{f}_i$. The final procedure that uses Dec and M uses at most $\bar{t} = s \cdot N^{O(\gamma)} < N^{\epsilon_0/10}$ queries to D.

The "furthermore" part. We partition the queries of R into $\bar{t}/s$ sets of size s in the natural way (i.e., each subset corresponds to a set of s queried made by one of the executions of M).[28] The variable $\mathbf{h}$ is the choice of hash function, and as proved in Fact 4.1.1, with probability at least $1 - 2^{-\bar{N}^{1-\gamma}} > 1 - 1/N$, there are no distinct $z, z' \in \bar{S}$ such that $h(z) = h(z')$. We also modify the definition of $\bar{D}$ and S and T above to use the value $\nu = 1/3$ instead of $\nu = \Pr_{r \in \{0,1\}^N}[D(r) = 1]$, and let $\alpha = 1/3 + .005$.

<u>Deficient oracles.</u> Assume, for a moment, that all the queries that R makes to M during its execution are to indices in substrings $\bar{f}_i$ such that $i \in I$. Then, the soundness condition for deficient oracles follows immediately by the definitions of M and R: If in any execution of M less than α of its queries are answered by 1 then $\bar{D}(z) = 0$ and M aborts (in which case R outputs $\perp$).

The only gap is that some queries to M might be to indices in substrings $\bar{f}_i$ where $i \notin I$. To handle this gap we change the definition of M as follows. Whenever $i \notin I$, the original machine just outputs 0, whereas our new modified machine will non-deterministically guess $z \in \{0,1\}^{\bar{N}}$ and verify that $z \notin S$, and only if the verification passes it outputs 0 (otherwise it aborts). This does not affect the original proof in any way, but now the soundness condition for deficient oracles holds even without the assumption that queries to M are such that $i \in I$.

<u>Honest oracle.</u> We now assume that D accepts a pseudorandom string of G^f with probability at least $1/2$. Observe that $\Pr_r[\bar{f}_i \in T] \geq \rho$ as in the proof of Proposition 4.1; to see this, use the same calculation as in Eq. (4.2) only replacing the original value $\Pr_{r \in \{0,1\}^N}[D(r) = 1]$ with the new value $\nu = 1/3$.

The string $\mathtt{adv}$ consists of the hash values $\{(i, h(\bar{f}_i) : i \in I\}$ where $I = \{i \in [L] : \bar{f}_i \in T\}$ and of the local list decoding circuit Dec. (The circuit Dec is just as in the proof of Proposition 4.1: We consider the execution of Dec on the corrupted codeword defined by D and h and I, use naive error-reduction, and hard-wire a good random string. This circuit is given to R as part of the advice $\mathtt{adv}$.) We can assume that the advice string is of length $|f|^{1-2\delta_0}$ (rather than δ_0) by choosing the parameter δ_0 to be smaller than in Proposition 4.1.

[28]Recall that the queries that Dec makes to M are non-adaptive and that the queries that M makes to D are also non-adaptive.

Now, recall that the non-determinism w for R yields non-deterministic strings for each of the execution of M. For every execution of M on query q and with non-determinism z, with probability at least $1 - 1/N^2$, if at least $\alpha \cdot t$ of M's oracle queries are answered by 1, and D indeed evaluates to 1 on these queries, then M outputs the value $\sigma \in \{0, 1, \perp\}$ such that $\Pr[M^D(q, z) = \sigma] \geq 2/3$. By a union-bound, with probability $1 - 1/O(N)$ this happens for all queries to M.

In this case, we can think of the oracle that Dec gets access to as a fixed string y in $\{0, 1, \perp\}$. Given any input, if Dec sees gets an answer of $\perp$ from M, then it outputs $\perp$; and otherwise it outputs the answer corresponding to the unique codeword determined by y and by the list-decoding index.[29]

To prove completeness, note that for every x there exists w for which $\sigma \in \{0, 1\}$ for every possible query q to M. Also, for such w, all the corresponding z's will be in $\bar{S}$. Thus, with probability $1 - 1/O(N)$ over the coins of M, at least an α-fraction of the queries of M to the oracle will be to 1-instances of D. In this case, when at least an α-fraction of the received answers are 1, then Dec (and hence also R) outputs f_x.

Dishonest oracles. Fix any advice adv to R. We can assume without loss of generality that adv consists of hash values $\{(i, h_i) : i \in I\}$ where $I \subseteq [L]$ such that $|I| \geq \rho$ and of the index $\eta \in [O(1/\rho)]$ of a codeword for the local decoder Dec and a random string for Dec (otherwise R can always output $\perp$, regardless of x and w).

We partition $[L]$ into $\bar{I} = [L] \setminus I$, and $I_0 = \{i \in I : \exists z \in \bar{S}, h(z) = h_i\}$, and $I_1 = I \setminus I_0$. For every $i \in I_0$ let $z(i)$ be the unique string in $\bar{S}$ such that $h(z(i)) = h_i$. Denote by $\bar{g} \in \{0, 1\}^{|\bar{f}|}$ the following string: For every $q \in [|\bar{g}|]$, let $i(q)$ be the $i \in [L]$ such that q indexes a location in $\bar{g}_i$, and let $\bar{g}(q) = \begin{cases} 0 & i(q) \in \bar{I} \cup I_1 \\ z(i)_q & i(q) \in I_0 \end{cases}$, where $z(i)_q$ is the bit in $z(i)$ corresponding to the location indexed by q. Note that $\bar{g}$ along with the index η define together a unique codeword g (i.e., g is the η^{th} codeword in the list of codewords that agree with $\bar{g}$ on at least ρ indices).

Now fix any (x, w). For any query q to M with non-determinism z', if $i(q) \notin I$ then by definition M can only output either $\bar{g}(q) = 0$ or $\perp$. Also, with probability at least $1 - 1/N^2$ over γ, if at least $\alpha \cdot s$ of M's oracle queries are answered by 1 and D evaluates to 1 on these queries:

1. If $i(q) \in I_0$ then M outputs $\begin{cases} \bar{g}(q) & z' = z(i) \\ \perp & o.w. \end{cases}$.
2. If $i(q) \in I_1$ then M outputs $\perp$. (Because the oracle answers guarantee that $z' \notin \bar{S}$, and hence $h(z') \neq h_i$.)

By a union bound, with probability $1 - 1/O(N)$ the above holds for all queries that Dec makes to M. Now, consider an execution of Dec on input $x \in [|f|]$ with oracle access to $\bar{g}$ and with the index η and the randomness that R provides it. Denote by g the string such that g_x is the answer of Dec on input $x \in [|f|]$ in such an execution. Note that g depends only on D, on h and on the advice adv.

The last step is to analyze the behavior of Dec when its gets oracle access to M rather than to $\bar{g}$, and under the condition on M's random choices above and the assumption that $a_{x,w,\gamma}$ is α-indicative of $d_{x,w,\gamma}$. Recall that Dec issues its queries in parallel, and therefore it makes the same queries to M and to $\bar{g}$. We now consider two

[29]To be accurate, in the latter case Dec returns the answer corresponding to the string y' in which $\perp$'s are replaced with 0's.

cases: If at least one query of Dec to M is answered by $\perp$, then R aborts and outputs $\perp$ (by the definition of R). Otherwise, all of the queries of Dec to M are answered according to $\bar{g}$; in this case Dec outputs g_x. ■

5.2 The superfast derandomization result: Basic version

In this section we prove Theorem 1.2. First, in Section 5.2.1 we present two PRGs that use the refined construction from Proposition 5.2. Then, in Section 5.2.2 we show how to compose these PRGs to obtain our hardness vs randomness results for $\mathcal{AM}$ protocol.

5.2.1 Two PRGs that will be composed later

The first PRG, presented next, uses a truth-table that can be recognized in near-linear time, and is hard for $\mathcal{MAM}$ protocols running in time $2^{(1-\delta)\cdot n}$ with $2^{(1-\delta)\cdot n}$ bits of non-uniform advice, for a small constant $\delta > 0$. This PRG allows to reduce the number of coins the verifier uses in any T-time protocol to be approximately $\log(T)$. (Recall that in our final result we want the number of coins to be approximately $\log(n)$.)

Proposition 5.3 (the "outer PRG" – radically reducing the number of random coins)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists an $\mathcal{NTIME}[N^{1+\epsilon/3}]$-constructive property $\mathcal{L}$ useful against $\mathcal{MAM}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$. Then, for every time bound T it holds that $pr\mathcal{AMTIME}[T] \subseteq \mathcal{AMTIME}[T^{1+\epsilon}, (1+\epsilon)\cdot \log(T)]$.*

Proof. Fix a problem $\Pi = (\mathsf{Y}, \mathsf{N}) \in pr\mathcal{AMTIME}[T]$, and let V be a T-time verifier for Π. Given input $x \in \{0,1\}^n$, let $N = T(n)$. We guess $f_n \in \{0,1\}^{N^{1+\epsilon/3}}$ and verify that $f_n \in \mathcal{L}$ (otherwise we abort). Consider the generator G from Proposition 5.2 with $\epsilon_0 = \epsilon$ and input 1^N oracle access to f_n, and denote its set of outputs by $s_1, \ldots, s_{N^{1+\epsilon}} \in \{0,1\}^N$. The new verifier V' chooses a random $i \in [N^{1+\epsilon}]$ and simulates V at input x with random coins s_i. Note that this verifier indeed runs in time $O(N^{1+\epsilon})$.

The reconstruction argument. Assume towards a contradiction that there exists an infinite set S of pairs (x, f_n) such that $x \in \mathsf{N}$ and $\Pr_{i\in[N^{1+\epsilon}]}[\exists w : V(x, s_i, w) = 1] \geq .5$, where the s_i's are the outputs of G on non-deterministic guess f_n. For every such pair, denote by $D_x\colon \{0,1\}^N \to \{0,1\}$ the function $D_x(z) = 1 \iff \exists w : V(x, z, w) = 1$, and note that $\Pr_{r\in\{0,1\}^N}[D_x(r) = 1] \leq 1/3$ (because $x \in \mathsf{N}$).

We design an $\mathcal{MAM}$ protocol for f_n as follows. Consider the reconstruction algorithm R from Proposition 5.2 with oracle access to D_x and with the corresponding advice string. Given input $z \in [|f_n|]$, our protocol acts as follows:

1. The prover sends non-determinism w for R.
2. The verifier simulates R using random coins, computes a set $q_1, .., .q_t$ of queries to D_x, and sends them back to the prover.
3. The prover sends t responses $w_1, \ldots, w_t$, to be used as non-determinism for D_x with queries $q_1, \ldots, q_t$.
4. (Deterministic step.) The verifier computes the values $\{d_i = V(q_i, z, w_i)\}_{i\in[t]}$. Then it simulates R with witness w and with the query answers $\{d_i\}_{i\in[t]}$, and accepts iff R accepts.

We now use the "furthermore" part of Proposition 5.2. By the "honest oracle" part, and our assumption about D_x, there exists adv and $s \in \mathbb{N}$ and $\alpha \in (0,1)$ such that for every x there exists w for which with high probability over R's random coins, the oracle queries can be answered in a manner that will be (s,α)-indicative of the (s,α)-valid sequence of answers by D_x, and whenever that happens R outputs f_x; and whenever the sequence of answers to oracle queries are (s,α)-indicative of the sequence of answers by D_x, then R outputs a value in $\{\perp, f_x\}$.

We hard-wire $\texttt{adv}, s, \alpha$ as non-uniform advice to the $\mathcal{MAM}$ protocol. For every x the prover can send the "correct" w in the first step, and witnesses $w_1, \ldots, w_t$ in the third step such that $d_i = D_x(q_i)$ for all $i \in [t]$, in which case the output of R is f_x, with high probability. To establish the soundness of the protocol, observe that we always have $d_i \le D_x(q_i)$ (i.e., it is impossible that $d_i = 1$ whereas $D_x(q_i) = 0$). Then, by Proposition 5.2, with high probability the following holds: If the sequence of answers to the verifier's queries is (α, s)-deficient, then R outputs $\perp$; and otherwise, it means that the answers are (s,α)-indicative of the sequence of answers by D_x, in which case we are in the soundness case and the output is either f_x or $\perp$.

By hard-wiring appropriate advice for every input length n such that $(x, f_n) \in S$, the protocol above computes a language whose truth-tables are included in $\mathcal{L}$ infinitely often. (As in the proof of Theorem 4.2, on input lengths for which there are no suitable x and f_n, the advice string indicates that the protocol should compute the all-zero function.) The time complexity of the protocol is at most

$$O(\underbrace{|f_n|^{1-\delta_0}}_{\text{complexity of } R} + \underbrace{N^{\epsilon/10}}_{\text{number of queries (i.e. } t)} \cdot \underbrace{N}_{\text{complexity of } V}) < |f_n|^{1-\delta}, \tag{5.1}$$

where the inequality relies on a sufficiently small choice of δ; and similarly, the advice needed for this protocol is of length $|x| + |f_n|^{1-\delta_0} + O(1) < |f_n|^{1-\delta}$. This contradicts the usefulness of $\mathcal{L}$. ■

Remark 5.4 (handling $\mathcal{AM}$ with imperfect completeness)**.** Observe that the proof above can be easily adapted to yield derandomization of $\mathcal{AM}$ protocols that do not have perfect completeness, at the cost of strengthening the hardness assumption. Specifically, assume that f_n is hard even for $\mathcal{MA}$ protocols (with the specific running time and advice) that can issue *oracle queries* to $\mathcal{AMTIME}[O(n)]$. Then, given $x \in \mathsf{Y}$ such that $\Pr[\exists w : V(x, s_i, w) = 1]$ is too low, we can define $\bar{D}_x = 1$ to be the negation of the function D_x defined in the proof above, and run the reconstruction argument with oracle access to $\bar{D}_x$, contradicting the hardness of f_n. We chose to present our result as above merely since $\mathcal{AM}$ with perfect completeness is a natural class to consider, and we preferred using the weaker hypothesis.

A tangent: Derandomization $pr\mathcal{BPP}$**.** Using the proof approach of Proposition 5.3, we can obtain a derandomization of $pr\mathcal{BPP}$ (rather than of $\mathcal{AM}$) in time that is optimal under #NSETH, using hypotheses that compare favorably to previous results. Specifically, we show that:

Theorem 5.5 (optimal derandomization of $pr\mathcal{BPP}$ without OWFs)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that for any polynomial $T(n)$ the following holds. Assume that:*

1. *There exists $L \notin \texttt{i.o.}\mathcal{MATIME}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ and a deterministic algorithm that gets input 1^n, runs in time $2^{(1+\epsilon/3)\cdot n}$, and prints the truth-table of L on n-bit inputs.*

2. *For a constant $k = k_T \geq 1$ there exists $L \notin \texttt{i.o.}\mathcal{DTIME}[2^{(k-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ and a deterministic algorithm that gets input 1^n, runs in time $2^{(k+1)\cdot n}$, and prints the truth-table of L on n-bit inputs.*

Then, $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[n \cdot T^{1+\epsilon}]$.

The conclusion in Theorem 5.5 is identical to that in [CT21b], and so is the hypothesis in Item (2). The new part is that the hypothesis in Item (1) of Theorem 5.5 replaces the cryptographic hypothesis that one-way functions exist in [CT21b].

Proof sketch for Theorem 5.5. Using the hypothesis in Item (1), we mimic the proof of Proposition 5.3 to argue that $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{BPTIME}[T^{1+\epsilon/2}, (1+\epsilon/2) \cdot \log(T)]$, where the latter class is that of problems that can be decided in time T with $(1+\epsilon/2) \cdot \log(T)$ random coins. (This follows since the generator in Proposition 5.3 can now deterministically print f_n, which is the truth-table of L on $(1+\epsilon/6) \cdot \log(N)$ input bits; and since the reconstruction protocol does not need an additional round, because the distinguisher is now just a deterministic function.)

We then use the hypothesis in Item (2) with the Nisan-Wigderson generator, when the latter is instantiated for small output length (see Theorem 5.16 for a statement of the generator's parameters). Instantiating this generator with the truth-table of L on inputs of length $(1+\Theta(\epsilon)) \cdot \log(n)$ as in [CT21b, Section 4.2] or in Proposition 5.17, we deduce that $pr\mathcal{BPTIME}[T^{1+\epsilon/2}, (1+\epsilon/2) \cdot \log(T)] \subseteq pr\mathcal{DTIME}[n \cdot T^{1+\epsilon}]$ ■

The inner PRG. The next PRG that we present assumes that there is a function whose truth-tables can be recognized in time approximately n^k and that is hard for $\mathcal{MAM}$ protocols running in time $2^{(1-\delta)\cdot k \cdot n}$ with $2^{(1-\delta)\cdot n}$ bits of non-uniform advice (again $\delta > 0$ here is a small constant). Under this assumption, any protocol that uses at most linearly-many random coins can be derandomized with time overhead that is multiplicative in the input length.

Proposition 5.6 (the "inner PRG" – derandomizing $\mathcal{AM}$ protocols with few random coins)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Fix $k \geq 1$, and assume that there exists an $\mathcal{NTIME}[N^{k+\epsilon}]$-constructive property $\mathcal{L}$ useful against $\mathcal{MAM}[2^{(1-\delta)\cdot k \cdot n}]/2^{(1-\delta)\cdot n}$. Then,*

$$\mathcal{AMTIME}[n^k, n] \subseteq \mathcal{NTIME}[n^{1+(1+\epsilon)\cdot k}] .$$

Proof. Fix $\Pi \in \mathcal{AMTIME}[n^k, n]$, and let V be the n^k-time verifier that uses at most n random coins. Given input $x \in \{0,1\}^n$ and witness $w \in \{0,1\}^{n^k}$, we guess $f_n \in \{0,1\}^{n^{1+\epsilon/3}}$ and verify that $f_n \in \mathcal{L}$. Consider G from Proposition 5.2 with $\epsilon_0 = \epsilon$ and input 1^n oracle access to f_n. We enumerate over the output-set $s_1, \ldots, s_{n^{1+\epsilon}} \in \{0,1\}^n$ of G and output $\mathsf{MAJ}_i \{V(x, s_i, w)\}$. Note that this verifier indeed runs in time $O(n^{(1+\epsilon/3)\cdot(k+\epsilon)} + n^{1+\epsilon} \cdot n^k) < n^{1+(1+\epsilon)\cdot k}$.

The reconstruction argument is essentially identical to the one in the proof of Proposition 5.3, the only difference being its time complexity. Specifically, using a calculation analogous to Eq. (5.1), the time complexity of the reconstruction in our parameter setting is

$$O(\underbrace{|f_n|^{1-\delta_0}}_{\text{complexity of } R} + \underbrace{n^{\epsilon/10}}_{\text{number of queries}} \cdot \underbrace{n^k}_{\text{complexity of } V}) < |f_n|^{(1-\delta)\cdot k} ,$$

where the inequality relies on the fact that $k + \epsilon/10 < (1+\epsilon/3)(1-\delta) \cdot k$, using a sufficiently small choice of $\delta > 0$. ■

5.2.2 Composing the two PRGs

By a straightforward combination of Proposition 5.3 and Proposition 5.6, we obtain the following result, which is the first conclusion in Theorem 1.2:

Corollary 5.7 (superfast derandomization of $\mathcal{AM}$; the first part of Theorem 1.2)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that for every $k \geq 1$ there exists an $\mathcal{NTIME}[N^{k+\epsilon/3}]$-constructive property useful against $\mathcal{MAM}[2^{(1-\delta)\cdot k \cdot n}]/2^{(1-\delta)\cdot n}$. Then, for every polynomial T it holds that $pr\mathcal{AMTIME}[T] \subseteq pr\mathcal{NTIME}[n \cdot T^{1+\epsilon}]$.*

Proof. Let $T(n) = n^c$. By Proposition 5.3 with parameter value $\epsilon/3$ (using our hypothesis with $k = 1$), we have that $pr\mathcal{AMTIME}[T] \subseteq pr\mathcal{AMTIME}[T^{1+\epsilon/3}, (1 + \epsilon/3) \cdot \log(T)]$. By Proposition 5.6 instantiated with parameter values $\epsilon/3$ and $k = (1 + \epsilon/3) \cdot c$, such that $T^{1+\epsilon/3} = n^k$, we have that $pr\mathcal{AMTIME}[n^k, O(\log(n))] \subseteq pr\mathcal{AMTIME}[n^{1+(1+\epsilon/3)\cdot k}]$, and note that $n^{1+(1+\epsilon/3)\cdot k} \leq n \cdot T^{1+\epsilon}$. ■

To prove the "furthermore" part of Theorem 1.2 we combine the foregoing result with a careful application of the standard round-reduction procedure for $\mathcal{AM}$ protocols by Babai and Moran [BM88].

Corollary 5.8 (superfast derandomization of $\mathcal{AMTIME}^{[\rightleftharpoons c]}$; the "furthermore" part of Theorem 1.2, restated)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that for every $k \geq 1$ there exists an $\mathcal{NTIME}[N^{k+\epsilon/3}]$-constructive property useful against $\mathcal{MAM}[2^{(1-\delta)\cdot k \cdot n}]/2^{(1-\delta)\cdot n}$. Then, for every polynomial T and constant $c \in \mathbb{N}$,*

$$pr\mathcal{AMTIME}^{[\rightleftharpoons c]}[T] \subseteq pr\mathcal{NTIME}[n \cdot T^{\lceil c/2 \rceil + \epsilon}] .$$

and

$$pr\mathcal{MATIME}^{[\rightleftharpoons c]}[T] \subseteq pr\mathcal{NTIME}[T^{\lfloor c/2 \rfloor + 1 + \epsilon}] .$$

Proof. Let us first prove the claim about $\mathcal{AM}$, and note that the case of $c = 2$ was proved in Corollary 5.7. For $c \geq 3$, we first use the standard round-reduction of Babai and Moran [BM88] to unconditionally simulate the class in $\mathcal{AM}$, while carefully tracking the simulation overheads, and then we apply Corollary 5.7.

Specifically, denote by $pr\mathcal{AMTIME}^{[\rightleftharpoons c, T']}[T]$ (resp., pr$\mathcal{MATIME}^{[\rightleftharpoons c, T']}[T]$) a protocol with c turns in which the verifier runs in time T in each turn and the prover sends T' bits in each turn. We use the following result:

Proposition 5.8.1 (the round-reduction of [BM88] for $\mathcal{AM}$; see, e.g., [Gol08, Appendix F.2.2.1, Extension])**.** *For every constant $c \geq 3$ we have that $pr\mathcal{AMTIME}^{[\rightleftharpoons c, T']}[T] \subseteq pr\mathcal{AMTIME}^{[\rightleftharpoons \min\{c-2,2\}, O(T')]}[T \cdot T']$.*

We can apply Proposition 5.8.1 for $c' = \lceil c/2 \rceil - 1$ times to deduce that

$$pr\mathcal{AMTIME}^{[\rightleftharpoons c]}[T] \subseteq \mathcal{AMTIME}[O(T^{\lceil c/2 \rceil})] ,$$

and the claim follows from Corollary 5.7.

The case of $\mathcal{MATIME}^{[\rightleftharpoons c]}$. Let $\Pi = (\mathsf{Y}, \mathsf{N}) \in pr\mathcal{MATIME}^{[\rightleftharpoons c]}[T]$ and let V be a corresponding T-time protocol. Let $V_{>1}$ be the sub-protocol of V after the first turn, which takes an input $x \in \{0,1\}^n$ to Π and a first prover message $w_1 \in \{0,1\}^{T(n)}$ as input. Note that $V_{>1}$ is an $\mathcal{AMTIME}^{[\rightleftharpoons c-1]}[O(n)]$ protocol on $n + T(n)$ bits of input.

From the definition of $\mathcal{MATIME}^{[\rightleftharpoons c]}[T]$, we have

$$\begin{cases} \exists\, w \in \{0,1\}^{T(n)} \text{ s.t. } V_{>1}(x,w) \text{ accepts with probability } 1 & \text{for } x \in \mathsf{Y}\,, \\ \forall\, w \in \{0,1\}^{T(n)}\ V_{>1}(x,w) \text{ accepts with probability at most } 1/3 & \text{for } x \in \mathsf{N}\,. \end{cases}$$

Let $\bar{\Pi} = (\bar{\mathsf{Y}}, \bar{\mathsf{N}})$ such that $(x,w) \in \bar{\mathsf{Y}}$ if $V_{>1}(x,w)$ accepts with probability at least $2/3$, and $(x,w) \in \bar{\mathsf{N}}$ if $V_{>1}(x,w)$ accepts with probability at most $1/3$. Note that $\bar{\Pi} \in pr\mathcal{AMTIME}^{[\rightleftharpoons c-1]}[O(n)]$, and hence (by the first part of the proof), we have that $\bar{\Pi} \in pr\mathcal{NTIME}[n^{1+\lceil (c-1)/2 \rceil + \epsilon}] = pr\mathcal{NTIME}[n^{1+\lfloor c/2 \rfloor+\epsilon}]$.

Let $M(x,w)$ be the corresponding nondeterministic machine that decides $\bar{\Pi}$. We can decide Π as follows: Given input $x \in \{0,1\}^n$, guess $w \in \{0,1\}^{T(n)}$, simulate $M(x,w)$ and accept iff M accepts. Note that for $x \in \mathsf{Y}$, there exists $w \in \{0,1\}^{T(n)}$ such that $(x,w) \in \bar{\mathsf{Y}}$, and hence $M(x,w)$ accepts on this particular w. And when $x \in \mathsf{N}$, for all $w \in \{0,1\}^{T(n)}$ we have $(x,w) \in \bar{\mathsf{N}}$, and thus $M(x,w)$ rejects on all possible w. Therefore, we have that $\Pi \in pr\mathcal{NTIME}[T^{1+\lfloor c/2 \rfloor+\epsilon}]$. ■

5.3 The superfast derandomization result: Stronger version

In this section we prove the stronger version of Theorem 1.2, which was mentioned after the original theorem's statement. Our main goal will be to relax the hypothesis that is needed for values of $k > 1$, by requiring hardness only against $\mathcal{NTIME}$ machines with advice, rather than against $\mathcal{MAM}$ protocols with advice.

To do so, we replace the "inner" generator from Proposition 5.6 with the Nisan-Wigderson generator, which is similar to a proof in [CT21b]. The key challenge is that the latter generator requires hardness against algorithms (with advice) that have oracle access to $\mathcal{NTIME}$, whereas we only want to assume hardness against $\mathcal{NTIME}$ machines (with advice). To bridge this gap, we first show, in Section 5.3.1, how to transform a truth-table that is hard for $\mathcal{NTIME}$ machines with advice into a truth-table that is hard for $\mathcal{DTIME}$ machines with advice that have oracle access to $\mathcal{NTIME}$. This proof follows an argument from [SU06], but uses a more careful analysis to obtain tighter bounds on the running time. Then, in Section 5.3.2 we combine this transformation with an application of the Nisan-Wigderson generator as above to prove the stronger version of Theorem 1.2 (for the result statement, see Theorem 5.16).

5.3.1 A refined analysis of the "random curve reduction"

In this section it will be more convenient to work with the notion of non-uniform programs, rather than with Turing machines that take advice. A non-uniform program A on n-bit inputs with advice complexity α and running time T is a RAM program Π of description size α (*i.e.*, $|\Pi| \leq \alpha$). Given an input $x \in \{0,1\}^n$, $A(x)$ is defined to be the output of running the program Π on input x for at most T steps (if, Π does not stop, we define the output to be 0). The notation "A on n-bit inputs" means that we only care about A's outputs on n-bit inputs.[30]

[30]This is the reason why we use *program* instead of *algorithm*. We want to emphasize the fact that a non-uniform program is a *non-asymptotic* object and we only care about its behaviors on a fixed-input length n; this is similar to a circuit with n-bit inputs.

We will need the notions of non-uniform single-valued programs (which are non-uniform programs analogous of $\mathcal{NP} \cap co\mathcal{NP}$) and of non-uniform non-adaptive SAT-oracle program (which are non-uniform programs analogous to $\mathcal{P}$ with oracle access to $\mathcal{NP}$), defined as follows.

Definition 5.9 (non-uniform SVN programs)**.** *A single-valued nondeterministic program A on n-bit inputs with advice complexity α and running time T is a nonuniform program with the same advice complexity and running time which receives two inputs: an input $x \in \{0,1\}^n$ and a second input y of length at most T, and outputs two bits: the* ***value*** *bit and the* ***flag*** *bit. A computes the function $f\colon \{0,1\}^n \to \{0,1\}$ if the following hold:*

- *For every $x \in \{0,1\}^n$ and y, if the* ***flag*** *bit of $A(x,y)$ equals 1, then the* ***value*** *bit of $A(x,y)$ equals $f(x)$.*
- *For every x, there is y such that the* ***flag*** *bit of $A(x,y)$ equals 1.*

We note that if $\alpha = T$, then a single-valued nondeterministic program is essentially a single-valued nondeterministic circuit (see [SU06]).

Definition 5.10 (non-uniform programs)**.** *A non-adaptive non-uniform* SAT*-oracle program A on n-bit inputs is a pair of non-uniform programs A_{pre} and A_{post}. The program A_{pre} has n-bit inputs, and an input $x \in \{0,1\}^n$, it outputs queries $q_1, \dots, q_k$.* [31] *The program A_{post} receives $x \in \{0,1\}^n$ together with k bits $a_1, \dots, a_k$ where $a_i = 1$ if and only if $q_i \in \mathsf{SAT}$, and outputs a single answer bit.*

The running time (resp. advice complexity) of A is defined as the sum of the running time (resp. advice complexity) of A_{pre} and A_{post}. We also call k the query complexity of A.

For convenience, we will assume that the SAT oracle takes the description of a formula ϕ as input, and the number of variables in ϕ is at most the description length of ϕ. In other words, to prove that a formula ϕ with m-bit description length is satisfiable, one only needs to provide a satisfying assignment with at most m bits.

The transformation. The main result that we prove in this section is a transformation of truth-tables of functions that are hard for nondeterministic programs to truth-tables of functions that are hard for SAT-oracle programs, as follows:

Theorem 5.11 (the "random curve reduction" with a careful analysis of overheads)**.** *There is a universal constant $c > 1$ and an algorithm $A_{\mathsf{low\text{-}d}}$ which takes an input function $f\colon \{0,1\}^n \to \{0,1\}$, a real $\epsilon \in (0,1)$, and an integer $k \le 2^{\epsilon n/100}$ as input, and outputs the truth-table of another function $g\colon \{0,1\}^{(1+\epsilon)n} \to \{0,1\}$ such that for all sufficiently large $n \in \mathbb{N}$:*

1. *$A_{\mathsf{low\text{-}d}}(f,\epsilon,k)$ runs in $\tilde{O}(2^{(1+\epsilon)n})$ time.*
2. *If f does not have a single-valued nondeterministic program with running time T and advice complexity α, then g does not have a non-adaptive non-uniform* SAT*-oracle program with running time $T \cdot (n^{1/\epsilon} \cdot k)^{-c}$, advice complexity $\alpha - O(n^2 \cdot k)$, and query complexity k.*

To prove Theorem 5.11 we will need the following technical tools. First, we need the standard low-degree extension of Boolean functions, and we use the precise definition from [SU06].

[31] On all inputs from $\{0,1\}^n$ the program A_{pre} outputs the same number of queries.

Definition 5.12 (low-degree extension). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function, h, q be powers of 2 such that $h \le q$ and $d \in \mathbb{N}$ such that $h^d \ge 2^n$. Let H be the first h elements from $\mathbb{F}_q$, and I be an efficiently computable injective mapping from $\{0,1\}^n \setminus H^d$. The low-degree extension of f with respect to q, h, d is the unique d-variable polynomial $\hat{f}\colon \mathbb{F}_q^d \to \mathbb{F}_q$ with degree $h-1$ in each variable, such that $\hat{f}(I(x)) = f(x)$ for all $x \in \{0,1\}^n$ and $\hat{f}(v) = 0$ for all $v \in (H^d \setminus \mathit{Im}(I))$.*

We will also consider the Boolean function of $\hat{f}$, denoted as $f_{\mathsf{bool}}\colon \{0,1\}^{d \log q + \log\log q} \to \{0,1\}$ and defined by $f_{\mathsf{bool}}(x,i) = \hat{f}(x)_i$, where $\hat{f}(x)_i$ denotes the i-th bit of the binary representation of $\hat{f}(x)$.

Since in Definition 5.12 we always consider finite fields $\mathbb{F}_q$ whose size is a power of 2, we can naturally encode each element in $\mathbb{F}_q$ by exactly $\log q$ bits. Hence, the mapping I can be constructed as follows: given $x \in \{0,1\}^n$, partition it into d consecutive blocks $x^{(1)}, \dots, x^{(d)}$ each of size $\log h$ and ignore the remaining bits (note that $n \ge d \cdot \log h$), and then interpret each $x^{(i)}$ as an element of H via a natural bijection (first interpret $x^{(i)}$ as an integer from $[h]$, and then interpret the obtained integer u as the u-th element from H). Also, by standard interpolation, the truth-table of $\hat{f}$ or f_{bool} can be computed in $\tilde{O}(q^d)$ time from the truth-table of f. For simplicity, we will chose the representation of $\mathbb{F}_q$ in such that a way that for every $u \in \{0,1\}^n$, $f(u) = f_{\mathsf{bool}}(I(u), 1)$.

We also need the notion of parametric curves, and a concentration bound for functions on random curves that was proved in [SU06].

Definition 5.13 (parametric curves). *Let q be a prime power and $f_1, \dots, f_q$ be an enumeration of elements from $\mathbb{F}_q$. For convenience we will assume $f_1 = 0$. Given r elements $v_1, \dots, v_r \in \mathbb{F}_q^d$ for $r \le q$, we define the curve passing through $v_1, \dots, v_r$ to be the unique degree $r-1$ polynomial function $c\colon \mathbb{F}_q \to \mathbb{F}_q^d$ such that $c(f_i) = v_i$ for every $i \in [r]$. We say that a curve c is* one to one *if $c(f_i) \ne c(f_j)$ for every distinct pair i, j from $[q]$.*

The following technical lemma from [SU06] asserts that any function (or set of functions) satisfy good concentration bounds on random curves. To state it, denote by $c(x, c_1, \dots, c_r)$ the unique curve passing through $x, c_1, \dots, c_r$, and note that $c(x, c_1, \dots, c_r)(0) = x$ by its definition. We also use $\mathbb{F}_q^*$ to denote $\mathbb{F}_q \setminus \{0\}$. For two finite sets W, Z such that $W \subseteq Z$ and $h\colon Z \to [0,1]$, we define

$$\mu_W(h) = \frac{1}{|W|} \sum_{i \in W} h(i) \, .$$

Then, the technical lemma from [SU06] is as follows:

Lemma 5.14 (curves are good samplers; see [SU06]). *Let q be a prime power and r be an integer such that $2 \le r < q$. For every point $x \in \mathbb{F}_q^d$, a list of k functions $h_1, \dots, h_k\colon \mathbb{F}_q^d \to [0,1]$, and $\delta \in (0,1)$, the probability over a random choice of points $v_1, \dots, v_r \in \mathbb{F}_q^d$ that $c(x, c_1, \dots, c_r)$ is one to one and*[32]

$$\left| \mu_{c(x,c_1,\dots,c_r)(\mathbb{F}_q^*)}(h_i) - \mu_{\mathbb{F}_q^d}(h_i) \right| < \delta$$

for every $i \in [k]$ is at least

$$1 - \left(8k \cdot \left(\frac{2r}{(q-1) \cdot \delta^2} \right)^{r/2} + \frac{1}{q^{d-2}} \right) .$$

[32]We use $c(x, c_1, \dots, c_r)(\mathbb{F}_q^*)$ to denote the set $\{c(x, c_1, \dots, c_r)(u) : u \in \mathbb{F}_q^*\}$.

We now turn to the actual proof of Theorem 5.11.

Proof of Theorem 5.11. We mimic the proof of [SU06, Theorem 3.2] with a more careful choice of parameters. We also keep track of the running time and non-uniformity separately, instead of a single measure of non-uniform circuit size as in [SU06].

We define $\mathsf{pw}(x) = 2^{\lceil \log x \rceil}$. That is, $\mathsf{pw}(x)$ is the smallest power of 2 that is at least x. Let $c_0 \in \mathbb{N}$ be a large enough constant to be chosen later. We first define the following parameters:

1. $r = c_0 \cdot n$.
2. $h = \mathsf{pw}(\tilde{h})$ where $\tilde{h} = \max\left(c_0 \cdot r^2 \cdot (kn)^4, (c_0 \cdot (n+3) \cdot r)^{3/\epsilon}\right)$.
3. $d = \lceil n/\log h \rceil + 3$.
4. $q = \mathsf{pw}(\tilde{q}/2)$, where $\tilde{q} = c_0 \cdot h \cdot d \cdot r$.

Construction of the function g. Now, we define $\hat{f} \colon \mathbb{F}_q^d \to \mathbb{F}_q$ and $f_{\mathsf{bool}} \colon \{0,1\}^{d \cdot \log q + \log\log q} \to \{0,1\}$ as the low-degree extension of f with respect to q, h, d, according to Definition 5.12. And we define our output function g so that given an input $x \in \{0,1\}^{(1+\epsilon)\cdot n}$, $g(x)$ computes f_{bool} on the length-$(d \cdot \log q + \log\log q)$ prefix of x.

Fact 5.14.1. *By our parameter choices we have that $(1+\epsilon) \cdot n \geq d \cdot \log q + \log\log q$, and hence g is well-defined.*

Proof. By the definition of q, we have that $q \leq \tilde{q} = h \cdot (c_0 \cdot d \cdot r)$. By the definition of h, we have that

$$h \geq \tilde{h} \geq (c_0 \cdot (n+3) \cdot r)^{3/\epsilon} \geq (c_0 \cdot d \cdot r)^{3/\epsilon} \ .$$

The above further implies that $q \leq \tilde{q} \leq h^{1+\epsilon/3}$. Hence, for a sufficiently large $n \in \mathbb{N}$, we have that

$$\begin{aligned} d \cdot \log q + \log\log q &= (\lceil n/\log h \rceil + 3) \cdot \log q + \log\log q \\ &\leq n \cdot \frac{\log q}{\log h} + 5 \cdot \log q \leq (1+\epsilon/3) \cdot n + 5 \cdot \log q \ . \end{aligned}$$

Next, from the definition of q and the assumption that $k \leq 2^{\epsilon n/100}$, we have

$$\begin{aligned} \log q &\leq O(\log n) + \log h \leq O(\log n) + 4 \cdot \log k \\ &\leq O(\log n) + 4 \cdot \epsilon/100 \cdot n \leq \epsilon/20 \cdot n \ . \end{aligned}$$

Putting the above together, we have

$$d \cdot \log q + \log\log q \leq (1 + \epsilon/3 + \epsilon/4) \cdot n < (1+\epsilon) \cdot n \ .$$

□

As mentioned after Definition 5.12, the truth-table of g can be computed in $\tilde{O}(2^{(1+\epsilon)n})$ time. This proves Item (1).

Construction of a probabilistic program B' for f_{bool}. To prove Item (2), it suffices to show the following

- If f_{bool} has a non-adaptive non-uniform SAT-oracle program $A = (A_{\mathsf{pre}}, A_{\mathsf{post}})$ with running time $T \geq 1$, advice complexity $\alpha \geq 0$, and query complexity k,
- then f_{bool} has a single-valued nondeterministic program B with advice complexity $\alpha + O(n^2 \cdot k)$ and running time $O(T \cdot q + \mathrm{poly}(q))$.[33]

We will first construct a probabilistic single-valued nondeterministic program B' that computes f_{bool} and then fix its randomness to obtain the desired program that computes f (the definition of such a probabilistic program will be clear later in the proof). Using our assumption about f_{bool}, we can construct a non-adaptive non-uniform SAT-oracle program $\hat{A} = (\hat{A}_{\mathsf{pre}}, \hat{A}_{\mathsf{post}})$ with query complexity $m = k \cdot \log q$, advice complexity α, and running time $T \cdot \log q$ that computes $\hat{f} \colon \mathbb{F}_q^d \to \mathbb{F}_q$.

For $x \in \mathbb{F}_q^d$, let $Q_1(x), \ldots, Q_m(x)$ and $A_1(x), \ldots, A_m(x)$ be the queries and answers associated with $\hat{A}$, respectively, on input x. We then define $p_i = \mu_{\mathbb{F}_q^d}(A_i)$ for every $i \in [m]$ and $\delta = 1/(9m)$. The probabilistic program B' takes the α-bit advice of $\hat{A}$ together with all the $p_1, \ldots, p_m$ as advice.[34]

On an input $(x, b) \in \{0,1\}^{d \log q} \times [\log q]$, B' works as follows:

1. Pick $v_1, \ldots, v_r$ uniformly at random, and set $x_a = c_{x, v_1, \ldots, v_r}(a)$ for every $a \in \mathbb{F}_q$. Simulate $\hat{A}_{\mathsf{pre}}$ to compute queries $Q_i(x_a)$ for every $i \in [m]$ and $a \in \mathbb{F}_q^*$.
2. Set $n_i = \lfloor (p_i - \delta) \cdot (q-1) \rfloor$. For every $i \in [m]$, guess $z_i \in \{0,1\}^{\mathbb{F}_q^*}$ with exactly n_i ones, and T-bit strings $\{w_{i,a}\}_{a \in \mathbb{F}_q^*}$.
3. For every $i \in [m]$ and $a \in \mathbb{F}_q^*$, check that $(z_i)_a = 1$ implies that $w_{i,a}$ is a witness that query $Q_i(x_a)$ is answered positively (recall that the query is to SAT); otherwise, set the **flag** bit in the output to be 0 and halt.
4. For every $a \in \mathbb{F}_q^*$, compute $y_a = \hat{A}_{\mathsf{post}}(x_a, (z_1)_a, (z_2)_a, \ldots, (z_m)_a)$.
5. Run the algorithm from Lemma 3.12 on the $q-1$ pairs (f_a, y_a) with degree $u = hdr$ to obtain a polynomial $\tau \colon \mathbb{F}_q \to \mathbb{F}_q$ of degree u. Set the **flag** bit in the output to be 1. If no such τ exists, set the **value** bit in the output to be 0, and otherwise set the **value** bit to be the b-th bit of $\tau(0)$.

Analysis of the program B'. We need the following claim.

Claim 5.15. *For every $(x, b) \in \mathbb{F}_q^d \times [\log q]$, with probability more than $1 - \frac{2^{-n}}{2 \log q}$ over the choice of $v_1, \ldots, v_r$, the following two conditions hold:*

1. *For all guesses z_i and $w_{i,a}$ such that the **flag** bit of the output is set to 1, then the **value** bit of the output is $f_{\mathsf{bool}}(x, b)$.*
2. *There exist guesses z_i and $w_{i,a}$ such that the **flag** bit of the output is set to 1.*

[33]By the discussion after Definition 5.12, we have $f(u) = f_{\mathsf{bool}}(I(u), 1)$. Hence B can be used to compute f as well with a minor overhead in the running time.

[34]Note that each p_i can be described by a single integer between 0 and q^d. Hence these m reals take $m \cdot O(d \cdot \log q) = O(m \cdot n)$ bits to store.

Proof. Fix $x \in \mathbb{F}_q^d$. We apply Lemma 5.14 to show that over a random choice of points $v_1, \ldots, v_r \in \mathbb{F}_q^d$,

$$c_{(x,v_1,\ldots,v_r)} \text{ is one to one and for all } i \in [m],\ \left|\mu_{c(x,c_1,\ldots,c_r)(\mathbb{F}_q^*)}(A_i) - \mu_{\mathbb{F}_q^d}(A_i)\right| < \delta \qquad (1)$$

holds with probability at least

$$1 - \left(8m \cdot \left(\frac{2r}{(q-1)\cdot\delta^2}\right)^{r/2} + \frac{1}{q^{d-2}}\right).$$

We need to show the above is lower bounded by $1 - \frac{2^{-n}}{2\log q}$. First, by our choices of q, h, d, we have

$$\frac{1}{q^{d-2}} = \frac{1}{q^{\lceil n/\log h\rceil+1}} \le \frac{1}{h^{\lceil n/\log h\rceil}} \cdot \frac{1}{q} \le \frac{2^{-n}}{4\log q}.$$

Next, note that $\frac{2r}{(q-1)\cdot\delta^2} \le \frac{4\cdot 9^2\cdot rm^2}{q} < 1/2$ by our choice of q (note that $q > h > c_0 \cdot r^2 \cdot (kn)^4 > c_0 \cdot r \cdot m^2$, and c_0 is sufficiently large). We then have

$$8m \cdot \left(\frac{2r}{(q-1)\cdot\delta^2}\right)^{r/2} < 8m \cdot 2^{-r/2} \le \frac{2^{-n}}{4\log q},$$

the last inequality follows from $r = c_0 \cdot n$ for a sufficiently large $c_0 \in \mathbb{N}$, and $m \le \log q \cdot k \le \log q \cdot 2^{\epsilon n/100}$. Putting everything together, we have that (1) holds with probability more than $1 - \frac{2^{-n}}{2\log q}$.

Next we show whenever (1) holds, the two items in the claim hold. For the second item, note that since (1) holds, for every $i \in [m]$ we know that $A_i(x_a) = 1$ for at least n_i distinct elements $a \in \mathbb{F}_q^*$. Therefore, if for every $i \in [m]$, the guess z_i is the string with exactly n_i ones in entries indexed by those a with $A_i(x_a) = 1$, then there are witnesses $w_{i,a_{a\in\mathbb{F}_q^*}}$ that pass the check together with all the z_i.

For the first item, for all guesses z_i and $w_{i,a}$ such that the **flag** bit is 1, we know that for all $i \in [m]$ and $a \in \mathbb{F}_q^*$, $(z_i)_a = 1$ implies $A_i(x_a) = 1$, and there are exactly n_i ones in z_i. On the other hand, by (1), for every $i \in [m]$, the number of $a \in \mathbb{F}_q^*$ such that $A_i(x_a) = 1$ is at most $\lceil (p_i + \delta)\cdot(q-1)\rceil$. Hence, we can bound the number of errors associated with query i as follows:

$$\left|\{a \in \mathbb{F}_q^* \ :\ A_i(x_a) \neq (z_i)_a\}\right| \le \lceil (p_i+\delta)\cdot(q-1)\rceil - \lfloor (p_i-\delta)\cdot(q-1)\rfloor \le 2\delta q.$$

The above in particular means that for all but at most $2\delta q \cdot m$ many $a \in \mathbb{F}_q^*$, we have $(z_i)_a = A_i(x_a)$ for all $i \in [m]$, and consequently $y_a = \hat{f}(x_a)$. Letting p be the restriction $\hat{f} \circ c_{(x,v_1,\ldots,v_r)}$, from the discussions above, it follows that for at least $(q-1) - 2\delta qm = (1-2\delta m)q - 1 = \frac{7}{9}\cdot q - 1$ of the pairs (a, y_a) we have $y_a = p(a)$. Note that the degree of p is at most hdr (since $\hat{f}$ a d-variate polynomial of individual degree at most $h-1$, and we compose it with a curve of degree r). and $q \ge \tilde{q}/2 \ge c_0/2 \cdot hdr$. Since c_0 is a sufficiently large constant, we can apply Lemma 3.12 to compute $p(0) = \hat{f}(x)$, and output the b-th bit of $\hat{f}(x)$ as desired, which completes the proof. □

Finally, let $I\colon \{0,1\}^n \to \mathbb{F}_q^d$ be the injective function associated the low-degree extension $\hat{f}$. Let $S = \{(x,b) : x \in I(\{0,1\}^n), b \in [\log q]\}$. Since $|S| \le 2^n \cdot \log q$, by a union bound, there exists fixed $\hat{v}_1, \ldots, \hat{v}_r$ such that B' with randomness fixed to $\hat{v}_1, \ldots, \hat{v}_r$ correctly computes f_{bool} on all inputs from S. We then define the single-valued nondeterministic program B by fixing the randomness of B' to $\hat{v}_1, \ldots, \hat{v}_r$.

Verifying the complexity of B. We have already established that B computes f_{bool}. It remains to verify the running time and advice complexity of B. First, B takes the following advice:

1. Advice for $\hat{A}$, of length α.
2. Description of the reals $p_1, \ldots, p_m \in [0,1]$, of total length $O(m \cdot n)$.
3. Fixed randomness $\hat{v}_1, \ldots, \hat{v}_r$, which takes $d \log q \cdot r \leq O(n^2)$ bits.

Hence, the number of advice bits of B is $\alpha + O(mn) + O(n^2) \leq \alpha + O(n^2 \cdot k)$.

Finally, the running time of B is dominated by the simulation of $(q-1)$ calls to $\hat{A}$, the final decoding algorithm, and the time to compute the curve $c(x, \hat{v}_1, \ldots, \hat{v}_r)$ by direct interpolation (which takes $\text{poly}(r,q) = \text{poly}(q)$ time). The total running time of B can thus be bounded by

$$O(q \cdot T) + \text{poly}(q) \leq T \cdot \text{poly}(q) \ .$$

and the final bound follows since $q \leq n^{8/\epsilon} \cdot k^4$. ■

5.3.2 An inner PRG relying on weaker hypotheses

We will use the following version of the Nisan-Wigderson [NW94] generator, when it is combined with the locally decodable code of Sudan, Trevisan, and Vadhan [STV01] and with the weak designs of Raz, Reingold, and Vadhan [RRV02]. This version of the generator is instantiated for a sufficiently small output length, and we carefully bound its running time, the running time of the reconstruction argument, and the number of advice bits that the reconstruction argument needs.

Theorem 5.16 (NW generator for small output length)**.** *There exists a universal constant $c_{\mathsf{NW}} > 1$ such that for all $\epsilon_{\mathsf{NW}} > 0$ and $\mu_{\mathsf{NW}} > c_{\mathsf{NW}} \cdot \sqrt{\epsilon_{\mathsf{NW}}}$ there exist two algorithms that for any $N \in \mathbb{N}$ and $f \in \{0,1\}^N$ satisfy the following:*

1. **(Generator.)** *When given input 1^N and oracle access to f, the generator G runs in time $N^{1+2c_{\mathsf{NW}}^2 \cdot \mu_{\mathsf{NW}}}$ and outputs a set of strings in $\{0,1\}^{N^{\epsilon_{\mathsf{NW}}}}$.*
2. **(Reconstruction.)** *For any $(1/N^{\epsilon_{\mathsf{NW}}})$-distinguisher $D \colon \{0,1\}^{N^{\epsilon_{\mathsf{NW}}}} \to \{0,1\}$ for $G(1^N)^f$ there exists a string* `adv` *of length $N^{1-\mu_{\mathsf{NW}}}$ such that the following holds. When the reconstruction R gets input $x \in [|f|]$ and oracle access to D and non-uniform advice* `adv`*, it runs in time $N^{c_{\mathsf{NW}} \cdot \sqrt{\epsilon_{\mathsf{NW}}}}$, makes non-adaptive queries to D, and outputs f_x.*

Proof Sketch. The proof is the standard analysis of the NW generator as in [STV01; RRV02]; specifically, we follow the proof in [CT21a, Appendix A.2] and explain the necessary changes. (These changes are needed since in the current statement we decouple μ_{NW} and ϵ_{NW}, instead of choosing a fixed value for μ_{NW}.)

Denote $\epsilon = \epsilon_{\mathsf{NW}}$ and $\mu = \mu_{\mathsf{NW}}$ and $M = N^{\epsilon}$. Instead of using designs with $\log(\rho) = (1-3\beta) \cdot \ell$ we use designs with $\log(\rho) = (1-2\mu) \cdot \ell$. Then, the seed length of the generator becomes $(1 + O(\beta + \mu)) \cdot \log(N) = (1 + O(\mu)) \cdot \log(N)$. In the reconstruction, the oracle machine P computing a corrupted version of f runs in time $\text{poly}(M) = N^{O(\epsilon)}$ and uses $M \cdot N^{(1+\beta)\cdot(1-2\mu)} = N^{1+2\beta-2\mu}$ bits of non-uniform advice. Thus, the final reconstruction uses $N^{1-\mu}$ bits of advice and runs in time $N^{O(\sqrt{\epsilon})}$. ■

We now use the NW generator above along with Theorem 5.11 to replace the "inner" PRG from Proposition 5.6 with a PRG that relies only on lower bounds against $\mathcal{NTIME}$ machines with advice, rather than against $\mathcal{MAM}$ protocols with advice.

Proposition 5.17 (the "inner PRG" – derandomizing $\mathcal{AM}$ protocols with few random coins)**.** *For every $\epsilon > 0$ there exist $\delta, \eta > 0$ such that the following holds. Fix $k \geq 1$, and assume that there exists an $\mathcal{NTIME}[N^{k+\epsilon/3}]$-constructive property $\mathcal{L}$ useful against $\mathcal{NTIME}[2^{(1-\delta)\cdot k\cdot n}]/2^{(1-\delta)\cdot n}$. Then,*

$$pr\mathcal{AMTIME}[n^k, n^\eta] \subseteq pr\mathcal{NTIME}[n^{1+(1+\epsilon)\cdot k}] .$$

Proof. Let $\eta > 0$ be a sufficiently small constant to be determined later, let $\Pi = (\mathsf{Y}, \mathsf{N}) \in pr\mathcal{AMTIME}[n^k, n^\eta]$, and let V be an n^k-time verifier for Π that uses at most n^η random coins. Given $x \in \{0,1\}^n$, the deterministic verifier guesses a witness $w \in \{0,1\}^{n^k}$ and $f_n \in \{0,1\}^{n^{1+\epsilon/7}}$ and checks that $f_n \in \mathcal{L}$. It then uses the algorithm from Theorem 5.11 with parameter $\epsilon/7$ and with a bound of $Q = n^{(1+\epsilon/3)\cdot c_{\mathsf{NW}}\cdot\sqrt{\eta}}$ on the number of queries to obtain a truth-table $g_n \in \{0,1\}^{n^{1+\epsilon/3}}$. [35]

The verifier uses G from Theorem 5.16 with parameters $\epsilon_{\mathsf{NW}} = \eta$ and $N = n^{1+\epsilon/3}$ and $\mu = C \cdot \sqrt{\eta}$ for a sufficiently large universal constant $C > 0$, giving G oracle access to g_n. Denoting the output strings of $G^{g_n}(1^N)$ by $s_1, \ldots, s_L$, the verifier outputs $\mathsf{MAJ}_i\{V(x, s_i, w)\}$. Note that the running time of G and its number of output strings L are bounded by $n^{(1+\epsilon/3)\cdot(1+O(\sqrt{\eta}))} < n^{1+\epsilon}$, assuming that $\eta > 0$ is sufficiently small. Thus, the deterministic verifier runs in time $n^{1+(1+\epsilon)\cdot k}$.

The reconstruction argument. For any $x \in \{0,1\}^n$, let $D_x \colon \{0,1\}^{n^\eta} \to \{0,1\}$ be the function

$$D_x(r) = 1 \iff \exists w : V(x, r, w) = 1 .$$

Assume towards a contradiction that for infinitely many pairs (x, f_n) such that $x \in \mathsf{N}$ and $f_n \in \{0,1\}^{|x|^{1+\epsilon/7}}$ it holds that $\Pr_{i\in[L]}[D_x(s_i) = 1] \geq 1/2$, where the s_i's are the result of running G with oracle access to g_n and with the parameters above.

For every such pair, there is an advice string $\mathtt{adv}$ of length $|g_n|^{1-C\cdot\sqrt{\eta}}$ such that the reconstruction R from Theorem 5.16 computes the function whose truth-table is g_n in time $|g_n|^{c_{\mathsf{NW}}\cdot\sqrt{\eta}}$ when given $\mathtt{adv}$ as non-uniform advice and oracle access to D_x. By adding x to the advice (we can assume that $|x| < |\mathtt{adv}|$ by choosing η to be sufficiently small), this is an algorithm that runs in time $|g_n|^{c_{\mathsf{NW}}\cdot\sqrt{\eta}}$, uses $|g_n|^{1-C\cdot\sqrt{\eta}}$ bits of advice, and makes Q non-adaptive oracle queries to $\mathcal{NTIME}[n^k]$. [36] Using an efficient reduction of $\mathcal{NTIME}[n^k]$ to SAT (see [Tou01; FLM+05]), we obtain an algorithm that runs in time $T' = \tilde{O}(n^k)$, uses $\alpha' = |g_n|^{1-C\cdot\sqrt{\eta}}$ bits of advice, and makes Q non-adaptive queries to a SAT oracle.

The algorithm R above yields a non-adaptive non-uniform SAT-oracle program with running time T' and advice complexity α' and query complexity Q, in the sense of Definition 5.10. By Theorem 5.11, it holds that g_n can be computed by a non-deterministic non-uniform program with running time $T = T' \cdot \gamma$ and advice complexity $\alpha = \alpha' \cdot \gamma$, where $\gamma = (Q, \log(|f_n|)^{1/\epsilon})^{c_{\mathsf{SU}}} = n^{c_{\mathsf{SU}}\cdot c_{\mathsf{NW}}\cdot(1+\epsilon/3)\cdot\sqrt{\eta}} = |g_n|^{c_{\mathsf{SU}}\cdot c_{\mathsf{NW}}\cdot\sqrt{\eta}}$ and c_{SU} is the constant from Theorem 5.11 that depends on $\epsilon > 0$.

[35]The truth-table that Theorem 5.11 outputs is of length $n^{(1+\epsilon/7)^2}$, and by padding it with zeroes we can assume that it is of length $n^{1+\epsilon/3}$.

[36]In more detail, queries to the oracle are of the form $(x, r) \in \{0,1\}^n \times \{0,1\}^{n^\eta}$, and the oracle answers "yes" iff there exists w such that $V(x, r, w) = 1$.

By the straightforward simulation of the infinite sequence of non-uniform programs to a Turing machine with advice,[37] there exists a non-deterministic machine M that runs in time $\tilde{O}(T)$ and an advice sequence of length α and infinitely many $g_n \in \mathcal{L}$ such that on input length $\log(|g_n|)$, when M is given the appropriate advice it computes g_n. Plugging in the parameters, we have that

$$\begin{aligned}\tilde{O}(T) &= \tilde{O}(n^k \cdot |g_n|^{c_{\mathsf{SU}} \cdot c_{\mathsf{NW}} \cdot \sqrt{\eta}}) = \tilde{O}(|g_n|^{k/(1+\epsilon/3)+c_{\mathsf{SU}} \cdot c_{\mathsf{NW}} \cdot \sqrt{\eta}}) < |g_n|^{k/(1+\epsilon/4)} \\ \alpha &= |g_n|^{1-C\cdot\sqrt{\eta}} \cdot |g_n|^{c_{\mathsf{SU}} \cdot c_{\mathsf{NW}} \cdot \sqrt{\eta}} \leq |g_n|^{1-\sqrt{\eta}}\end{aligned}$$

where the upper-bound on T follows by choosing a sufficiently small $\eta = \eta(\epsilon) > 0$. This contradicts the usefulness of $\mathcal{L}$ if we choose $\delta > 0$ to be sufficiently small. ■

By composing the "outer" PRG from Proposition 5.3 with the PRG from Proposition 5.17, in the exact same way as in Section 5.2.2, we obtain the following stronger version of Theorem 1.2:

Theorem 5.18 (stronger version of Theorem 1.2)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that:*

1. *There exists an $\mathcal{NTIME}[N^{k+\epsilon/3}]$-constructive property useful against $\mathcal{MAM}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$.*
2. *For every $k \geq 1$ there exists an $\mathcal{NTIME}[N^{k+\epsilon/3}]$-constructive property useful against $\mathcal{NTIME}[2^{(1-\delta)\cdot k\cdot n}]/2^{(1-\delta)\cdot n}$.*

Then, for every constant $c \in \mathbb{N}$ it holds that

$$pr\mathcal{AMTIME}^{[\rightleftharpoons c]}[T] \subseteq pr\mathcal{NTIME}[n \cdot T^{\lceil c/2 \rceil + \epsilon}] .$$

5.4 Uniform trade-offs for $\mathcal{AM} \cap co\mathcal{AM}$

In this section we prove Theorem 1.3. The main claim in the proof is the following, which shows that under the hardness assumption of Theorem 1.3, one can reduce the randomness complexity of any $(\mathcal{AM} \cap co\mathcal{AM})\mathcal{TIME}[T]$ protocol to roughly $\log T(n)$. That is:

Proposition 5.19 (radically reducing the number of random coins of $\mathcal{AM} \cap co\mathcal{AM}$)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists $L \notin \mathtt{i.o.}(\mathcal{MA} \cap co\mathcal{MA})\mathcal{TIME}_2^{[\rightleftharpoons 7]}[2^{(1-\delta)\cdot n}]$ such that truth-tables of L of length $N = 2^n$ can be recognized in nondeterministic time $N^{1+\epsilon/3}$. Then, for every time-computable T it holds that*

$$(\mathcal{AM} \cap co\mathcal{AM})\mathcal{TIME}[T] \subseteq (\mathcal{AM} \cap co\mathcal{AM})\mathcal{TIME}[T^{1+\epsilon}, (1+\epsilon) \cdot \log(T(n))].$$

Note that Theorem 1.3 follows immediately from Proposition 5.19 by enumerating all possible random choices; that is, the deterministic verifier asks the prover to send its responses to all possible $T^{1+\epsilon}$ random challenges, checks the responses for consistency, and computes the probability that the original random verifier would have accepted.

Proof of Proposition 5.19. Fixing any $L \in (\mathcal{AM} \cap co\mathcal{AM})\mathcal{TIME}[T]$, we prove that $L \in \mathcal{AMTIME}[T^{1+\epsilon}, (1+\epsilon) \cdot \log T(n)]$; since the same argument can also be applied to the complement of L, it follows that $L \in co\mathcal{AMTIME}[T^{1+\epsilon}, (1+\epsilon) \cdot \log T(n)]$.

[37]That is, the machine gets as advice the description of the program and simulates it.

By definition, there are two T-time verifiers $V_1(x,y,z)$ and $V_0(x,y,z)$ with $|y| = |z| = T(|x|)$, such that the following holds for every $x \in \{0,1\}^n$ and $\sigma = L(x)$:

$$\Pr_{y \sim \mathbf{u}_{T(n)}}[\exists\, z \text{ s.t. } V_\sigma(x,y,z) = 1] = 1\,,$$
$$\Pr_{y \sim \mathbf{u}_{T(n)}}[\exists\, z \text{ s.t. } V_{1-\sigma}(x,y,z) = 1] \le 1/3\,.$$

Let δ_0 be the corresponding constant from Proposition 5.2 when setting $\epsilon_0 = \epsilon$, and without loss of generality assume that $\delta_0 < \epsilon_0$.

Construction of the new verifier V'. Given input $x \in \{0,1\}^n$, let $N = T(n)$. Let $\ell = (1 + \epsilon/3 - \delta_0/10) \cdot \log N$. V' guesses $f_\ell \in \{0,1\}^{N^{1+\epsilon/3-\delta_0/10}}$ and verifies that $f_\ell \in \mathrm{tt}(L)$ by guessing the witness for the nondeterministic algorithm that recognizes $\mathrm{tt}(L)$ (otherwise V' rejects). V' then computes $\mathsf{Enc}(f_\ell)$ using the encoder in Theorem 3.11 with parameters $m = |f_\ell|$ and $\eta > 0$ that is a sufficiently small constant.

Next, we consider the following pair language

$$L_{\mathsf{pair}} = \{(1^\ell, \mathsf{Enc}(f_\ell))) : f_\ell \text{ is the truth-table of } L_{\mathsf{hard}} \text{ on } \ell\text{-bit inputs}\}.$$

Note that L_{pair} has stretch $K(\ell) = |\mathsf{Enc}(f_\ell)| = \tilde{O}(2^\ell)$, and is decidable in $\mathcal{NTIME}[\tilde{T}(\ell))]$ for some $\tilde{T}(\ell) = \tilde{O}(2^\ell)$. Let M be a $\tilde{T}(\ell)$-time nondeterministic machine such that $(x,y) \in L_{\mathsf{pair}}$ if and only if there exists $w \in \{0,1\}^{\tilde{T}(|x|)}$ such that $M((x,y),w) = 1$.

We apply Theorem 3.16 to L_{pair} to obtain a $\mathrm{poly}(\ell)$-time verifier V_{pair} and a $\tilde{O}(2^\ell)$ time algorithm A_{pair}. By Theorem 3.16, for some $r(\ell) \le \ell + O(\log \ell)$, and for every sufficiently large $\ell \in \mathbb{N}$, the followings hold:

1. For every $w \in \{0,1\}^{\tilde{T}(\ell)}$ such that $M((1^\ell, \mathsf{Enc}(f_\ell)), w) = 1$, $A_{\mathsf{pair}}(1^\ell, \mathsf{Enc}(f_\ell), w)$ outputs a proof $\pi \in \{0,1\}^{2^{r(\ell)}}$ such that
$$\Pr[V_{\mathsf{pair}}^{\mathsf{Enc}(f_\ell),\pi}(1^\ell, \mathbf{u}_r) = 1] = 1.$$
2. For every $y \in \{0,1\}^{K(\ell)}$ that has hamming distance at least $K(\ell)/20$ from $\mathsf{Enc}(f_\ell)$, for every $\pi \in \{0,1\}^{2^r}$ it holds that
$$\Pr[V_{\mathsf{pair}}^{\mathsf{Enc}(f_\ell),\pi}(1^\ell, \mathbf{u}_r) = 1] \le 1/3.$$

Then, V' guesses $w \in \{0,1\}^{\tilde{T}(\ell)}$ such that $M((1^\ell, \mathsf{Enc}(f_\ell)), w) = 1$ (V' reject immediately if this does not hold), computes $\pi_\ell^w = A_{\mathsf{pair}}(1^\ell, \mathsf{Enc}(f_\ell), w)$, which has length $2^{r(\ell)}$. Now we define

$$\Lambda_\ell^w = |\mathsf{Enc}(f_\ell)| \circ \pi_\ell \circ 0^{N^{1+\epsilon/3} - |\mathsf{Enc}(f_\ell)| - |\pi_\ell^w|}.$$

Note that $|\Lambda_\ell^w| = N^{1+\epsilon/3}$. Consider the generator G from Proposition 5.2 with $\epsilon_0 = \epsilon$, input 1^N, and oracle access to Λ_ℓ, and denote its list of outputs by $s_1^w, \ldots, s_{N^{1+\epsilon}}^w \in \{0,1\}^N$. The new verifier V' chooses a random $i \in [N^{1+\epsilon}]$ and simulates V_1 at input x with random coins s_i^w. Note that this verifier indeed runs in time $O(N^{1+\epsilon})$. For notational convenience, we now denoted the set of all accepted w by

$$\mathcal{W} = \{w : w \in \{0,1\}^{\tilde{T}(\ell)} \wedge M((1^\ell, \mathsf{Enc}(f_\ell)), w) = 1\}.$$

We note that $|\mathcal{W}| \ge 1$ since M decides L_{pair}.

The reconstruction argument. Assume towards a contradiction that V' fails to solve L. Since V_1 has perfect completeness and $|\mathcal{W}| \geq 1$, V' can only make mistakes when $x \notin L$. In particular, there exist $x \in \{0,1\}^n$ and $w^* \in \mathcal{W}$:

$$x \notin L \wedge \Pr_{i \in [N^{1+\epsilon}]}[\exists \omega : V_1(x, s_i^{w^*}, \omega) = 1] > .5 \ . \tag{5.2}$$

Denote by $D_x \colon \{0,1\}^N \to \{0,1\}$ the function $D_x(z) = 1 \iff \exists \omega : V_1(x, z, \omega) = 1$. From now on, we will use Λ_ℓ to denote $\Lambda_\ell^{w^*}$ for simplicity.

We first design an $\mathcal{MATIME}^{[\rightleftharpoons 7]}$ protocol Π_1 for f_ℓ. Let $\tau = c_0 \cdot \log N$ for a big enough constant $c_0 > 1$. Given an input $\mu \in [|f_\ell|]$, our protocol acts as follows. (See Figure 1 for a visual diagram of the protocol.)

1. The prover sends a (supposedly bad) input $x \in \{0,1\}^n$. From now on, we consider the reconstruction algorithm R from Proposition 5.2 with oracle access to D_x. Let $\bar{t} = N^{\epsilon_0/10}$ be the number of parallel queries R makes, and let $s, \alpha \in \mathbb{N}$ be the parameters from Proposition 5.2.

2. The verifier draws $h \sim \mathbf{h}$ ($\mathbf{h}$ is defined in Proposition 5.2) and τ queries $z_1, z_2, \ldots, z_\tau \in \{0,1\}^N$ uniformly at random, and sends them to the prover. Note that $h \in \{0,1\}^{|f_\ell|^{1-2\delta_0}}$.

3. The prover sends an advice $\texttt{adv} \in \{0,1\}^{|f_\ell|^{1-2\delta_0}}$, together with witnesses $\omega_1, \omega_2, \ldots, \omega_\tau \in \{0,1\}^N$.

4. The verifier then performs the following:

   (a) It rejects immediately if
   $$\Pr_{i \in [\tau]}[V_0(x, z_i, \omega_i) = 1] \leq 1/2.$$

   (b) Otherwise, it draws $\alpha_1, \alpha_2, \ldots, \alpha_\tau \in \{0,1\}^{r(\ell)}$, and then simulates $V_{\mathsf{pair}}(1^\ell, \alpha_i)$ for each $i \in [\tau]$ to obtain a list of $\mathrm{polylog}(\ell)$ queries to the input oracle (which it hopes will be $\mathsf{Enc}(f_\ell)$) and to the proof oracle (which it hopes will be π). We can view those queries as queries $q \in [N^{1+\epsilon/3}]$ to some values of Λ_ℓ.[38]

   (c) The verifier also simulates the local decoder for Enc with input μ, which issues $|f_\ell|^\eta$ many queries $\beta_1, \beta_2, \ldots, \beta_{|f_\ell|^\eta} \in [|\mathsf{Enc}(f_\ell)|]$. Again, we view all these $|f_\ell|^\eta$ queries as queries $q \in \left[N^{1+\epsilon/3}\right]$ to some values of Λ_ℓ.

   (d) To summarize, in this turn, the verifier obtained $N_{\mathsf{q}} = O(|f_\ell|^\eta)$ many queries $q_1, \ldots, q_{N_{\mathsf{q}}} \in [N^{1+\epsilon/3}]$ and sent them to the prover.

5. For every $i \in [N_{\mathsf{q}}]$, the prover sends a witness $w_i \in \{0,1\}^{|f^\ell|^{1-\delta_0}}$, which is supposed to be the witness for the execution of R on q_i.

6. For every $i \in [N_{\mathsf{q}}]$, the verifier simulates R on input q_i with witness w_i with fresh random coins γ_i, obtains queries $z_{i,1}, \ldots, z_{i,\bar{t}} \in [N]$, and sends them to the prover.

[38]More formally, querying the i-th bit of the proof oracle corresponding to $\mathsf{Enc}(f_\ell)$ translates to querying $(\Lambda_\ell)_i$, and querying the i-th bit of the proof oracle corresponding to π translates to querying $(\Lambda_\ell)_{i+|\mathsf{Enc}(f_\ell)|}$.

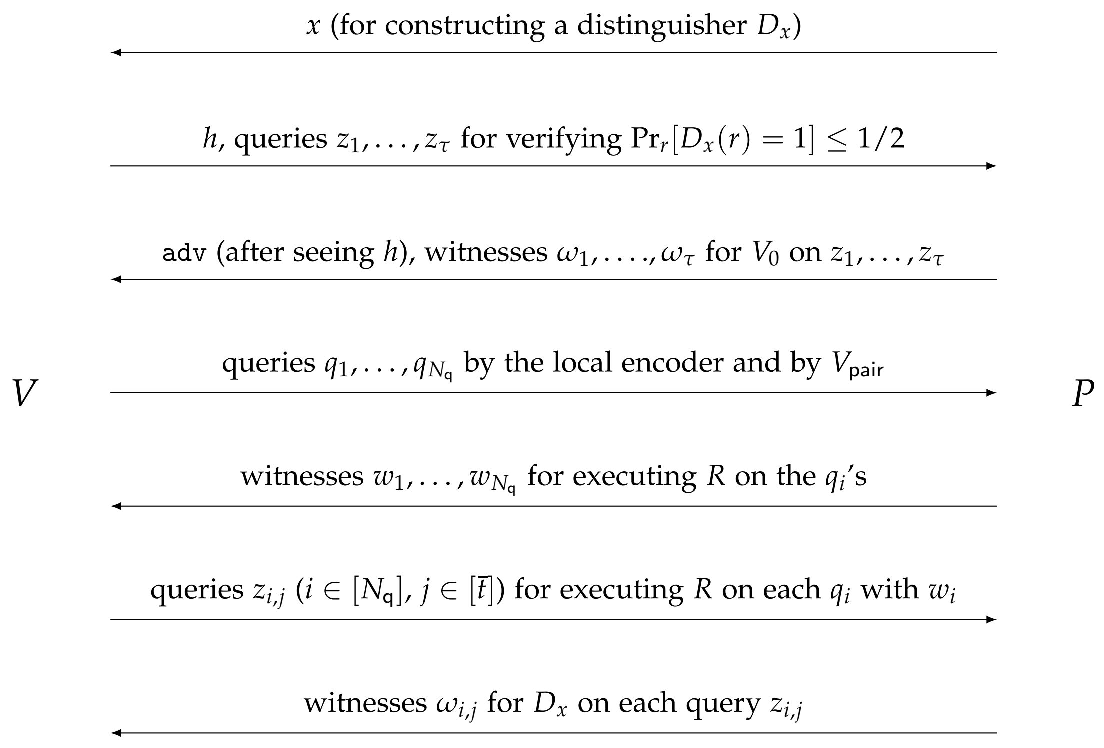
x (for constructing a distinguisher D_x)
h, queries $z_1, \dots, z_\tau$ for verifying $\Pr_r[D_x(r) = 1] \leq 1/2$
$\texttt{adv}$ (after seeing h), witnesses $\omega_1, \dots, \omega_\tau$ for V_0 on $z_1, \dots, z_\tau$
V
queries $q_1, \dots, q_{N_\mathsf{q}}$ by the local encoder and by V_{pair}
P
witnesses $w_1, \dots, w_{N_\mathsf{q}}$ for executing R on the q_i's
queries $z_{i,j}$ ($i \in [N_\mathsf{q}]$, $j \in [\bar{t}]$) for executing R on each q_i with w_i
witnesses $\omega_{i,j}$ for D_x on each query $z_{i,j}$

Figure 1: An illustration of the $\mathcal{MATIME}^{[\rightleftharpoons 7]}$ protocol for f_ℓ in the reconstruction argument. The verifier V is on the left-hand side and the prover P is on the right-hand side, and observe that the prover speaks first.

7. For every $i \in [N_\mathsf{q}]$ and $j \in [\bar{t}]$, the prover sends a witness $\omega_{i,j} \in \{0,1\}^N$, which is supposed to be the witness for $D_x(z_{i,j})$.
8. (Deterministic step.) Finally, the verifier performs the following verifications:
   (a) For every $i \in [N_\mathsf{q}]$, it constructs the sequence $\rho_i \in \{0,1\}^{\bar{t}}$ such that $(\rho_i)_j = V_1(x, z_{i,j}, \omega_{i,j})$ for every $j \in [\bar{t}]$. If for any $i \in [N_\mathsf{q}]$, ρ_i is (s, α)-deficient, then it immediately rejects.
   (b) Otherwise, for every $i \in [N_\mathsf{q}]$, let $d_i \in \{0,1\}^{\bar{t}}$ be such that $(d_i)_j = D_x(z_{i,j})$ for every $j \in [\bar{t}]$. We know that ρ_i is (s, α)-indicative of d_i.[39] The verifier then finishes the executions of R on all the q_i's, and rejects immediately if it gets $\perp$ form any of these executions. Next, the verifier uses the obtained values to finish the simulation of $V_{\mathsf{pair}}(1^\ell, \alpha_i)$ for every $i \in [\tau]$, and rejects immediately if any of the $V_{\mathsf{pair}}(1^\ell, \alpha_i) = 0$. Finally, the verifier finishes the simulation of the local decoder for Enc to obtain an output $\omega \in \{0,1\}$, and accepts iff $\omega = 1$.

Completeness. For every $\mu \in [|f_\ell|]$ such that $(f_\ell)_\mu = 1$, we will show that there exists a prover strategy in the protocol Π_1 such that the verifier accepts, with high probability. In Step (1) the prover sends the ("bad") input $x \in \{0,1\}^n$ such that (5.2) holds. Since $x \notin L$, we have that $\Pr[D_x(\mathbf{u}_N) = 1] \leq 1/3$ and that $\Pr_{z \sim \mathbf{u}_N}[\exists\ \omega\ V_0(x, z, \omega) = 1] = 1$.

[39]This holds since for every $j \in [\bar{t}]$ $(\rho_i)_j = 1$ implies $(d_i)_j = 1$ by the definition of ρ_i and D_x.

Therefore, in Step (3), the prover can always send $\omega_1, \dots, \omega_\tau$ so that the verifier does not reject in Step (4a) in Π_1.

By the completeness case of the "Honest oracle" part of Proposition 5.2, and since Eq. (5.2) holds, with probability $1 - 1/N$ over $h \sim \mathbf{h}$, there exists $\mathtt{adv} \in \{0,1\}^{|f_\ell|^{1-2\delta_0}}$ such that by sending $\mathtt{adv}$ to the verifier in Step (3), the following holds: Given correct witnesses in Step (5), with probability at least $1 - O(N_{\mathsf{q}})/N$, for all of the verifier's $O(N_{\mathsf{q}})$ simulations of R, given correct witnesses in Step (7), the verifier obtains the correct values in Λ_ℓ. In particular, it means that $V_{\mathsf{pair}}(1^\ell, \alpha_i) = 1$ for all $i \in [\tau]$, and the local decoder returns the correct value $(f_\ell)_\mu = 1$. Putting the above together, the verifier accepts with probability at least $2/3$.

Soundness. Let $\mu \in [|f_\ell|]$ such that $(f_\ell)_\mu = 0$. We first note that if thr prover sends $x \in L$ in Step (1), then we have $\Pr_{z \sim \mathbf{u}_N}[\exists\ \omega\ V_0(x, z, \omega) = 1] \le 1/3$, meaning that with probability at least $1 - 1/N$, the verifier rejects in Step (4a), no matter what witnesses $\omega_1, \dots, \omega_\tau$ it receives in Step (3). Therefore, we can assume that $x \notin L$ in Step (1) and that the verifier does not reject immediately in Step (4a). In particular, since $x \notin L$, we have that $\Pr_{z \sim \mathbf{u}_N}[D_x(z)] \le 1/3$.

Now, by the "Dishonest oracles" case of Proposition 5.2, with probability $1 - 1/N$ over $h \sim \mathbf{h}$, for every possible $\mathtt{adv}$ sent by the prover in Step (3), there exists $g \in \{0,1\}^{N^{1+\epsilon/3}}$ such that the following holds for every $i \in [N_{\mathsf{q}}]$: for every possible w_i sent by the prover in Step (5), with probability at least $1 - 1/N$ over the verifier's randomness γ_i drawn in Step (6), either the verifier rejects in Step (8a), or the simulated $R(q_i, w_i)$ in Step (8b) given advice $(h, \mathtt{adv})$ outputs either $g(q_i)$ or $\perp$ (this holds since ρ_i in in Step (8) is either (s,α)-deficient, in which case the verifier rejects, or (s,α)-indicative of d_i, in which case $R(q_i, w_i)$ outputs either $g(q_i)$ or $\perp$).

By the above discussions and a union bound, with probability at least $1 - O(N_{\mathsf{q}})/N$, at Step (8b), either the verifier rejects (meaning that some of $R(q_i, w_i)$ outputs $\perp$), or all the simulated $R(q_i, w_i)$ outputs $g(x_i)$. We will denote the above as event $\mathcal{E}$.

From now on we condition on the event $\mathcal{E}$ and we assume that the verifier does not reject in Step (8a). Let y be the first $|\mathsf{Enc}(f_\ell)|$ bits of g, and π be the next $|\pi_\ell|$ bits of g. Note that g is already determined by the end of Step (3) and hence the verifier's randomness $\alpha_1, \dots, \alpha_\tau$ in Step (4) is independent of g. Hence, if y has Hamming distance at least $|\mathsf{Enc}(f_\ell)|/20$ from $\mathsf{Enc}(f_\ell)$, then with probability at least $1 - 1/N$, $V_{\mathsf{pair}}^{y,\pi}(1^\ell, \alpha_i) = 0$ for at least one $i \in [\tau]$, and the verifier rejects in Step (8b).[40] Hence, we can assume that y has hamming distance at most $|\mathsf{Enc}(f_\ell)|/20$ from $\mathsf{Enc}(f_\ell)$. By Theorem 3.11, the local decoder returns the correct value $(f_\ell)_\mu = 0$ with probability at least $1 - 1/N$, and the verifier rejects at the end. Putting everything together, the verifier rejects with probability at least $2/3$.

Protocol Π_0 for the complement of f_ℓ. Finally, we modify Π_1 to obtain another $\mathcal{MATIME}^{[\rightleftharpoons 7]}$ protocol Π_0. The only difference between Π_0 and Π_1 is that at Step (8b) of Π_0, the verifier in Π_0 accepts if and only if $\omega = 0$ instead of $\omega = 1$. The completeness and soundness of Π_0 for computing the complement of f_ℓ follow from the same proof as that for Π_1.

[40]Here we crucially used the fact that g (and thus y and π) is fixed before the verifier draws the α_i's.

Running time of the protocols Π_1 and Π_0. Finally, we bound the running time of the verifier in Π_1 (and hence also in Π_0), as follows:

$$\underbrace{\tilde{O}\left(|\Lambda_\ell|^{1-2\delta_0}\right)}_{\text{Step (2)}} + \underbrace{\tilde{O}(N) + \mathrm{polylog}(N) + |f_\ell|^\eta}_{\text{Steps (4a)+ (4b)+ (4c)}} + \underbrace{O(N_\mathsf{q} \cdot |\Lambda_\ell|^{1-\delta_0})}_{\text{Step (6)}}$$
$$+ \underbrace{O(N_\mathsf{q} \cdot N \cdot \bar{t})}_{\text{Step (8a)}} + \underbrace{O(N_\mathsf{q} \cdot |\Lambda_\ell|^{1-\delta_0} + \mathrm{polylog}(N) + |f_\ell|^\eta)}_{\text{Step (8b)}}$$
$$\le N_q \cdot O(N \cdot \bar{t} + |\Lambda_\ell|^{1-\delta_0})$$
$$= |f_\ell|^\eta \cdot O(N^{1+\epsilon_0/10} + |\Lambda_\ell|^{1-\delta_0}) \,,$$

and this can be made smaller than $|\Lambda_\ell|^{1-\delta}$ by choosing η and δ to be sufficiently small. This contradicts the hardness of f_ℓ. ■

6 Optimality under #NSETH

In this section we prove that the derandomization conclusions in Theorems 1.1 and 1.2 are essentially optimal, under the assumption #NSETH. First we lower bound the derandomization overhead of protocols in which the prover speaks first (i.e., of $\mathcal{MA}$ and $\mathcal{MATIME}^{[\rightleftharpoons c]}$), as follows:

Theorem 6.1 (a lower bound on derandomization of $\mathcal{MATIME}^{[\rightleftharpoons c]}$, under #NSETH)**.** *Suppose that #NSETH holds. Then, for every integer $c \ge 2$ and real number $d \ge 1$ and $\epsilon \in (0,1)$, letting $T(n) = n^d$, it holds that*

$$\mathcal{MATIME}^{[\rightleftharpoons c]}[T] \not\subseteq \mathcal{NTIME}[T^{\lfloor c/2 \rfloor + 1 - \epsilon}] \,.$$

Proof. Without loss of generality we can assume $\epsilon \in (0, 0.01)$. For the sake of contradiction, we assume that

$$\mathcal{MATIME}^{[\rightleftharpoons c]}[T] \subseteq \mathcal{NTIME}[T^{\lfloor c/2 \rfloor + 1 - \epsilon}] \,.$$

We instantiate Theorem 3.17 with $k = \lfloor c/2 \rfloor$ and $\delta = \frac{1}{k+1}$, in which case $\gamma = \frac{1}{k+1}$. It follows that there is an $\mathcal{MATIME}^{[\rightleftharpoons 2k]}[2^{n/(k+1)+o(n)}]$ protocol Π that computes the number of satisfying assignment to a formula C with $2^{o(n)}$ size and n bits input.

We first define a decisional $\mathcal{MATIME}^{[\rightleftharpoons 2k]}$ protocol Π^D, such that $\Pi^\mathsf{D}(C,z) = 1$ for an n-input formula C and an integer $z \in \{0,1,\dots,2^n\}$, if the number of satisfying assignments to C is z. (Indeed, Π^D can be constructed by simply simulating Π on the input C and only accepting if Π accepts and the accepted output is z.)

Now we pad the input of the protocol Π^D to be of length $N = 2^{\frac{n(1+\tau)}{(k+1)d}}$ for a sufficiently small constant $\tau \in (0,1)$ to be specified later. Then, the running time of the protocol Π^D is bounded by $2^{n/(k+1)+o(n)} \le 2^{(1+\tau)n/(k+1)} = T(N)$. Hence, by our assumption, Π^D has a $T(N)^{k+1-\epsilon} = 2^{\frac{(1+\tau)n(k+1-\epsilon)}{(k+1)}}$ time nondeterministic algorithm M_D.

Setting $\tau = \epsilon/4(k+1)$, it holds that $\frac{(1+\tau)n(k+1-\epsilon)}{(k+1)} < (1-\tau)\cdot n$. Now we can construct a nondeterministic algorithm refuting #NSETH as follows: given a formula $C\colon \{0,1\}^n \to \{0,1\}$ of size $2^{o(n)}$, guess $z \in \{0,1,\dots,2^n\}$, simulate M_D on input (C,z), output z if M_D accepts and $\perp$ otherwise. By the above discussion, this is a nondeterministic algorithm that counts the number of solutions to n-bit formulas of size $2^{o(n)}$ in time $2^{(1-\tau)\cdot n}$, a contradiction to #NSETH. ■

By a more careful argument, we now lower bound the derandomization overhead of protocols in which the verifier speaks first (i.e., of $\mathcal{AMTIME}^{[\rightleftharpoons c]}$), as follows:

Theorem 6.2 (a lower bound on derandomization of $\mathcal{AMTIME}^{[\rightleftharpoons c]}$, under #NSETH)**.** *Suppose that #NSETH holds. Then, for every integer $c \geq 2$ and real number $d \geq 1$ and $\epsilon \in (0,1)$, letting $T(n) = n^d$, it holds that*

$$\mathcal{AMTIME}^{[\rightleftharpoons c]}[T] \not\subseteq \mathcal{NTIME}[n \cdot T^{\lceil c/2 \rceil - \epsilon}] .$$

Proof. Without loss of generality we can assume $\epsilon \in (0, 0.01)$. For the sake of contradiction, we assume that

$$\mathcal{AMTIME}^{[\rightleftharpoons c]}[T] \subseteq \mathcal{NTIME}[n \cdot T^{\lceil c/2 \rceil - \epsilon}] .$$

We instantiate Theorem 3.17 with $k = \lceil c/2 \rceil$ and $\delta = \frac{1}{dk+1}$ and $\gamma = (1-\delta)/k$. Note that since $d \geq 1$, we have $\delta \leq \gamma$. It follows that there is an $\mathcal{MATIME}^{[\rightleftharpoons 2k]}[2^{\gamma \cdot n + o(n)}]$ protocol Π that computes the number of satisfying assignment to a formula C with $2^{o(n)}$ size and n bits input, and the first message of Π has length $2^{\delta \cdot n + o(n)}$.

As in the proof of Theorem 6.1, we first define a decisional $\mathcal{MATIME}^{[\rightleftharpoons 2k]}$ protocol Π^{D}, such that $\Pi^{\mathsf{D}}(C, z) = 1$ for an n-input formula C and an integer $z \in \{0, 1, \ldots, 2^n\}$, if the number of satisfying assignments to C is z. We note that the prover messages of the honest prover in Π^{D} are identical to those in Π. Furthermore, by the moreover part of Theorem 3.17, the first message of Π^{D} is such that the acceptance probability of the subsequent protocol is either 1 or at most 1/3. (Indeed, Π^{D} inherits this property from Π.)

Let $\Pi^{\mathsf{D}}_{>1}$ be the sub-protocol of Π^{D} after the first message and let $\ell(n) = 2^{(1+\tau) \cdot \delta n}$, where $\tau \in (0,1)$ is a small enough constant to be specified later. We define a new language L' such that $L'(x, \pi, 1^{\ell(n) - |x| - |\pi|}) = 1$ if $\Pi^{\mathsf{D}}_{>1}$ accepts the input/first-message pair (x, π) with probability 1, and $L'(x, \pi, 1^{\ell(n) - |x| - |\pi|}) = 0$ if $\Pi^{\mathsf{D}}_{>1}$ accepts (x, π) with probability at most 1/3. Note that (by the discussion above) the problem L' is indeed a language (i.e., a total function rather than a promise problem), and L' can be decided by $\Pi^{\mathsf{D}}_{>1}$, which is an $\mathcal{AMTIME}^{[\rightleftharpoons 2k-1]}$ protocol with running time $2^{\gamma \cdot n + o(n)}$.

Note that $\Pi^{\mathsf{D}}_{>1}$ takes $\ell(n)$ bits as input, and that $\ell(n)^d = 2^{(1+\tau)\delta \cdot d \cdot n} > 2^{\gamma \cdot n + o(n)}$ by the definition of δ and γ. Hence, the language $L' \in \mathcal{AMTIME}^{[\rightleftharpoons c]}[T]$. By our assumption, $L' \in \mathcal{NTIME}[n \cdot T^{k-\epsilon}]$.

Now we construct an algorithm that refutes #NSETH: Given an n-bit formula C of size $2^{o(n)}$, guess $z \in \{0, 1, \ldots, 2^n\}$, guess a proof π of length $2^{\delta \cdot n + o(n)}$, outputs z if $L'((C,z), \pi, 1^{\ell(n) - |(C,z)| - |\pi|}) = 1$, and outputs $\perp$ otherwise. This nondeterministic algorithm indeed counts the number of satisfying assignments, and its running time is at most

$$\ell(n) \cdot \ell(n)^{d(k-\epsilon)} = \ell(n)^{d(k-\epsilon)+1} = 2^{\frac{(1+\tau) \cdot (d(k-\epsilon)+1)}{dk+1} \cdot n} < 2^{(1-\Omega(1)) \cdot n},$$

where the last inequality follows by setting τ to be small enough. This is a contradiction to #NSETH. ∎

7 Deterministic doubly efficient argument systems

In this section we prove the results from Section 1.3 concerning derandomization of doubly efficient proof systems; that is, we prove Theorems 1.7, 1.4 and 1.8.

Let us first set up some preliminaries. The derandomization algorithms in this section will be *non-black-box*, and in particular will use the following construction of a *reconstructive targeted HSG* from our very recent work [CT21a].

Theorem 7.1 (a reconstructive targeted HSG, see [CT21a, Proposition 6.2])**.** *For every $\alpha', \beta' > 0$ and sufficiently small $\eta = \eta_{\alpha',\beta'} > 0$ the following holds. Let $\bar{T}, k\colon \mathbb{N} \to \mathbb{N}$ be time-computable functions such that $\bar{T}(N) \geq N$, and let $g\colon \{0,1\}^N \to \{0,1\}^k$ (where $k = k(N)$) such that the mapping of $(x,i) \in \{0,1\}^N \times [k]$ to $g(x)_i$ is computable in time $\bar{T}(N)$. Then, there exists a deterministic algorithm G_g and a probabilistic algorithm* Rec *that for every $z \in \{0,1\}^N$ satisfy the following:*

1. **Generator.** *When G_g gets input z and $\eta > 0$, it runs in time $k \cdot \bar{T}(N) + \mathrm{poly}(k)$ and outputs a list of $\mathrm{poly}(k)$ strings in $\{0,1\}^{k^\eta}$.* [41]

2. **Reconstruction.** *When* Rec *gets as input z and $\eta > 0$, and gets oracle access to a function $D_z\colon \{0,1\}^{k^\eta} \to \{0,1\}$ that $(1/k^\eta)$-distinguishes the uniform distribution over the output-list of $G_g(z,\eta)$ from a uniform k^η-bit string, it runs in time $\tilde{O}(k^{1+\beta'}) + k^{\beta'} \cdot \bar{T}(N)$, makes $\tilde{O}(k^{1+\beta'})$ queries to D_z, and with probability at least $1 - 2^{-k^\eta}$ outputs a string that agrees with $g(z)$ on at least $1 - \alpha'$ of the bits.*

The reconstruction algorithm above can be thought of as *approximately printing* the string $g(z)$ (i.e., printing a string that agrees with $g(z)$ on at least $1 - \alpha'$ of the bits). For convenience, we define the following corresponding notion of hardness, which is *failing* to approximately print.

Definition 7.2 (failing to approximately print)**.** *We say that a probabilistic algorithm M* fails to approximately print a function $f\colon \{0,1\}^n \to \{0,1\}^*$ with error α on a given string *$x \in \{0,1\}^n$ if $\Pr[M(x)_i = f(x)_i] \leq 1 - \alpha$, where the probability is over $i \in [|f(x)|]$ and over the random coins of M. We will use shorthand notation and say that M* fails to approximately print $f(x)$ with error α.

7.1 Warm-up: The case of an $\mathcal{MA}$-style system

Towards presenting our result, we first present an appealing special case whose proof is far less involved. Denote by $\mathsf{de}\mathcal{IP}_{MA}^{[\rightleftharpoons 2]}[T]$ a doubly efficient proof system in which the prover speaks first, sending a proof π, and then the verifier tosses random coins and decides whether to accept or reject the input x with the proof π. Under suitable hardness assumptions, we simulate $\mathsf{de}\mathcal{IP}_{MA}^{[\rightleftharpoons 2]}$ by deterministic doubly efficient argument systems, with essentially no time overhead.

Theorem 7.3 (derandomizing $\mathsf{de}\mathcal{IP}_{MA}^{[\rightleftharpoons 2]}$ into deterministic doubly efficient argument systems, with almost no overhead)**.** *Suppose that non-uniformly secure one-way functions exist. Let $T(n)$ be any polynomial, and assume that for every $\epsilon' > 0$ there exist $\alpha, \beta \in (0,1)$ and a function $f = f^{(\epsilon')}$ mapping $n + T(n)$ bits to $k(n) = n^{\epsilon'}$ bits such that:*

1. *There exists a non-deterministic unambiguous machine that gets input $((x,\pi), i) \in \{0,1\}^{n+T} \times [n^{1+\epsilon'}]$ and outputs the i^{th} bit of $f(x,\pi)$ in time $\bar{T} = T(n) \cdot k$.*

[41]The fact that the number of strings is $\mathrm{poly}(k)$ is not mentioned in the original statement in [CT21a, Proposition 6.2], but this is just an omission. This fact is established in the proof of the proposition and the applications of the proposition rely on it.

2. *For every probabilistic algorithm M running in time $\bar{T} \cdot n^{\beta}$ and every distribution* $\mathbf{P}$ *over* $\{0,1\}^{n+T}$ *that is samplable in polynomial time, with probability at least* $1 - n^{-\omega(1)}$ *over* $(x,\pi) \sim \mathbf{P}$ *it holds that M fails to approximately print $f(x,\pi)$ with error α.*

Then, for every $\epsilon > 0$ it holds that $\mathsf{de}\mathcal{IP}_{MA}^{[\rightleftharpoons 2]}[T] \subseteq \mathsf{de}\mathcal{DARG}[n^{\epsilon} \cdot T]$.

Proof. Let $L \in \mathsf{de}\mathcal{IP}_{MA}^{[\rightleftharpoons 2]}[T]$, let V be a corresponding verifier for L, and let

$$\mathsf{Y}_V = \left\{(x,\pi) : \Pr_r[V(x,\pi,r) = 1] \geq 2/3\right\}$$
$$\mathsf{N}_V = \left\{(x,\pi) : \Pr_r[V(x,\pi,r) = 1] \leq 1/3\right\} ,$$

where in the expressions $V(x,\pi,r)$ above x is an input and π is a proof and r is a random string. Note that the promise-problem $(\mathsf{Y}_V, \mathsf{N}_V)$ can be decided in probabilistic linear time.

Let $\epsilon' = \epsilon/c$ for a sufficiently large constant $c > 1$, and let $k(n) = n^{\epsilon'}$. Let $f = f^{(\epsilon')}$ be the corresponding function from our hypothesis, and note that the upper bound in the first item is $\bar{T} = T \cdot k$, whereas the lower bound in the second item is $\bar{T} \cdot k^{1+\beta}$.

First step: Reduce the number of random coins to n^{η}. For a sufficiently small constant $\mu = \mu(\epsilon') > 0$ that will be determined later, let G^{crypto} be the PRG from Theorem 3.15, instantiated with stretch $n^{\mu} \mapsto T(n)$. Consider the verifier V' that uses only n^{μ} coins and is defined by $V'(x,\pi,s) = V(x,\pi,G^{\mathsf{crypto}}(s))$. Note that for every fixed x,π we have that $\Pr_s[V'(x,\pi,s) = 1] \in \Pr_r[V(x,\pi,r) = 1] \pm n^{-\omega(1)}$. Hence, $\mathsf{Y}_{V'} = \mathsf{Y}_V$, where $\mathsf{Y}_{V'} \overset{\text{def}}{=\!=} \{(x,\pi) : \Pr_r[V'(x,\pi,r) = 1] \geq .66\}$, and similarly $\mathsf{N}_{V'} = \mathsf{N}_V$ where $\mathsf{N}_{V'} \overset{\text{def}}{=\!=} \{(x,\pi) : \Pr_r[V'(x,\pi,r) = 1] \leq .33\}$. The running time of V' is $O(T^{1+\mu})$ and its number of random coins is n^{μ}.

Main step: Targeted PRG using the transcript as a source of hardness. Now, let D be the following deterministic verifier. On input $x \in \{0,1\}^n$ and proof $\pi \in \{0,1\}^T$, consider the generator from Theorem 7.1, instantiated with parameters

$$N = n + T\,, \qquad \bar{T}(N) = T(n) \cdot k\,,$$
$$g = f^{\epsilon} \colon \{0,1\}^N \to \{0,1\}^{n^{1+\epsilon'}}\,, \qquad \text{sufficiently small } \alpha', \beta', \eta \quad ;$$

we now also fix the parameter μ of G^{crypto} above to be such that $k^{\eta} = n^{\mu}$. The verifier D runs G_g to obtain a set of $k \cdot \bar{T} + \mathrm{poly}(k) \leq T \cdot \mathrm{poly}(k)$ strings of length n^{μ} denoted $s_1, \ldots, s_{T \cdot \mathrm{poly}(k)}$ and outputs $\mathsf{MAJ}_{i \in [T \cdot \mathrm{poly}(k)]}\{V'(x,\pi,s_i)\}$. Assuming that the constant $c > 1$ is sufficiently large, the running time of this algorithm is at most $T \cdot n^{\epsilon}$.

Analysis. The honest prover for D is identical to that of V. We now show an algorithm F that runs in time $\bar{T} \cdot n^{\beta}$, and for every fixed $(x,\pi) \in \mathsf{Y}_V$ such that $D(x,\pi) = 0$ it holds that $F(x,\pi)$ α-approximates $f(x,\pi)$. By a symmetric argument (which is identical and omitted), there exists another algorithm F' with precisely the same guarantee for any $(x,\pi) \in \mathsf{N}_V$ such that $D(x,\pi) = 1$. By our hypothesis, for every polynomial-time samplable distribution $\mathbf{P}$ over $\{0,1\}^{n+T}$, with probability at least $1 - n^{-\omega(1)}$ over choice of $(x,\pi) \sim \mathbf{P}$ both algorithms F and F' fail to α-approximate $f(x,\pi)$. Hence, the probability that a probabilistic polynomial-time algorithm can find (x,π) such that $D(x,\pi)$ errs is at most $n^{-\omega(1)}$.

Thus, it is left to construct the algorithm F. Fix $(x,\pi) \in \mathsf{Y}_V$ such that $D(x,\pi) = 0$, and denote by $D_{x,\pi} \colon \{0,1\}^{k^{\eta}} \to \{0,1\}$ the function $D_{x,\pi}(r) = V(x,\pi,r)$. By our assumption $D_{x,\pi}$ is a $(1/10)$-distinguisher for the uniform distribution over the output-set of G_g. We invoke the reconstruction Rec from Theorem 7.1; the running time of algorithm, accounting for answering its $\tilde{O}(k^{1+\beta'})$ queries to $D_{x,\pi}$, is

$$\tilde{O}(k^{1+\beta'}) + k^{\beta'} \cdot \bar{T}(n) + \tilde{O}(k^{1+\beta'}) \cdot T < \tilde{O}(T \cdot k \cdot k^{\beta'}) < \bar{T} \cdot n^{\beta} ,$$

for a sufficiently small choice of β'; and with probability $1 - o(1) > 1 - \alpha/2$ the reconstruction outputs a string string that agrees with $f(x, \pi)$ on at least $1 - \alpha' > 1 - \alpha/2$ of the bits, for a sufficiently small $\alpha' > 0$. ∎

Note that the proof above works as-is even if the initial $\mathsf{de}\mathcal{IP}_{MA}^{[\rightleftharpoons 2]}$ system has imperfect completeness.

7.2 Basic case: Doubly efficient proof systems with few random coins

The main goal in this section is to state and prove Theorem 1.7, which asserts that under strong hardness assumptions, we can simulate every $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}$ protocol by a deterministic doubly efficient argument system, with essentially no time overhead.

In Section 7.2.1 we state Theorem 1.7 and discuss its hypothesis, and then in Section 7.2.2 we prove the result. In Section 7.2.3 we state and prove a strong version of the result that holds for doubly efficient proof systems that have an efficient univesal prover, and in Section 7.2.4 we deduce Theorem 1.4 as a corollary of the latter.

7.2.1 The result statement and a discussion of the hypothesis

The following is a generic form of the hardness hypothesis that we will use in Theorem 1.7. It is inspired by a hardness assumption from our previous work [CT21a], but the current assumption is stronger because it refers to probabilistic *oracle machines* (rather than just to probabilistic algorithms). In the assumption we fix a "complexity parameter" $n \in \mathbb{N}$, and think of all other parameters as functions of n.

Assumption 7.4 (non-batch-computability assumption; Assumption 1.6, restated)**.** *For $N, K, K', \bar{T} \colon \mathbb{N} \to \mathbb{N}$ and $\eta \colon \mathbb{N} \to (0,1)$, the $(N \mapsto K, K', \eta)$-*non-batch-computability *assumption for time $\bar{T}$ with oracle access to $\mathcal{O}$ is the following. There exists $f \colon \{0,1\}^* \to \{0,1\}^*$ that for every $n \in \mathbb{N}$ maps $N(n)$ bits to $K(n)$ bits and satisfies:*

1. *There exists a deterministic algorithm that gets input $(z, i) \in \{0,1\}^{N(n)} \times [K(n)]$ and outputs the i^{th} bit of $f(z)$ in time $\bar{T}(n)$.*
2. *For every oracle machine M running in time $\bar{T} \cdot K'$ and having oracle access to $\mathcal{O}$, and every collection $\mathbf{z} = \left\{\mathbf{z}_{N(n)}\right\}_{n \in \mathbb{N}}$ of distributions such that $\mathbf{z}_{N(n)}$ is over $\{0,1\}^{N(n)}$ and can be sampled in time polynomial in $\bar{T}(n)$, and every sufficiently large $n \in \mathbb{N}$, with probability at least $1 - \bar{T}(n)^{-\omega(1)}$ over choice of $z \sim \mathbf{z}_{N(n)}$ it holds that $M^{\mathcal{O}}$ fails to approximately print $f(z)$ with error $\eta(n)$.*

We will use Assumption 7.4 with $\bar{T}$ that is a polynomial, in which case the error bound $\bar{T}(n)^{-\omega(1)}$ is just $n^{-\omega(1)}$; that is, the error bound is any negligible function in the input length. We are now ready to state Theorem 1.7.

Theorem 7.5 (derandomizing constant-round doubly efficient proof systems into deterministic doubly efficient argument systems, with almost no overhead; Theorem 1.7, restated). *For every $\alpha, \beta \in (0,1)$ there exists $\eta > 0$ such that the following holds. Let $c \in \mathbb{N}$ be a constant, let $T(n)$ be a polynomial, let $R(n) < T(n)$ be time-computable, and let $N(n) = n + c \cdot T(n)$ and $K(n) = R(n)^{1/\eta}$. Assume that the $(N \mapsto K, K^{\beta}, \alpha)$-non-batch-computability assumption holds for time $\bar{T} = T \cdot K$ with oracle access to $pr\mathcal{AMTIME}_2^{[\rightleftharpoons c]}[n]$ on inputs of length $O(T)$. Then,*

$$\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T, R] \subseteq \mathsf{de}\mathcal{DARG}[T \cdot R^{O(c/\eta)}] ,$$

where the O-notation hides a universal constant.

To discuss the hardness hypothesis, let us fix the parameter value $R = T^{o(1)}$, which will be the value we use in the proofs of Theorems 1.4 and 1.8. Then, loosely speaking, the assumption in Theorem 7.5 can be described as follows: There exists a function f from T bits to $K = T^{o(1)}$ bits such that –

1. Each output bit of f can be computed in time $\bar{T} = T \cdot K = T^{1+o(1)}$.
2. The entire string $f(x)$ cannot be (approximately) printed in probabilistic time $\bar{T} \cdot K^{\beta}$, while making oracle queries of length T to a $pr\mathcal{AMTIME}^{[\rightleftharpoons c]}$ protocol that runs in time linear in T. (And this hardness holds whp over any polynomial-time samplable distribution of T-bit strings x.)

Our proof allows relaxing the hypothesis further: For example, the oracle machine against which we assume hardness only makes *non-adaptive* queries to its oracle, and the number of queries is only $\text{poly}(k)$. (We avoid including these relaxations in the assumption statement for simplicity of presentation.)

The difference from known results about batch-computability. The complexity of the oracle machine for which we assume hardness is reminiscent of the complexity of *unconditionally known* interactive protocols for batch-computing functions. However, there is a crucial difference between the two settings, which we now explain.

For any $f\colon \{0,1\}^T \to \{0,1\}^K$ whose individual output bits are computable in time $\bar{T}$, Reingold, Rothblum, and Rothblum [RRR18, Theorem 13] (following their previous result in [RRR21]) constructed a constant-round protocol for batch verification of the K output bits of f such that the verifier runs in time $\tilde{O}(K \cdot T) + K^{\beta} \cdot \bar{T}^{1+\beta} = O(K^{\beta} \cdot \bar{T}^{1+\beta})$, where $\beta > 0$ is an arbitrarily small constant.[42]

The running time of their protocol almost matches (from above) the complexity of our oracle machine, which might seem alarming. However, we stress that our oracle machine is not an interactive protocol by itself, but rather only *makes queries* to a protocol; these queries are of length $T = \bar{T}/K$, and *the protocol is only allowed linear running time $O(T)$*. Therefore, the main point of difference is that the interactive proof oracle in our hypothesized lower bound *is allowed less running time than the upper bound $\bar{T}$ on computing even a single output bit of f*. Needless to say, the techniques for batch-verification from [RRR21; RRR18] do not extend to such a setting.

In fact, a hardness assumption that is even stronger than the one we make still makes sense: The hypothesis would still seem reasonable if the probabilistic machine would make queries of length T to a linear-space oracle (i.e., to a machine running in space $O(T)$ and time $2^{O(T)}$), rather than to a linear-time interactive proof.

[42]The result of [RRR18] applies in a more general setting than the one we compare to, since they consider applying f to K different inputs $x_1, \ldots, x_k$, since their prover is efficient (i.e. runs in time $\text{poly}(\bar{T})$), and since their result holds even when the upper bound on the complexity of f is $\mathcal{UP}$ rather than $\mathcal{P}$.

7.2.2 Proof of Theorem 7.5

Let $L \in \mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T,R]$, and let V be a T-time verifier in a protocol with c rounds for L such that V sends R random coins in each turn. Denote by $c' = \lceil c/2 \rceil$ be the number of turns of the verifier in the interaction. Given any prover P and input x, we think of the corresponding interaction as a function of c' strings of R random coins that are chosen (uniformly) in advance, but are revealed to P during the interaction. We denote by

$$\langle V,P,x\rangle\,(r_1,\ldots,r_{c'})$$

the result of the interaction between V and P on input x with random coins $r_1,\ldots,r_{c'}$.

A verifier for L with $O(\log(R))$ random coins. Our goal now is to construct an alternative verifier V' for L such that in each round of interaction, V' uses $O(\log(R))$ random coins. Let $G = G_f$ be the generator from Theorem 7.1, instantiated with the function $f\colon \{0,1\}^N \to \{0,1\}^{R^{1/\eta}}$, with sufficiently small $\beta' < \beta$ and $\alpha' < \alpha$, and with output length R; the number of output strings of G_f is $\bar{R} \overset{\text{def}}{=} \text{poly}(K) = R^{O(1/\eta)}$, where $\eta = \eta_{\alpha',\beta'}$ is the parameter from Theorem 7.1, and its running time is $\bar{T}\cdot\text{poly}(K) = T\cdot R^{O(1/\eta)}$. For any given $z \in \{0,1\}^N$ and $s \in [\bar{R}]$, let $G^z(s) = G(z)_s \in \{0,1\}^R$ be the s^{th} output string of G when G is given input z.

In turn $i \in [c']$, the verifier V' chooses a random seed $s \in [\bar{R}]$ and sends $G^{x,\pi_i}(s)$ to the prover, where x is the input and π_i is the interaction communicated between the parties up to turn i of the interaction, padded with zeroes to be of length precisely $N-n$ (for convience we denote $\pi_1 = 0^{N-n}$). Then it applies the same final predicate to the interaction as V. In other words, continuing our notation for interactions as functions of random coins, we have that

$$\langle V',P,x\rangle\,(s_1,\ldots,s_{c'}) = \langle V,P,x\rangle\,(G^{x,\pi_1}(s_1),\ldots,G^{x,\pi_{c'}}(s_{c'}))\,.$$

The running time of V' is larger than that of V, but we will carefully account for it later on. For now it suffices to observe that the only runtime overhead of V' on top of V comes from computing the generator G in each of the c' turns, rather than just sending random coins.

Analysis: Soundness of V'. For convenience, we will think of any probabilistic polynomial-time algorithm $\mathbf{P}$ as an efficiently samplable distribution over *deterministic* polynomial-time algorithms, obtained in the natural way (i.e., by randomly choosing coins in advance and running the algorithm with the fixed chosen coins).

Consider an arbitrary probabilistic polynomial-time algorithm $\mathbf{P}$. Our analysis uses the following hybrid argument. Let V_0 be the original verifier, and for $i \in [c']$ let V_i be the verifier in which in the first i turns, the verifier uses $\log(\bar{R})$ coins as above, instead of $R(n)$ random coins; that is, for any fixed $P \sim \mathbf{P}$,

$$\langle V_i,P,x\rangle\,(s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'}) = \langle V,P,x\rangle\,(G^{x,\pi_1}(s_1),\ldots,G^{x,\pi_i}(s_i),r_{i+1},\ldots,r_{c'})\,.$$

Observe that $V_0 = V$ and that $V_{c'} = V'$. For any fixed prover P and $1 \le i < j \le c'$, denote by $P^{i\ldots j}$ the partial prover strategy of P that refers only to rounds $i,i+1,\ldots,j$.[43] That is, we think of the prover as a sequence of c' functions, where the

[43]Note that we are now referring to *rounds* rather than to *turns*. That is, the interaction consists of c turns (where a turn is when one of the players speaks) and of $c' = \lceil c/2 \rceil$ rounds (where a round consists of two turns, except possibly the last round).

i^{th} function maps a transcript in the i^{th} round (which is of length $i \cdot R + (i-1) \cdot T$) to the prover's response (which is of length T); then, $P^{i...j}$ is simply a subsequence of the c' functions that define P. We use the notation $P^{i...j} \sim \mathbf{P}$ to denote the random variable that is obtained by sampling $P \sim \mathbf{P}$ and outputting the partial prover strategy $P^{i...j}$. And for two partial prover strategies $P^{1...i-1}$ and $P^{i...c'}$, we denote by $P^{1...i-1} \circ P^{i...c}$ the (complete) prover strategy that is obtained by combining both partial strategies in the obvious way (i.e., by simply concatenating the two sequences of functions).

Our main definition for the hybrid argument is a sequence of hybrids that involves both a partial replacement of the random coins, and a partial replacement of the "existential" prover strategy in the soundness condition (i.e., that there does not exist an all-powerful prover that convinces the verifier) with a partial strategy chosen according to $\mathbf{P}$. In more detail, for any fixed $i \in [c']$ we denote

$$p_{x,i} = \max_{P^{i+1...c'}} \left\{ \Pr_{s_1,...,s_i,r_{i+1},...,r_{c'},P^{1...i} \sim \mathbf{P}} \left[\left\langle V_i, P^{1...i} \circ P^{i+1,...,c'}, x \right\rangle (s_1, \ldots, s_i, r_{i+1}, \ldots, r_{c'}) = 1 \right] \right\} .$$

The perfect completeness of V' follows immediately from the perfect completeness of V, since V' simply simulates V with pseudorandom choices of coins. Thus, we focus on establishing the soundness of V'. To do so, we first argue that:

Fact 7.5.1. *For any fixed $x \notin L$ such that*

$$\Pr_{P \sim \mathbf{P}, s_1,...,s_{c'}} [\langle V', P, x \rangle (s_1, \ldots, s_{c'}) = 1] > .4 \, , \tag{7.1}$$

there exists $i \in [c']$ such that $p_{x,i} - p_{x,i-1} > 1/20c'$.

Proof. Fix $x \notin L$ such that Eq. (7.1) holds. Observe that $p_{x,0} \leq 1/3$, by the soundness of the original protocol V (which holds for every fixed x and an all-powerful P). On the other hand, we have that $p_{x,c'} = \Pr_{P \sim \mathbf{P}, s_1,...,s_{c'}} [\langle V', P, x \rangle (s_1, \ldots, s_{c'}) = 1]$, and thus $p_{x,c'} > .4$ by Eq. (7.1). Since $p_{x,c'} - p_{x,0} = \sum_{i \in [c']} p_{x,i} - p_{x,i-1}$, for some $i \in [c']$ it holds that $p_{x,i} - p_{x,i-1} > (p_{x,c'} - p_{x,0})/c' > 1/20c'$. □

The main claim in the analysis is the following:

Claim 7.5.2. *For any $i \in [c']$, the probability over $x \sim \mathbf{x}$ that $x \notin L$ and*

$$p_{x,i} > p_{x,i-1} + 1/20c'$$

is negligible.

Since the proof of Claim 7.5.2 is quite involved, we defer it for a moment, and first complete the argument assuming that Claim 7.5.2 is true.

By combining Fact 7.5.1 and Claim 7.5.2, we deduce that for every probabilistic polynomial-time algorithm $\mathbf{P}$, with probability $1 - \mathsf{neg}(|x|)$ over $x \sim \mathbf{x}$, if $x \notin L$ then $\Pr_{P \sim \mathbf{P}, s_1,...,s_{c'}} [\langle V', P, x \rangle (s_1, \ldots, s_{c'}) = 1] \leq .4$. (Intuitively, this establishes that V' is an argument system that uses few random coins.)

From V' to a deterministic doubly efficient argument system. Given any input x, the total number of possible messages from V' to a prover (across all turns) is $\bar{R}^{c'}$, corresponding to a choice of seeds $\bar{s} \in [\bar{R}]^{c'}$. Our deterministic verifier V'' expects the prover to send a corresponding transcript for each choice of $\bar{s}$. It then:

1. Verifies that the sent transcripts are consistent with a prover strategy (i.e., that the prover did not respond to two identical prefixes in different ways).

2. For each transcript corresponding to a choice of $\bar{s}$, it verifies that the pseudorandom coins in each message (which are part of the transcript) are what one obtains by applying G to the transcript at that point, using the corresponding seed (which is part of $\bar{s}$).

3. Computes the average acceptance probability of V' over all choices of $\bar{s}$.

Recall that there is an efficient prover P such that for any $x \in L$ it holds that $\Pr_{r_1,\ldots,r_{c'}}[\langle V, P, x\rangle (r_1,\ldots,r_{c'}) = 1] = 1$. When interacting with P, the verifier V' simply uses a pseudorandom coins instead of random ones, and thus $\Pr_{s_1,\ldots,s_{c'}}[\langle V', P, x\rangle (s_1,\ldots,s_{c'}) = 1] = 1$. The honest prover for the verifier V'' enumerates over $\bar{s} \in [\bar{R}]^{c'}$, and for every choice of $\bar{s}$ it simulates the corresponding interaction with P, while computing the generator G at each round. Since P runs in time $\mathrm{poly}(T)$ and G can be computed in time polynomial in $\bar{T} \le \mathrm{poly}(T)$, the honest prover for V'' runs in time $\mathrm{poly}(T)$.

Now, the verifier V'' inherits its soundness immediately from that of V'.[44] Relying on the fact that the soundness holds against all polynomial-time algorithms $\mathbf{P}$, we further argue that the soundness error of V'' is not only .4 but actually $\mathsf{neg}(n)$:

Claim 7.5.3. *For every probabilistic polynomial-time algorithm* $\mathbf{P}$ *with probability* $1 - \mathsf{neg}(n)$ *over an n-bit* $x \sim \mathbf{x}$, *if* $x \notin L$ *then* $\Pr_{P\sim\mathbf{P}}[V(x, P(x)) = 1] < \mathsf{neg}(n)$.

Proof. Assume towards a contradiction that for some $\mathbf{P}$ and polynomial $p(n)$ there are infinitely many $n \in \mathbb{N}$ such that, with non-negligible probability over an n-bit $x \sim \mathbf{x}$ it holds that $x \notin L$ and $\Pr_{P\sim\mathbf{P}}[V''(x, P(x)) = 1] \ge 1/p(n)$.

Consider the prover $\mathbf{P}'$ that on input x runs $\mathbf{P}$ for $t = p(n)^2$ times to obtain candidate proofs $\pi_1, \ldots, \pi_t$, for each $i \in [t]$ checkes whether $V''(x, \pi_i) = 1$, and if it finds π_i for which the latter holds then it prints this π_i (otherwise it prints some fixed default proof). Note that $\mathbf{P}'$ runs in polynomial time, and for every x such that $\Pr_{P\sim\mathbf{P}}[V''(x, P(x)) = 1] \ge 1/p(n)$ we have that $\Pr_{P'\sim\mathbf{P}'}[V''(x, P'(x)) = 1] \ge .99$. Thus, there is a polynomial-time prover $\mathbf{P}'$ and infinitely many $n \in \mathbb{N}$ such that with non with non-negligible probability over an n-bit $x \sim \mathbf{x}$ it holds that $x \notin L$ and $\Pr_{P\sim\mathbf{P}}[V''(x, P(x)) = 1] > .4$, contradicting the soundness of V''. ■

We now bound the running time of V''. Recall that the prover sends $[\bar{R}]^{c'}$ transcripts to V'', corresponding to all possible choices of seeds $\bar{s} \in [\bar{R}]^{c'}$ by V'. We assume that the prover sends the transcripts in prefix-order of $\bar{s}$. For each prefix $s_1, \ldots, s_i$ of $\bar{s}$, the verifier checks that all transcripts corresponding to that prefix are identical (i.e., verifies that the prover strategy is consistent); and then computes the set of pseudorandom strings obtained by applying G to the foregoing transcript with all possible choices of s_{i+1}, "in a batch"; by Theorem 7.1, for each prefix this can be done in time $\bar{T} \cdot K + \mathrm{poly}(K) \le T \cdot R^{O(1/\eta)}$. In the end, it computes the original V on each of the $|\bar{R}|^{c'} = R^{O(c/\eta)}$ choices of seeds. The final running time of V'' is thus $T \cdot R^{O(c/\eta)}$.

Finally, to complete the proof the only missing piece is to prove Claim 7.5.2.

Proof of Claim 7.5.2. Fix $i \in [c']$ and assume that with non-negligible probability over $x \sim \mathbf{x}$ we have that $p_{x,i} > p_{x,i-1} + 1/20c'$. For any fixed $\bar{p} = P^{1\ldots i-1}$ and $\bar{s} = s_1, \ldots, s_{i-1}$,

[44]To see this, note that if the prover sends consistent answers to all message sequences then those answers yield a prover strategy that could have been used in the interaction with V'.

the interaction in the first $i-1$ rounds between any prover whose partial strategy in these rounds is $\bar{p}$ and any verifier that behaves like V_{i-1} in these rounds is also fixed (i.e., it is a deterministic function of $\bar{p}$ and $\bar{s}$); we denote the transcript of this interaction by $\pi_{\bar{p},\bar{s}}$. We denote by $(\mathbf{x}, \boldsymbol{\pi})$ the distribution that is obtained by sampling $x \sim \mathbf{x}$ and $\bar{p} = P^{1\ldots i-1} \sim \mathbf{P}$ and $\bar{s} \in ([\bar{R}])^{i-1}$ and outputting $(x, \pi_{\bar{p},\bar{s}})$. By our assumptions about $\mathbf{P}$ and $\mathbf{x}$ and the fact that G runs in polynomial time, the distribution $(\mathbf{x}, \boldsymbol{\pi})$ is samplable in polynomial time.

Fixing the first $i-1$ rounds of interaction. For every fixed $(x, \pi_{\bar{s},\bar{p}})$, denote

$$(p_{x,i-1}|\pi_{\bar{s},\bar{p}}) = \max_{P^{i\ldots c'}} \left\{ \Pr_{r_i, r_{i+1}, \ldots, r_{c'}} \left[\left\langle V_{i-1}, \bar{p} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, r_i, r_{i+1}, \ldots, r_{c'}) \right] \right\} ,$$

$$(p_{x,i}|\pi_{\bar{s},\bar{p}}) = \max_{P^{i+1\ldots c'}} \left\{ \Pr_{s_i, r_{i+1}, \ldots, r_{c'}, P^i \sim \mathbf{P}} \left[\left\langle V_i, \bar{p} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, s_i, r_{i+1}, \ldots, r_{c'}) \right] \right\} ,$$

and observe that:

Claim 7.5.3.1. *We have that* $p_{x,i} = \mathbb{E}_{\bar{s},\bar{p}} \left[p_{x,i}|\pi_{\bar{s},\bar{p}} \right]$ *and* $p_{x,i-1} = \mathbb{E}_{\bar{s},\bar{p}} \left[p_{x,i-1}|\pi_{\bar{s},\bar{p}} \right]$.

Proof. Note that in $p_{x,i}$ and in $(p_{x,i}|\pi_{\bar{s},\bar{p}})$ (resp., in $p_{x,i-1}$ and $(p_{x,i-1}|\pi_{\bar{s},\bar{p}})$) the maximum is over functions $P^{i+1\ldots c'}$ (resp., $P^{i\ldots c'}$) that take $\pi_{\bar{s},\bar{p}}$ as part of their input. Thus, first choosing $\pi_{\bar{s},\bar{p}}$ from some distribution and then taking the maximum-achieving function is identical to first taking the maximum-achieving function and then choosing $\pi_{\bar{s},\bar{p}}$ from the same distribution.[45] □

We call a pair $(x, \pi_{\bar{s},\bar{p}})$ good if $(p_{x,i}|\pi_{\bar{s},\bar{p}}) > (p_{x,i-1}|\pi_{\bar{s},\bar{p}}) + 1/40c'$, and argue that:

Claim 7.5.3.2. *With non-negligible probability over* $(x, \pi_{\bar{s},\bar{p}}) \sim (\mathbf{x}, \boldsymbol{\pi})$ *we have that* $(x, \pi_{\bar{s},\bar{p}})$ *is good.*

Proof. By our assumption, with non-negligible probability over $x \sim \mathbf{x}$ we have that $p_{x,i} > p_{x,i-1} + 1/20c'$. Now, let $\Delta_{\bar{s},\bar{p}} = (p_{x,i}|\pi_{\bar{s},\bar{p}}) - (p_{x,i-1}|\pi_{\bar{s},\bar{p}})$, and note that

$$\mathbb{E}_{\bar{s},\bar{p}}[\Delta_{\bar{s},\bar{p}}] = \mathbb{E}_{\bar{s},\bar{p}} \left[p_{x,i}|\pi_{\bar{s},\bar{p}} \right] - \mathbb{E}_{\bar{s},\bar{p}} \left[p_{x,i-1}|\pi_{\bar{s},\bar{p}} \right] = p_{x,i} - p_{x,i-1} > 1/20c' ,$$

where the second equality above relies on Claim 7.5.3.1. Thus, if we'd have that $\Pr_{\bar{s},\bar{p}}[\Delta_{\bar{s},\bar{p}} > 1/40c'] < n^{-\omega(1)}$, then we'd also have $\mathbb{E}_{\bar{s},\bar{p}}[\Delta_{\bar{s},\bar{p}}] \leq n^{-\omega(1)} \cdot (1/40c') + (1/40c') < 1/20c'$, a contradiction. □

Obtaining an $\mathcal{AMTIME}_2^{[\rightleftharpoons c]}$ distinguisher. Fixing a good $(x, \pi_{\bar{s},\bar{p}})$, we denote

$$p(r_i) = \max_{P^{i\ldots c'}} \left\{ \Pr_{r_{i+1}, \ldots, r_{c'}} \left[\left\langle V_{i-1}, \bar{p} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, r_i, \ldots, r_{c'}) = 1 \right] \right\} ,$$

and argue that:

[45] In other words, we use the fact that for every $\mathcal{D}$ and $\mathcal{R}$ and $\nu \colon \mathcal{R} \to \mathbb{R}$ it holds that

$$\max_{f \colon \mathcal{D} \to \mathcal{R}} \left\{ \mathbb{E}_{r \in \mathcal{D}} [\nu(f(r))] \right\} = \mathbb{E}_{r \in \mathcal{D}} \left[\max_{f \colon \mathcal{D} \to \mathcal{R}} \{\nu(f(r))\} \right] = \mathbb{E}_{r \in \mathcal{D}} [\nu(f^*(r))] ,$$

where f^* is the function that maps any $r \in \mathcal{D}$ to $t = f(r)$ such that $\nu(t)$ is maximal.

Claim 7.5.3.3. *The following two statements hold:*

1. $\mathop{\mathbb{E}}_{r_i \in \{0,1\}^R} [p(r_i)] = (p_{x,i-1} | \pi_{\bar{s},\bar{p}})$.
2. $\mathop{\mathbb{E}}_{s_i \in [\bar{R}]} [p(G^{x,\pi_{\bar{s},\bar{p}}}(s_i))] \geq (p_{x,i} | \pi_{\bar{s},\bar{p}})$.

Proof. For the first item, note that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{r_i \in \{0,1\}^R} [p(r_i)] \\
&= \mathop{\mathbb{E}}_{r_i \in \{0,1\}^R} \left[\max_{P^{i\ldots c'}} \left\{ \Pr_{r_{i+1},\ldots,r_{c'}} \left[\left\langle V_{i-1}, \bar{p} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, r_i, \ldots, r_{c'}) = 1 \right] \right\} \right] \\
&= \max_{P^{i\ldots c'}} \left\{ \Pr_{r_i, r_{i+1},\ldots,r_{c'}} \left[\left\langle V_{i-1}, \bar{p} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, r_i, \ldots, r_{c'}) = 1 \right] \right\} \\
&= (p_{x,i-1} | \pi_{\bar{s},\bar{p}}) \ ,
\end{aligned}$$

where the second inequality is because (as in the proof of Claim 7.5.3.2) the maximum is over functions $P^{1\ldots c'}$ that take r_i as part of their input (and thus first drawing a random r_i and then choosing a maximum-achieving function is identical to first choosing a maximum-achieving function and then drawing a random r_i).

For the second item, the argument is a bit more subtle, as follows:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{s_i \in [\bar{R}]} [p(G^{x,\pi_{\bar{s},\bar{p}}}(s_i))] \\
&= \mathop{\mathbb{E}}_{s_i \in [\bar{R}]} \left[\max_{P^{i\ldots c'}} \left\{ \Pr_{r_{i+1},\ldots,r_{c'}} \left[\left\langle V_{i-1}, \bar{p} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, G^{x,\pi_{\bar{s},\bar{p}}}(s_i), r_{i+1}, \ldots, r_{c'}) = 1 \right] \right\} \right] \\
&= \max_{P^{i\ldots c'}} \left\{ \Pr_{s_i, r_{i+1},\ldots,r_{c'}} \left[\left\langle V_{i-1}, \bar{p} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, G^{x,\pi_{\bar{s},\bar{p}}}(s_i), r_{i+1}, \ldots, r_{c'}) = 1 \right] \right\} \\
&= \max_{P^{i\ldots c'}} \left\{ \Pr_{s_i, r_{i+1},\ldots,r_{c'}} \left[\left\langle V_i, \bar{p} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, s_i, r_{i+1}, \ldots, r_{c'}) = 1 \right] \right\} \\
&\geq \max_{P^{i+1\ldots c'}} \left\{ \Pr_{s_i, r_{i+1},\ldots,r_{c'}, P^i \sim \mathbf{P}} \left[\left\langle V_i, \bar{p} \circ P^{i\ldots c'}, x \right\rangle (\bar{s}, s_i, r_{i+1}, \ldots, r_{c'}) = 1 \right] \right\} \\
&= (p_{x,i} | \pi_{\bar{s},\bar{p}}) \ ,
\end{aligned}$$

where the main difference from the proof of the first item is the inequality, which asserts that taking the maximum-achieving prover strategy in round i can only increase the acceptance probability compared to choosing $P_i \sim \mathbf{P}$. [46] □

Relying on Claim 7.5.3.3, for the fixed good $(x, \pi_{\bar{s},\bar{p}})$ we have that

$$\mathop{\mathbb{E}}_{s_i \in [\bar{R}]} [p(G^{x,\pi_{\bar{s},\bar{p}}}(s_i))] - \mathop{\mathbb{E}}_{r_i \in \{0,1\}^R} [p(r_i)] \geq (p_{x,i} | \pi_{\bar{s},\bar{p}}) - (p_{x,i-1} | \pi_{\bar{s},\bar{p}}) > 1/40c' \ .$$

The foregoing assertion implies that the real-valued function $p(\cdot)$ behaves differently on the pseudorandom distribution $G^{x,\pi_{\bar{s},\bar{p}}}(\mathbf{u}_{[\bar{R}]})$ and on the uniform distribution

[46] To see that the third equality above holds, note the following. In the upper row we are referring to the verifier V_{i-1}, which uses the i^{th} random string given to it as-is, and are giving V_i the string $G^{x,\pi_{\bar{s},\bar{p}}}(s_i)$ for a random s_i. In the bottom row we're referring to the verifier V_{i-1}, which applies $G^{x,\pi_{\bar{s},\bar{p}}}$ to the i^{th} random string given to it, and are giving V_{i-1} a random s_i. Thus, in both cases the random string used in the i^{th} round is $G^{x,\pi_{\bar{s},\bar{p}}}(s_i)$ for a random s_i.

$\mathbf{u}_T$. To obtain a Boolean-valued distinguisher (rather than a real-valued one such as $p(\cdot)$), and furthermore a Boolean-valued distinguisher that is computable by an $\mathcal{AMTIME}^{[\rightleftharpoons c]}$ protocol, we rely on the following claim: It asserts that if two real-valued RVs $\mathbf{x}$ and $\mathbf{y}$ in $[0,1]$ have expectations that differ by $\epsilon > 0$, then there are two thresholds ℓ' and $\ell' + \Theta(\epsilon)$ such that the probability that $\mathbf{x}$ exceeds $\ell' + \Theta(\epsilon)$ is noticeably higher than the probability that $\mathbf{y}$ exceeds ℓ'.

Lemma 7.5.3.4. *Let $\mathbf{x}, \mathbf{y}$ be two random variables taking values from $[0,1]$ and $\epsilon \in (0,1)$ such that $\epsilon^{-1} \in \mathbb{N}$. If $\mathbb{E}[\mathbf{x}] - \mathbb{E}[\mathbf{y}] > \epsilon$, then there exists $j \in [4/\epsilon]$ such that*

$$\Pr[\mathbf{x} \geq (\epsilon/4) \cdot (j+1)] - \Pr[\mathbf{y} \geq (\epsilon/4) \cdot j] > \epsilon/2.$$

The proof of Lemma 7.5.3.4 is elementary but tedious, so for convenience we defer it to Appendix B. Now, denote $\epsilon = 1/40c'$ and consider the following promise problem:

$$\begin{aligned} \mathsf{Y} &= \left\{(x, \pi_{\bar{s},\bar{p}}, r_i, \tau) : p(r_i) \geq \tau + \epsilon/4\right\} \\ \mathsf{N} &= \left\{(x, \pi_{\bar{s},\bar{p}}, r_i, \tau) : p(r_i) \leq \tau\right\} \ . \end{aligned}$$

Observe that the problem (Y, N) can be decided in $pr\mathcal{AMTIME}_2^{[\rightleftharpoons c]}[O(n)]$, since the verifier can run the original protocol V_{i-1} from the definition of $p(\cdot)$ for $O(1)$ times in parallel, estimate the acceptance probability of V_{i-1} with the given prover up to accuracy $\epsilon/8$ and with high probability, and accept if and only if this probability is more than $\tau + \epsilon/8$. Note that the runtime of this protocol is linear, because $\epsilon = \Omega(1)$ and the input size to this problem is $O(T)$.

Now, by instantiating Lemma 7.5.3.4 with the RVs $\mathbf{x} = p(G^{x,\pi_{\bar{s},\bar{p}}}(\mathbf{u}_{[\bar{R}]}))$ and $\mathbf{y} = p(\mathbf{u}_R)$, for some $\tau \in \{0, \epsilon/4, 2\epsilon/4, \ldots, 1\}$ it holds that

$$\Pr[(x, \pi_{\bar{s},\bar{p}}, G^{x,\pi_{\bar{s},\bar{p}}}(\mathbf{u}_{[\bar{R}]}, \tau)) \in \mathsf{Y}] > 1 - \Pr[(x, \pi_{\bar{s},\bar{p}}, \mathbf{u}_R, \tau)) \in \mathsf{N}] + \epsilon/2\,.$$

For any fixed (x, π) and $\tau \in \{0, \epsilon/4, 2\epsilon/4, \ldots, 1\}$ we define the following procedure $D = D_{x,\pi,\tau}$: Given $r \in \{0,1\}^R$ (which we think of as coming either from the uniform distribution or from the pseudorandom distribution) the procedure creates the string $z = (x, \pi, r, \tau)$ and solves the promise problem (Y, N) on input z. Note that indeed $D \in pr\mathcal{AMTIME}_2^{[\rightleftharpoons c]}[O(n)]$, and that

$$\Pr[D(G^{x,\pi_{\bar{s},\bar{p}}}(\mathbf{u}_{[\bar{R}]})) = 1] > \Pr[D(\mathbf{u}_R) = 1] + \epsilon/2\,,$$

where the inequality relies on the fact that $\Pr[D(\mathbf{u}_R) = 1] = 1 - \Pr[D(\mathbf{u}_R) = 0 \geq 1 - \Pr[\mathbf{u}_T \in \mathsf{N}]$.

<u>The reconstruction argument.</u> To obtain a contradiction, we show an algorithm that runs in time $\bar{T} \cdot K^{\beta}$, where $\bar{T} = T \cdot K$, and with non-negligible probability over the polynomial-time samplable distribution $(x, \pi) \sim (\mathbf{x}, \boldsymbol{\pi})$ manages to approximately print $f(x, \pi)$ (with high probability over its random coins).

Consider an algorithm F_0 gets input (x, π) and randomly chooses $\tau \in \{0, \epsilon/4, 2\epsilon/4, \ldots, 1\}$. Then F_0 runs the reconstruction Rec with input (x, π), giving it oracle access to $D^{\tau,x,\pi}$. Specifically, whenever Rec queries $D^{\tau,x,\pi}$ on input r, the algorithm F_0 queries D^{τ} on input (x, π, r, τ) and returns the corresponding answer. The algorithm F_0 runs in time

$$\underbrace{\tilde{O}(K^{1+\beta'})}_{\text{\#queries}} \cdot \underbrace{T}_{\text{length of a query to } D^{\tau}} + \underbrace{\bar{T} \cdot K^{\beta'}}_{\text{add'l runtime of Rec}} = \tilde{O}(\bar{T} \cdot K^{\beta'})$$

and assuming that the guess of τ' was correct, with high probability F_0 prints a string that agrees with $f(x,\pi)$ on $1-\alpha'$ of the bits.

To construct an algorithm F that succeeds with high probability, we run F_0 for $O(c')$ times, such that with high probability at least one iteration successfully printed a string that agrees with $f(x,\pi)$ on $1-\alpha'$ of the bits. Then, for each of the $O(c')$ candidate strings, the algorithm F estimates the agreement of this string with $f(x,\pi)$ up to a sufficiently small constant error, by randomly sampling output bits and computing the corresponding bits of $f(x,\pi)$ (recall that each bit can be computed in time $\bar{T}$). The running time of F is at most $\tilde{O}(\bar{T}\cdot K^{\beta'}) < \bar{T}\cdot K^{\beta}$, and with high probability it succeeds in outputting a string that agrees with f on at least $1-\alpha$ of the bits. □

Having proved Claim 7.5.2, this concludes the proof of Theorem 7.5.

Remark 7.6 (handling imperfect completeness)**.** The proof of Theorem 7.5 implies similar derandomization for $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}$ protocols with *imperfect completeness* that use a small number of random coins. To see this, observe that the well-known transformation of $\mathcal{AM}$ protocols into protocols with perfect completeness [FGM+89] yields the following: Any $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}$ protocol in which the verifier uses only R coins can be simulated by a $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}$ protocol with perfect completeness such that the new protocol has the same number of rounds, the verifier uses $\bar{R}=\tilde{O}(R)$ random coins, its communication complexity and running time increase by a multiplicative factor of $O(\bar{R})$, and the prover is still efficient but it is now *probabilistic*.[47]

Since we think of R as small in the result above (indeed, we will use this result with $R=O(\log(T))$ in Theorems 7.9 and Corollary 7.12), we can start from a protocol with imperfect completeness, simulate it by a protocol with similar running time, and appeal to Theorem 7.5 as a black-box. Indeed, the only gap is that the honest prover in the resulting $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}$ protocol is now probabilistic rather than deterministic.

Remark 7.7 (argument systems for $\mathcal{NP}$-relations)**.** The honest prover in the argument system that is constructed in the proof of Theorem 7.5 is very similar to the original honest prover, and in particular has almost the same time complexity. (This is because the new prover enumerates over all possible seeds for the targeted PRG, across all rounds, and for each choice it simulates the verifier's interaction with the original prover using the chosen seeds to generate pseudorandom coins.) One implication of this fact is that our results extend naturally to constant-round probabilistic proof systems in which the honest prover gets a witness as auxiliary input.

In more detail, let R be a relation, let L_R be the decision problem defined by R, and let Π be a probabilistic proof system for L_R with c turns of interaction such that the verifier in Π runs in time T, and the honest prover in Π runs in time $\mathrm{poly}(T)$ when given a witness for the input (in the relation R).[48] Then, under the same assumption as in Theorem 7.5, our proof gives a deterministic argument system for L_R in which the verifier runs in time $T^{1+\epsilon}$, soundness is precisely as in Definition 3.8, and the honest prover runs in time $\mathrm{poly}(T)$ when given a witness (in the relation R) for the input.

[47]These properties are not explicitly stated in the original work but they readily follow from the proof. Specifically, in the original proof the message lengths by the verifier and the prover are coupled, but the proof only relies on the error being smaller than $1/R$; and the original proof shows that for every $x\in L$, with high probability over choice of initial strings by the prover (as a basis for simulating copies of the protocol in parallel) the verifier to accept with probability 1.

[48]That is, we consider the interaction between the verifier and the honest prover when the former is given input x and the latter is given input $(x,w)\in R$. Note that any such relation is in $\mathcal{MATIME}[\mathrm{poly}(T)]$, since a verifier can guess w and simulate the interaction with the honest prover.

7.2.3 A stronger result for doubly efficient proof systems with a universal prover

We now argue that in the special case of doubly efficient proof systems that have an efficient universal prover, we can derandomize such systems under a hypothesis that is weaker than the one in Theorem 7.5. Specifically, we require hardness only against probabilistic algorithms that have oracle access to $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}$, rather than to $\mathcal{AMTIME}^{[\rightleftharpoons c]}$. (We will use this generic claim in the proof of Theorem 1.4, since the proof system for #SAT indeed has an efficient universal prover.)

Theorem 7.8 (derandomizing constant-round doubly efficient proof systems with an efficient universal prover into deterministic doubly efficient argument systems)**.** *For every $\alpha, \beta \in (0,1)$ there exists $\eta > 0$ such that the following holds. Let $c \in \mathbb{N}$ be a constant, let $T(n)$ be a polynomial, let $R(n) < T(n)$ be time-computable, and let $N(n) = n + c \cdot T(n)$ and $K(n) = R(n)^{1/\eta}$. Assume that the $(N \mapsto K, K^{\beta}, \alpha)$-non-batch-computability assumption holds for time $\bar{T} = T \cdot K$ with oracle access to pr-$\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[n]$ on inputs of length $O(T)$. Then,*

$$\mathsf{de}\mathcal{IP}_{\mathsf{uni}}^{[\rightleftharpoons c]}[T, R] \subseteq \mathsf{de}\mathcal{DARG}[T \cdot R^{O(c/\eta)}] \;,$$

where the O-notation hides a universal constant, and $\mathsf{de}\mathcal{IP}_{\mathsf{uni}}^{[\rightleftharpoons c]}[T, R]$ denotes all problems solvable by $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T, R]$ systems that, for every constant $\mu > 0$, have a μ-approximate universal prover running in time $\mathrm{poly}(T)$.

Proof. The proof is very similar to that of Theorem 7.5, the only difference being that the reconstruction algorithm in Claim 7.5.2 can now only access a *pr*-$\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[n]$ oracle rather than a $\mathcal{AMTIME}_2^{[\rightleftharpoons c]}[n]$ oracle.

Recall that in the proof of Claim 7.5.2, the algorithm uses an oracle that solves the following promise problem, which is defined with respect to a parameter τ:

$$\begin{aligned}\mathsf{Y} &= \left\{(x, \pi_{\bar{s},\bar{p}}, r_i, \tau) : p(r_i) \geq \tau + \epsilon/4\right\} \\ \mathsf{N} &= \left\{(x, \pi_{\bar{s},\bar{p}}, r_i, \tau) : p(r_i) \leq \tau\right\} \;.\end{aligned}$$

Our goal is to argue that $(\mathsf{Y}, \mathsf{N}) \in$ *pr*-$\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[n]$, by arguing that there is an efficient honest prover. (The rest of the proof then continues without change.)

The key observation is that our current assumption asserts the existence of an *efficient* prover that, on any x and $(\pi_{\bar{s},\bar{p}}, r_i)$, almost maximizes the residual acceptance probability of the protocol when the prefix of the transcript is fixed to $(\pi_{\bar{s},\bar{p}}, r_i)$. For any constant μ, we denote the μ-approximate universal prover by P_{μ}, and on a fixed $\bar{x} = (x, \pi_{\bar{s},\bar{p}}, r_i)$, we denote by $\nu = \nu_{\bar{x}}$ the maximal acceptance probability of the residual protocol, across all provers.

The verifier for (Y, N) is identical to that in the original proof of Claim 7.5.2; that is, the verifier simulates the residual protocol for $\mathrm{poly}(1/\epsilon)$ times in parallel, and accepts if and only if the average probability across simulations, denoted $\tilde{\nu}$, satisfies $\tilde{\nu} \geq \tau + \epsilon/8$. When $(\bar{x}, \tau) \in \mathsf{N}$, for any prover, with high probability we have that $\tilde{\nu} < \tau + \epsilon/8$. On the other hand, when $(\bar{x}, \tau) \in \mathsf{Y}$, a prover that simulates $P_{\epsilon/16}$ on the $\mathrm{poly}(1/\epsilon)$ parallel instantiations of the protocol yields $\tilde{\nu} \geq \tau + \epsilon/8$, with high probability. Thus, $(\mathsf{Y}, \mathsf{N}) \in \mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[n]$ as we wanted. ■

7.2.4 A deterministic argument system for #SAT with runtime $2^{\epsilon \cdot n}$

We now state and prove Theorem 1.4, as a particularly appealing special case of Theorem 7.8. Specifically, we show that under strong hardness assumptions, we can solve

#SAT by a deterministic doubly efficient argument system running in time $2^{\epsilon \cdot n}$, for an arbitrarily small $\epsilon > 0$.

Theorem 7.9 (a deterministic argument system for #SAT with runtime $2^{\epsilon \cdot n}$)**.** *For every $\alpha, \beta \in (0,1)$ there exists $\eta > 0$ such that the following holds. Assume that for every constant $c \in \mathbb{N}$ and logarithmic function $\ell(n) = O(\log(n))$, the $(N \mapsto K, K^{\beta}, \alpha)$-non-batch-computability assumption holds for time $\bar{T} = T \cdot K$ with oracle access to pr-$\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[n]$ on inputs of length $O(T)$, where*

$$\begin{aligned} T(n) &= O(n) , \\ N(n) &= n + c \cdot T(n) , \\ K(n) &= \ell(n)^{1/\eta} . \end{aligned}$$

Then, for every $\epsilon > 0$ there exists a deterministic verifier V that gets as input an n-bit formula Φ of size at most $2^{o(n)}$, runs in time $2^{\epsilon \cdot n}$, and satisfies the following:

1. *There exists an algorithm that gets input Φ, runs in time $2^{O(n)}$, and prints a proof π such that $V(\Phi, \pi) = \#\mathsf{SAT}(\Phi)$.*

2. *For every probabilistic algorithm P running in time $2^{O(n)}$ and every sufficiently large $n \in \mathbb{N}$, the probability that $P(1^n)$ outputs an n-bit formula Φ of size $2^{o(n)}$ and proof π such that $V(\Phi, \pi) \notin \{\perp, \#\mathsf{SAT}(\Phi)\}$ is $2^{-\omega(n)}$.*

Proof. Let $\epsilon' > 0$ be a sufficiently small constant. Using Theorem 3.17 with a sufficiently large constant $k \in \mathbb{N}$ and with $\delta > 0$ such that $\delta/(1-\delta) = 1/k$, we have an $\mathsf{de}\mathcal{IP}_{\mathsf{uni}}^{[\rightleftharpoons 2k+1]}[2^{\epsilon' \cdot n}]$ protocol for counting the number of satisfying assignments of an n-bit formula of size $2^{o(n)}$, which uses at most $O(n)$ random coins.[49]

By a padding argument, we think of the input as of size $\bar{n} = 2^{\epsilon' \cdot n}$ and of the protocol as running in linear time with logarithmically many coins. Also, we consider the $2n = O(\log \bar{n})$ Boolean protocols $U_U, \dots, P_n, \bar{U}_1, \dots, \bar{U}_n$, where each U_i is a protocol for proving that the i^{th} bit of $\#\mathsf{SAT}(\Phi)$ is 1 and each $\bar{U}_i$ is a protocol for proving that the i^{th} bit of $\#\mathsf{SAT}(\Phi)$ is 0.

We apply Theorem 7.8 to each of $2n$ protocols, with parameters $T(\bar{n}) = O(\bar{n})$ and $R(\bar{n}) = O(\log \bar{n})$, to obtain a sequence of $2n$ deterministic verifiers $D_1, \dots, D_n, \bar{D}_1, \dots, \bar{D}_n$ each running in time $\tilde{O}(T)$. Given a formula Φ, our verifier V expects to receive from the prover a value $\rho \in \{0,1\}^n$ and n witnesses $w_1, \dots, w_n \in \{0,1\}^{\tilde{O}(\bar{n})}$ such that for every $i \in [n]$ it holds that $\begin{cases} D_i(w_i) = 1 & \rho_i = 1 \\ \bar{D}_i(w_i) = 1 & \rho_i = 0 \end{cases}$. Whenever this happens V outputs ρ, otherwise it outputs $\perp$.

For every algorithm P running in time $2^{O(n)}$, by a union-bound, the probability over $\Phi \sim \mathbf{\Phi}$ that P outputs a proof that falsely convinces some D_i or $\bar{D}_i$ is negligible in N. Also, the running time of V is $\tilde{O}(n \cdot \bar{n}) < 2^{\epsilon \cdot n}$. ■

7.3 The general case: Constant-round doubly efficient proof systems

In this section we prove Theorem 1.8. First, in Section 7.3.1, we show yet another refinement of the reconstructive PRG from Proposition 5.2, which will be used in the proof. Then in Section 7.3.2 we prove Theorem 1.8.

[49]Note that Theorem 3.17 yields an $\mathcal{MATIME}^{[\rightleftharpoons 2k]}$ protocol that has a universal prover running in time $2^{O(n)}$. In particular, such a protocol can be simulated by a $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons 2k+1]}$ protocol with the same verifier running time.

7.3.1 Yet another refinement of the reconstructive PRG

We now further refine the reconstructive PRG from Proposition 5.2. The goal now will be for the PRG to work not only with distinguishers, but also with distinguishers that solve a promise problem (i.e., evaluate to Y on a pseudorandom input more than they evaluate to "not N" on a random input). The crux of the proof is to show that the advice depends only on the promise-problem, rather than on any particular oracle that solves the problem (and may behave arbitrarily on queries outside the promise).

Proposition 7.10 (an extension of the PRG from Proposition 5.2 to "promise problem" distinguishers)**.** *For any constant $\epsilon' > 0$, we can replace the "furthermore" claim in Proposition 5.2, with the following claim. Fix a promise-problem (Y, N) such that*

$$\Pr[G^f(\mathbf{u}_{(1+\epsilon_0)\cdot\log(N)}) \in \mathsf{Y}] > \Pr[\mathbf{u}_N \notin \mathsf{N}] + \epsilon' \ .$$

Then, there exists $s \in \mathbb{N}$ satisfying $s|\bar{t}$, and $\alpha \in (0,1)$, and an advice string $\mathtt{adv}$ *of length $|f|^{1-\delta_0}$ such that the following holds. Denoting by $a_{x,w,\gamma}$ the sequence of answers to R's queries on input $x \in [|f|]$ and witness w and random choices γ, we have that:*

1. **(Completeness.)** *For any x there exists w such that with probability $1 - 1/N$ over γ, any function that agrees with (Y, N) yields a sequence of answers to the oracle queries that is (s, α)-valid, and if $a_{x,w,\gamma}$ is (s, α)-indicative of a sequence that agrees with (Y, N), then $R(x, w)$ outputs f_x.*

2. **(Soundness.)** *For every (x, w), with probability at least $1 - 1/N$ over γ, if $a_{x,w,\gamma}$ is (s, α)-indicative of a sequence that agrees with (Y, N) on the oracle queries, then $R(x, w)$ outputs either $\perp$ or f_x.*

3. **(Deficient oracles.)** *If $a_{x,w,\gamma}$ is (s, α)-deficient then R outputs $\perp$.*

Proof. We instantiate $\mathsf{Samp}\colon \{0,1\}^{\bar{N}} \times [\bar{L}] \to \{0,1\}^N$ just as in the proof of Proposition 4.1, and recall that its accuracy δ is sub-constant. We instantiate Enc with an agreement parameter $\rho = \rho(\epsilon')$ that is now a sufficiently small constant that depends on $\epsilon' > 0$. Other than that, we use the exact same generator G, and denote again the uniform distribution over the output-set of G^f by $\mathbf{G}$.

In the current proof, instead of assuming that we have oracle access to a function $D\colon \{0,1\}^N \to \{0,1\}$ that is a $(1/10)$-distinguisher for $\mathbf{G}$, we are first fixing a promise-problem (Y, N) such that

$$\Pr[\mathbf{G} \in \mathsf{Y}] > \Pr[\mathbf{u}_N \notin \mathsf{N}] + \epsilon' \ , \tag{7.2}$$

and are only assuming that we have access to *some* (arbitrary) $D\colon \{0,1\}^N \to \{0,1\}$ that solves (Y, N). Our goal is for the advice to depend only on (Y, N), but for the reconstruction to work with *any* oracle D that agrees with (Y, N).

The error-reduced "promise distinguisher". We define the following two sets:

$$S = \left\{ z \in \{0,1\}^{\bar{N}} : \Pr_{j \in [\bar{L}]}[\mathsf{Samp}(z, j) \notin \mathsf{N}] \le \Pr[\mathbf{u}_N \notin \mathsf{N}] + \epsilon'/100 \right\} ,$$

and

$$T = \left\{ z \in \{0,1\}^{\bar{N}} : \Pr_{j \in [\bar{L}]}[\mathsf{Samp}(z, j) \in \mathsf{Y}] > \Pr[\mathbf{u}_N \notin \mathsf{N}] + \epsilon'/10 \right\} .$$

Observe that $T \subseteq \bar{S}$; this is the case because for any z we have that $\Pr_j[\mathsf{Samp}(z,j) \in \mathsf{Y}] \le \Pr_j[\mathsf{Samp}(z,j) \notin \mathsf{N}]$, and thus if $z \in T$ then $z \notin S$. Also note that $|\bar{S}| \le 2^{\bar{N}^{1-\gamma}}$, by the properties of Samp (we crucially use the fact that S is defined with respect to the specific event of being not in N). Finally, similarly to Eq. (4.2), we have that

$$\begin{aligned}\Pr[\mathbf{G} \in \mathsf{Y}] &= \Pr_{i,j}\left[\mathsf{Samp}(\bar{f}_i, j) \in \mathsf{Y}\right] \\ &\le \Pr_i[\bar{f}_i \in T] + \Pr_{i,j}\left[\mathsf{Samp}(\bar{f}_i, j) \in \mathsf{Y} | \bar{f}_i \notin T\right] \\ &\le \Pr_i[\bar{f}_i \in T] + \left(\Pr[\mathbf{u}_N \notin \mathsf{N}] + \epsilon'/10\right) ,\end{aligned}$$

and using Eq. (7.2) we deduce that $\Pr_i[\bar{f}_i \in T] \ge \rho$, relying on the assumption that $\rho = \rho(\epsilon')$ is sufficiently small.

Computing a corrupted codeword. Analogously to the proofs of Propositions 4.1 and 5.2, we construct a machine M that, given any oracle D that agrees with (Y,N), computes a "corrupted" version of $\bar{f}$. We first fix a hash function $h \in \mathcal{H}$ such that there are no collision in $\bar{S}$, as in the previous proofs. The machine M gets as advice h, the value $\Pr[\mathbf{u}_N \notin \mathsf{N}]$, the set $I = \left\{(i, h(i)) :\in [L] \wedge \bar{f}_i \in T\right\}$, and the value $\alpha = \Pr[\mathbf{u}_N \notin \mathsf{N}] + .005$. We stress the following fact:

Observation 7.10.1. *The advice to M depends only on (Y,N), on Samp, and on h.*

Now, given $q \in [|\bar{f}|]$, the machine M first guesses a preimage $z \in \{0,1\}^{\bar{N}}$ for $\bar{f}_i$ and verifies that $h(z) = \bar{f}_i$ using the stored hash value. Then, M uniformly samples $s = O(\log(N))$ values $j_1, \ldots, j_s \in [\bar{L}]$, calls its oracle D on these values, and proceeds if and only if $\nu \stackrel{\text{def}}{=} \Pr_{k\in[t]}[D(\mathsf{Samp}(z, j_k)) = 1] \ge \Pr[\mathbf{u}_N \notin \mathsf{N}] + \epsilon'/50$. If both verifications succeeded, then M outputs the bit in z corresponding to index q, and otherwise M outputs $\perp$. The reconstruction R then uses the list-decoder (with fixed index and randomness) just as in the proof of Proposition 5.2.

The claim about deficient oracles follows immediately by the definition of M above (recall that R outputs $\perp$ whenever M returns $\perp$ in at least one execution). To demonstrate completeness, observe that for any x there exists w such that, with probability at least $1 - 1/N$, on every query q to M, any sequence of answers that is consistent with (Y,N) will have at least an α-fraction of answers that are not in N (since the corresponding z is in T); and any answers with an α-fraction of 1's cause M to correctly compute the corrupted codeword (since the corresponding z is the right preimage for the query under h). When this happens, Dec (and R) output f_x.

For the soundness case, fix (x, w) and again recall that the non-determinism w for R yields non-deterministic strings for each of the execution of M. For every execution of M on query q and with non-determinism z, if $z \in S$ then with probability at least $1 - 1/N^2$ it holds that

$$\Pr_{k\in[t]}[\mathsf{Samp}(z, j_k) \notin \mathsf{N}] < \Pr[\mathbf{u}_N \notin \mathsf{N}] + \epsilon'/50 ,$$

in which case there does not exist a sequence of answers with at least an α-fraction of 1's that agrees with (Y,N). In this case the soundness claim holds vacuously (because there does not exist (s,α)-valid sequence of answers that agrees with (Y,N)).

Therefore, by a union-bound over the queries, we may assume that $z \in \bar{S}$ for all queries to M and that M outputs either $\perp$ (if $z \in \bar{S}$ is not the unique preimage under h for the corresponding query) or the corresponding bit in the corrupted codeword (if

$z \in \bar{S}$ is indeed the unique preimage under h for the corresponding query). In case one of the queries of Dec is answered by $\perp$, then R outputs $\perp$ by definition; and otherwise, the execution of Dec is identical to an execution with access to the corrupted codeword, in with case $R(x, w) = f_x$. ■

7.3.2 The proof of Theorem 1.8

We now show that under strong hardness assumptions we can reduce the number of random coins in an arbitrary doubly efficient proof system to be logarithmic, without increasing the number of rounds. The hardness assumptions will refer to a function whose truth-tables can be recognized in near-linear time, but that is hard for multi-round $\mathcal{AM}$ protocols running in time $2^{(1-\delta)\cdot n}$ with $2^{(1-\delta)\cdot n}$ bits of non-uniform advice, for a small $\delta > 0$.

Theorem 7.11 (drastically reducing the number of random coins in a constant-round proof system)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Let $c \in \mathbb{N}$ be a constant, and assume that there exists $L^{\mathsf{hard}} \notin \mathcal{MATIME}^{[\rightleftharpoons c+1]}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ such that given $n \in \mathbb{N}$, the truth-table of L^{hard} of n-bit inputs can be printed in time $2^{(1+\epsilon/3)\cdot n}$. Then, for every polynomial $T(n)$ it holds that*

$$\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T^{1+\epsilon}, (1+\epsilon)\cdot \log(T)] \,.$$

Proof. Let $L \in \mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T]$, and let V be the corresponding T-time verifier for L. Denote by $c' = \lceil c/2 \rceil$ the number of turns of the verifier in the interaction. As in the proof of Theorem 7.5, we denote by $\langle V, P, x\rangle (r_1, \ldots, r_{c'})$ the result of an interaction between V and P on common input x where the coins $r_1, \ldots, r_{c'}$ are gradually revealed in the c' turns of V.

The new verifier. We define a verifier V' that gets input $x \in \{0,1\}^n$, and acts as follows. For $N = T(n)$, let f_n be the truth-table of L^{hard} on inputs of length $(1+\epsilon/3)\cdot \log(N)$. Consider the generator G from Proposition 7.10 with $\epsilon_0 = \epsilon$ and input 1^N and oracle access to f_n, and denote the number of its outputs by $\bar{N} = N^{1+\epsilon}$.

The verifier V' computes f_n, and in each turn i, it chooses a random $s_i \in [\bar{N}]$ and sends $G^{f_n}(s_i)$ it to the prover, instead of a random N-bit string. In the end V' applies the predicate V to the transcript. Note that V' uses $(1+\epsilon)\cdot \log(N)$ random coins in each turn, and runs in time $O(N^{1+\epsilon})$.

Completeness and soundness. The honest prover for V' behaves identically to the prover for V. Since the new protocol simulates the original protocol with a pseudorandom subset of the random strings, completeness follows immediately. We thus focus on proving the soundness of V'. Fix $x \notin L$.

<u>Notation.</u> Let us introduce some useful notation. For every $i \in [c']$ let V_i be the verifier that in the first i turns uses pseudorandom coins as above, and in the rest of the interaction uses random coins. By definition, we have that

$$\langle V_i, P, x\rangle (s_1, \ldots, s_i, r_{i+1}, \ldots, r_{c'}) = \langle V, P, x\rangle \left(G^{f_n}(s_1), \ldots, G^{f_n}(s_i), r_{i+1}, \ldots, r_{c'} \right) \,.$$

We define a sequence of hybrids, as follows:

$$p_{x,i} = \max_P \left\{ \Pr_{s_1,\ldots,s_i,r_{i+1},\ldots,r_{c'}} [\langle V_i, P, x\rangle (s_1, \ldots, s_i, r_{i+1}, \ldots, r_{c'}) = 1] \right\} \,, \tag{7.3}$$

where $i = 0, \dots, c'$. Indeed $p_{x,0}$ is just the maximal acceptance probability of V on input x whereas $p_{x,c'}$ is the maximal acceptance probability of V' on input x (where in both cases the maximum is taken over all provers).

Now, let $\bar{P}$ be the prover that maximizes $p_{x,c'}$. For any $i \in [c']$ and prover P, we think P as a concatenation $P^{1\dots i} \circ P^{i+1\dots c'}$,[50] and for any $i \in \{0, \dots, c'\}$ we denote

$$(p_{x,i}|\bar{P}) = \max_{P^{i+1\dots c'}} \left\{ \Pr_{s_1,\dots,s_i,r_{i+1},\dots,r_{c'}} \left[\left\langle V_i, \bar{P}^{1\dots i} \circ P^{i+1\dots c'}, x \right\rangle (s_1, \dots, s_i, r_{i+1}, \dots, r_{c'}) = 1 \right] \right\} .$$

A hybrid argument. Assume that $p_{x,c'} = (p_{x,c'}|\bar{P}) \geq 1/2$. Since $x \notin L$, we have that $(p_{x,0}|\bar{P}) \leq 1/3$. It follows that

$$1/6 < (p_{x,c'}|\bar{P}) - (p_{x,0}|\bar{P}) = \sum_{i \in [c']} (p_{x,i}|\bar{P}) - (p_{x,i-1}|\bar{P}) ,$$

and hence for some $i \in [c']$ it holds that $(p_{x,i}|\bar{P}) - (p_{x,i-1}|\bar{P}) > 1/20c'$. Observe that

$$\begin{aligned}
&(p_{x,i}|\bar{P}) - (p_{x,i-1}|\bar{P}) \\
&= \max_{P^{i+1\dots c'}} \left\{ \Pr_{s_1,\dots,s_i,r_{i+1},\dots,r_{c'}} \left[\left\langle V_i, \bar{P}^{1\dots i} \circ P^{i+1\dots c'}, x \right\rangle (s_1, \dots, s_i, r_{i+1}, \dots, r_{c'}) = 1 \right] \right\} \\
&\quad - \max_{P^{i\dots c'}} \left\{ \Pr_{s_1,\dots,s_{i-1},r_i,\dots,r_{c'}} \left[\left\langle V_{i-1}, \bar{P}^{1\dots i-1} \circ P^{i\dots c'}, x \right\rangle (s_1, \dots, s_{i-1}, r_i, \dots, r_{c'}) = 1 \right] \right\} \\
&\geq \max_{P^{i\dots c'}} \left\{ \Pr_{s_1,\dots,s_i,r_{i+1},\dots,r_{c'}} \left[\left\langle V_i, \bar{P}^{1\dots i-1} \circ P^{i\dots c'}, x \right\rangle (s_1, \dots, s_i, r_{i+1}, \dots, r_{c'}) = 1 \right] \right\} \\
&\quad - \max_{P^{i\dots c'}} \left\{ \Pr_{s_1,\dots,s_{i-1},r_i,\dots,r_{c'}} \left[\left\langle V_{i-1}, \bar{P}^{1\dots i-1} \circ P^{i\dots c'}, x \right\rangle (s_1, \dots, s_{i-1}, r_i, \dots, r_{c'}) = 1 \right] \right\} \\
&= \mathop{\mathbb{E}}_{s_1,\dots s_i} \left[\max_{P^{i\dots c'}} \left\{ \Pr_{r_{i+1},\dots,r_{c'}} \left[\left\langle V_i, \bar{P}^{1\dots i-1} \circ P^{i\dots c'}, x \right\rangle (s_1, \dots, s_i, r_{i+1}, \dots, r_{c'}) = 1 \right] \right\} \right] \\
&\quad - \mathop{\mathbb{E}}_{s_1,\dots,s_{i-1},r_i} \left[\max_{P^{i\dots c'}} \left\{ \Pr_{r_{i+1},\dots,r_{c'}} \left[\left\langle V_{i-1}, \bar{P}^{1\dots i-1} \circ P^{i\dots c'}, x \right\rangle (s_1, \dots, s_{i-1}, r_i, \dots, r_{c'}) = 1 \right] \right\} \right] ,
\end{aligned} \tag{7.4}$$

where the last equality is justified using the precise same argument as in Claim 7.5.3.1.

Defining a distinguisher. For any $\bar{s} = (s_1, \dots, s_{i-1})$ and $r_i \in \{0,1\}^N$, denote

$$p_{\bar{s}}(r_i) = \max_{P^{i\dots c'}} \left\{ \Pr_{r_{i+1},\dots,r_{c'}} \left[\left\langle V_{i-1}, \bar{P}^{1\dots i-1} \circ P^{i\dots c'}, x \right\rangle (\bar{s}, r_i, \dots, r_{c'}) = 1 \right] \right\} ,$$

and observe that Eq. (7.4) can thus be rewritten as

$$\mathop{\mathbb{E}}_{\bar{s}} \left[\mathop{\mathbb{E}}_{s_i \in [\bar{N}]} \left[p_{\bar{s}}(G^{f_n}(s_i)) \right] - \mathop{\mathbb{E}}_{r_i \in \{0,1\}^N} \left[p_{\bar{s}}(r_i) \right] \right] > 1/20c' .$$

We fix $\bar{s} = (s_1, \dots, s_{i-1})$ such that the expected difference above is attained. Now, using Lemma 7.5.3.4 with $\mathbf{x} = p_{\bar{s}}(G^{f_n}(\mathbf{u}_{[}\bar{N}]))$ and $\mathbf{y} = p(\mathbf{u}_N)$ and $\epsilon = 1/20c'$, there exists $\tau \in \{0, \epsilon/4, 2\epsilon/4, \dots, 1\}$ such that

$$\Pr\left[p_{\bar{s}}(G^{f_n}(\mathbf{u}_{[\bar{N}]})) \geq \tau + \epsilon/4 \right] - \Pr\left[p(\mathbf{u}_N) \geq \tau \right] > \epsilon/2 . \tag{7.5}$$

[50]Recall that, as in the proof of Theorem 7.5, we think of P as a sequence of c' functions, mapping transcripts to N-bit responses.

Now, consider the following promise problem:

$$\begin{aligned} \mathsf{Y} &= \left\{(x, r_i, \pi_{\bar{s},\bar{p}}, \tau) : p_{\bar{s}}(r_i) \geq \tau + \epsilon/4\right\} \\ \mathsf{N} &= \left\{(x, r_i, \pi_{\bar{s},\bar{p}}, \tau) : p_{\bar{s}}(r_i) \leq \tau\right\} \ . \end{aligned}$$

Note that by Eq. (7.5), we have that

$$\epsilon/2 > \Pr[G^{f_n}(\mathbf{u}_{[\bar{N}]}) \in \mathsf{Y}] - \Pr[\mathbf{u}_N \notin \mathsf{N}] \ .$$

We further argue that (Y, N) is in $pr\mathcal{AMTIME}^{[\rightleftharpoons c]}[O(n)]$, via the following protocol U. (Note that the input length to this problem is N, and thus the running time $O(N)$ that we will prove is linear in its input length.) The protocol U simulates the protocol underlying $p_{\bar{s}}$ (i.e., interacts with its prover using random coins and applies the final predicate that V applies to the interaction, when the prefix is π) for constantly many times in parallel, to estimate its acceptance probability, with high probability, up to an error of $\epsilon/8$. The procedure accepts if and only if its estimate is more than $\tau + \epsilon/8$. Note that U is indeed an $\mathcal{AMTIME}^{[\rightleftharpoons c]}$ protocol that runs in linear time, uses a linear number of advice bits, and solves (Y, N).

A reconstruction argument. Let D be a protocol for the $pr\mathcal{AMTIME}^{[\rightleftharpoons c]}[O(n)]$ above. Consider the reconstruction algorithm R from Proposition 7.10, with the advice string `adv` and $s \in \mathbb{N}$ and $\alpha > 0$ that correspond to (Y, N) above. Our protocol for the hard function f will simulate R, and resolve its queries by simulating D with our prover.

By the completeness of R, for every z there exists w for which with high probability over R's random coins, any sequence of answers to oracle queries that agrees with (Y, N) is (s, α)-valid, and any sequence of answers that is (s, α)-indicative of a sequence that agrees with (Y, N) causes R to output f_z. Thus, the prover can simply convince the verifier of the veracity of all queries that are Y instances: In this case, regardless of what the protocol answers for N instances, the sequence of answers agrees with (Y, N) and is thus (s, α)-indicative of itself, causing R to output f_z.

To establish the soundness of the protocol, denote the sequence of answers to the queries of R by $d_1, \ldots, d_{\bar{t}}$, and observe that with high probability, for every query q_i we have $d_i = 1 \Rightarrow q_i \notin \mathsf{N}$. Then, by Proposition 7.10, with high probability the following holds: If the sequence of answers to the verifier's queries is (α, s)-deficient, then R outputs $\perp$; and otherwise, it means that the answers are (s, α)-indicative of the sequence of answers that agree with (Y, N), in which case we are in the soundness case and the output is either f_z or $\perp$.

The procedure above is an $\mathcal{MATIME}^{[\rightleftharpoons c+1]}$ protocol (since we are using the first round to receive w, then c rounds to simulate D) and it runs in time

$$\underbrace{|f_n|^{1-\delta_0}}_{\text{complexity of } R} + \underbrace{N^{\epsilon/10}}_{\text{number of queries}} \cdot \underbrace{O(N)}_{\text{length of each query}} < |f_n|^{1-\delta} \ ,$$

relying on a sufficiently small choice of $\delta > 0$. Similarly, the total number of advice bits that it uses is $|f_n|^{1-\delta_0} + O(N) < |f_n|^{1-\delta}$. This contradicts the hardness of L^{hard}. ■

By combining Theorems 7.11 and 7.5, we now show that under strong hardness assumptions we can simulate any $\mathcal{AMTIME}^{[\rightleftharpoons c]}$ protocol by a deterministic doubly efficient argument system, with essentially no time overhead.

Corollary 7.12 (simulating proof systems by deterministic doubly efficient argument systems; Theorem 1.8, restated)**.** *For every $\alpha, \beta, \epsilon \in (0,1)$ there exists $\eta, \delta > 0$ such that for every $c \in \mathbb{N}$ the following holds. Assume that:*

1. *There exists $L^{\mathsf{hard}} \notin \mathcal{MATIME}^{[\rightleftharpoons c+1]}[2^{(1-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$ such that given $n \in \mathbb{N}$, the truth-table of L^{hard} of n-bit inputs can be printed in time $2^{(1+\epsilon/3)\cdot n}$.*
2. *The $(N \mapsto K, K^{\beta}, \alpha)$-non-batch-computability assumption holds for time T' with oracle access to $\mathrm{pr}\mathcal{AMTIME}_2^{[\rightleftharpoons c]}[n]$ on inputs of length $\Theta(T^{1+\epsilon/2} \cdot R)$, where $R = (1+\epsilon)\cdot \log(T)$ and $N = n + (c \cdot \lceil c/w \rceil) \cdot T^{1+\epsilon/2} \cdot R$ and $K = R^{1/\eta}$ and $T' = T^{1+\epsilon/2} \cdot K \cdot R$.*

Then, $\mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T] \subseteq \mathsf{de}\mathcal{DARG}[T^{1+\epsilon}]$.

Proof. Let $L \in \mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[T]$. By the first hypothesis and Theorem 7.11 we have that $L \in \mathsf{de}\mathcal{IP}^{[\rightleftharpoons c]}[\bar{T}, R]$ for $\bar{T} = T^{1+\epsilon/2}$ and $R = (1+\epsilon/2)\cdot \log(T)$. And by the second hypothesis and Theorem 7.5 with time bound $\bar{T}$ and randomness R, we have that $L \in \mathsf{de}\mathcal{DARG}[\bar{T} \cdot \mathrm{poly}(R)] \subseteq \mathsf{de}\mathcal{DARG}[T^{1+\epsilon}]$. ■

As in Remark 7.7, the derandomization in Corollary 7.12 extends to the setting in which the original honest prover (in the probabilistic proof system) was efficient only when given a witness in a relation, and in this case the honest prover in the argument system is also efficient only when given a witness in the same relation. (This is because, similarly to Remark 7.7, the honest prover in the argument system underlying Theorem 7.11 is identical to the original honest prover.)

Acknowledgements

The authors are very grateful to Ron Rothblum, for explaining to us some of the possibilities for batch-computing functions with doubly efficient interactive proofs. We also thank Avi Wigderson, Ryan Williams, Oded Goldreich, Vinod Vaikuntanathan and Alex Lombardi for useful discussions. The first author thanks Avi Wigderson for hosting him at the IAS for a visit in which most of this work was carried out.

The first author is supported by NSF CCF-2127597 and an IBM Fellowship. Part of this work was conducted while the second author was at DIMACS and partially supported by the National Science Foundation under grant number CCF-1445755.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: A modern approach*. Cambridge University Press, Cambridge, 2009.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. “PRIMES is in P”. In: *Annals of Mathematics. Second Series* 160.2 (2004), pp. 781–793.

[AKS19] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. “Errata: PRIMES is in P [MR2123939]”. In: *Annals of Mathematics. Second Series* 189.1 (2019), pp. 317–318.

[BGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. “Short PCPs Verifiable in Polylogarithmic Time”. In: *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*. IEEE Computer Society, 2005, pp. 120–134.

[BM88] László Babai and Shlomo Moran. "Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes". In: *Journal of Computer and System Sciences* 36.2 (1988), pp. 254–276.

[BSGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. "Robust PCPs of proximity, shorter PCPs and applications to coding". In: *SIAM Journal of Computing* 36.4 (2006), pp. 889–974.

[CT21a] Lijie Chen and Roei Tell. "Hardness vs Randomness, Revised: Uniform, Non-Black-Box, and Instance-Wise". In: *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2021.

[CT21b] Lijie Chen and Roei Tell. "Simple and fast derandomization from very hard functions: Eliminating randomness at almost no cost". In: *Proc. 53st Annual ACM Symposium on Theory of Computing (STOC)*. 2021.

[DMO+20] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. "Nearly Optimal Pseudorandomness From Hardness". In: *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2020.

[FF93] Joan Feigenbaum and Lance Fortnow. "Random-self-reducibility of complete sets". In: *SIAM Journal of Computing* 22.5 (1993), pp. 994–1005.

[FGM+89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. "On Completeness and Soundness in Interactive Proof Systems". In: *Advances in Computing Research* 5 (1989).

[FLM+05] Lance Fortnow, Richard Lipton, Dieter van Melkebeek, and Anastasios Viglas. "Time-space lower bounds for satisfiability". In: *Journal of the ACM* 52.6 (2005), pp. 835–865.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating computation: interactive proofs for muggles". In: *Journal of the ACM* 62.4 (2015), Art. 27, 64.

[GLR+91] Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. "Self-Testing/Correcting for Polynomials and for Approximate Functions". In: *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*. Ed. by Cris Koutsougeras and Jeffrey Scott Vitter. ACM, 1991, pp. 32–42.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. New York, NY, USA: Cambridge University Press, 2008.

[Gol18] Oded Goldreich. "On doubly-efficient interactive proof systems". In: *Foundations and Trends® in Theoretical Computer Science* 13.3 (2018), front matter, 1–89.

[GR18] Oded Goldreich and Guy N. Rothblum. "Simple doubly-efficient interactive proof systems for locally-characterizable sets". In: *Proc. 9th Conference on Innovations in Theoretical Computer Science (ITCS)*. Vol. 94. 2018, Art. No. 18, 19.

[GSTS03] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. "Uniform hardness versus randomness tradeoffs for Arthur-Merlin games". In: *Computational Complexity* 12.3-4 (2003), pp. 85–130.

[GW14] Oded Goldreich and Avi Widgerson. “On derandomizing algorithms that err extremely rarely”. In: *Proc. 46th Annual ACM Symposium on Theory of Computing (STOC)*. Full version available online at *Electronic Colloquium on Computational Complexity: ECCC*, 20:152 (Rev. 2), 2013. 2014, pp. 109–118.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. “In search of an easy witness: exponential time vs. probabilistic polynomial time”. In: *Journal of Computer and System Sciences* 65.4 (2002), pp. 672–694.

[KM98] Adam Klivans and Dieter van Melkebeek. “Graph Nonisomorphism has Subexponential Size Proofs Unless the Polynomial-Time Hierarchy Collapses”. In: *Electronic Colloquium on Computational Complexity: ECCC* 5 (1998), p. 75.

[LFK+92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. “Algebraic methods for interactive proof systems”. In: *Journal of the Association for Computing Machinery* 39.4 (1992), pp. 859–868.

[MNT93] Yishay Mansour, Noam Nisan, and Prasoon Tiwari. “The computational complexity of universal hashing”. In: vol. 107. 1. 1993, pp. 121–133.

[MV05] Peter Bro Miltersen and N. V. Vinodchandran. “Derandomizing Arthur-Merlin games using hitting sets”. In: *Computational Complexity* 14.3 (2005), pp. 256–279.

[NW94] Noam Nisan and Avi Wigderson. “Hardness vs. randomness”. In: *Journal of Computer and System Sciences* 49.2 (1994), pp. 149–167.

[RR97] Alexander A. Razborov and Steven Rudich. “Natural proofs”. In: *Journal of Computer and System Sciences* 55.1, part 1 (1997), pp. 24–35.

[RRR18] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. “Efficient Batch Verification for UP”. In: *Proc. 33rd Annual IEEE Conference on Computational Complexity (CCC)*. 2018, 22:1–22:23.

[RRR21] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. “Constant-round interactive proofs for delegating computation”. In: *SIAM Journal of Computing* 50.3 (2021), STOC16–255–STOC16–340.

[RRV02] Ran Raz, Omer Reingold, and Salil Vadhan. “Extracting all the randomness and reducing the error in Trevisan’s extractors”. In: *Journal of Computer and System Sciences* 65.1 (2002), pp. 97–128.

[Sha92] Adi Shamir. “IP = PSPACE”. In: *Journal of the ACM* 39.4 (1992), pp. 869–877.

[Sip88] Michael Sipser. “Expanders, randomness, or time versus space”. In: *Journal of Computer and System Sciences* 36.3 (1988), pp. 379–383.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. “Pseudorandom generators without the XOR lemma”. In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 236–266.

[SU05] Ronen Shaltiel and Christopher Umans. “Simple extractors for all min-entropies and a new pseudorandom generator”. In: *Journal of the ACM* 52.2 (2005), pp. 172–216.

[SU06] Ronen Shaltiel and Christopher Umans. “Pseudorandomness for Approximate Counting and Sampling”. In: *Comput. Complex.* 15.4 (2006), pp. 298–341.

[SU07] Ronen Shaltiel and Christopher Umans. “Low-end uniform hardness vs. randomness tradeoffs for AM”. In: *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC)*. 2007, pp. 430–439.

[Tel21] Roei Tell. “How to Find Water in the Ocean: A Survey on Quantified Derandomization”. In: *Electronic Colloquium on Computational Complexity: ECCC* 28 (2021), p. 120.

[Tha22] Justin Thaler. *Proofs, Arguments, and Zero-Knowledge*. Accessed at `https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html`, March 28, 2022. 2022.

[Tou01] Iannis Tourlakis. “Time-space tradeoffs for SAT on nonuniform machines”. In: *Journal of Computer and System Sciences* 63.2 (2001), pp. 268–287.

[TSZS06] Amnon Ta-Shma, David Zuckerman, and Shmuel Safra. “Extractors from Reed-Muller codes”. In: *Journal of Computer and System Sciences* 72.5 (2006), pp. 786–812.

[WB86] Lloyd R Welch and Elwyn R Berlekamp. *Error correction for algebraic block codes*. US Patent 4,633,470. 1986.

[Wil13] Ryan Williams. “Improving Exhaustive Search Implies Superpolynomial Lower Bounds”. In: *SIAM Journal of Computing* 42.3 (2013), pp. 1218–1244.

[Wil16] Richard Ryan Williams. “Strong ETH breaks with Merlin and Arthur: short non-interactive proofs of batch evaluation”. In: *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*. Vol. 50. 2016, Art. No. 2, 17.

[Žák83] Stanislav Žák. “A Turing machine time hierarchy”. In: *Theoretical Computer Science* 26.3 (1983), pp. 327–333.

A Useful properties are necessary for derandomization of $\mathcal{MA}$

In this appendix we prove that useful properties that are *constructive in near-linear time* are necessary for derandomization of $\mathcal{MA}$. We first prove that *infinitely often* useful properties are necessary for *any* derandomization of $\mathcal{MA}$, following the proof approach of Williams [Wil13; Wil16]; and then we prove that useful properties (in the standard sense, i.e. not only infinitely often) are necessary for black-box superfast derandomization of $\mathcal{MA}$.

Definition A.1 (io-useful property; compare with Definition 3.3). *We say that a collection $\mathcal{L} \subseteq \{0,1\}^*$ of strings is a $\mathcal{C}'$-constructive property io-useful against $\mathcal{C}$ if for every $N = 2^n$ it holds that $\mathcal{L}_n = \mathcal{L} \cap \{0,1\}^N \neq \emptyset$, and $\mathcal{L} \in \mathcal{C}'$, and for every $L \in \mathcal{C}$ there are infinitely many $n \in \mathbb{N}$ such that $L_n \notin \mathcal{L}_n$, where $L_n \in \{0,1\}^{2^n}$ is the truth-table of L on n-bit inputs.*

Proposition A.2 (io-useful properties are necessary for derandomization of $\mathcal{MA}$). *If every unary language in $\mathcal{MA}$ is also in $\mathcal{NP}$ then there is a quasilinear-time-constructive property io-useful against circuits of size $2^{\epsilon \cdot n}$, for some constant $\epsilon > 0$.*

Proof. We first use the well-known proof approach from [Wil13; Wil16] to argue that witnesses for a certain unary language in $\mathcal{NTIME}[2^n]$ have exponential circuit complexity, infinitely often; and then follow [Wil16] in observing that the latter statement yields an io-useful property for exponential-sized circuits. Details follow.

Let L be a unary language in $\mathcal{NTIME}[2^n] \setminus \mathcal{NTIME}[2^{n/2}]$ (see [Žák83]), and let V be a PCP verifier for L with running time $\text{poly}(n)$ and randomness $n + O(\log n)$ and perfect completeness and soundness error $1/3$ (see [BSGH+06]). Consider an $\mathcal{MA}$ verifier V^{PCP} that on n-bit inputs guesses a circuit C of size $2^{\epsilon \cdot n}$, where $\epsilon > 0$ is sufficiently small, and runs V while resolving its oracle queries using C.

Assume towards a contradiction that for every sufficiently large $n \in \mathbb{N}$, if $1^n \in L$ then there is $w \in \{0,1\}^{2^n}$ that is the truth-table of a function computable by circuits of size $2^{\epsilon \cdot n}$ such that $\Pr[V(1^n, w) = 1] = 1$. In this case, V^{PCP} is an $\mathcal{MA}$ verifier that decides L in time $\tilde{O}(2^{\epsilon \cdot n})$, and thus (by our hypothesis and relying on $\epsilon > 0$ being small) it holds that $L \in \mathcal{NTIME}[2^{n/2}]$, a contradiction.

Thus, there is an infinite set $S \subseteq \mathbb{N}$ such that for every $n \in S$ there exists $w \in \{0,1\}^{2^n}$ such that $\Pr[V(1^n, w) = 1] = 1$, and every w satisfying this condition is the truth-table of a function whose circuit complexity is more than $2^{\epsilon \cdot n}$. Our io-useful property consists of all such w's, for every $n \in S$. The constructive algorithm for the property gets $f_n \in \{0,1\}^{2^n}$ and enumerates over the randomness of V to compute $\Pr[V(1^n, f_n) = 1]$; if the latter value is 1 then it accepts, otherwise it rejects. Indeed, the property is non-trivial infinitely often, its truth-tables are hard for circuits of size $2^{\epsilon \cdot n}$, and it is constructive in time $\tilde{O}(2^n)$. ■

We now prove that useful properties (which are not only "infinitely often" useful) that are constructive in near-linear time are necessary for any superfast derandomization of $\mathcal{MA}$ that uses non-deterministic PRGs (NPRGs). The proof follows the standard transformation of PRGs into hard functions, adapting it to our setting: Replacing PRGs with NPRGs, and hard functions with useful properties.

Definition A.3 (non-deterministic PRG)**.** *A non-deterministic machine M is a* non-deterministic ϵ-PRG *(ϵ-NPRG, in short) for a circuit class $\mathcal{C}$ if for every $n \in \mathbb{N}$, when M is given input 1^n it satisfies the following:*

1. *There exists a non-deterministic guess such that M prints $S \subseteq \{0,1\}^n$ for which there is no circuit on n inputs in $\mathcal{C}$ that is an ϵ-distinguisher.*[51]
2. *For every non-deterministic guess, either M prints $S \subseteq \{0,1\}^n$ for which there is no circuit on n inputs in $\mathcal{C}$ that is an ϵ-distinguisher, or M outputs $\perp$.*

Proposition A.4 (useful properties are necessary for derandomization with NPRGs)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. If there is a $(1/10)$-NPRG for linear-sized circuits that is computable in time $n^{1+\epsilon}$, then there is an $\mathcal{NTIME}[N^{1+3\epsilon}]$-computable property $\mathcal{L}$ useful against circuits of size $2^{(1-\delta)\cdot n}$.*

Proof. For every $n \in \mathbb{N}$, let $\mathcal{S}_n$ be the collection of all possible sets S that the NPRG can print on input length 1^n (i.e., when considering all non-deterministic guesses that do not cause M to output $\perp$). Fix $S \in \mathcal{S}_n$, and note that $|S| \le n^{1+\epsilon}$ (due to the running time of M). For $\ell(n) = \lceil (1+2\epsilon) \cdot \log(n) \rceil$, let $f_S \colon \{0,1\}^\ell \to \{0,1\}$ such that $f_S(x) = 1$ if and only if x is a prefix of some $z \in S$. Note that there is no circuit C of size

[51]That is, for every $C \in \mathcal{C}$ on n input bits it holds that $\Pr_{s \in S}[C(s) = 1] \in \Pr_{x \in \{0,1\}^n}[C(x) = 1] \pm \epsilon$.

$O(n) = 2^{\Omega(\ell/(1+2\epsilon))} = 2^{(1-\delta)\cdot\ell}$ that computes f_S (otherwise we could use C to construct a circuit that avoids S but has high acceptance probability).

For every $\ell \in \mathbb{N}$, we include in $\mathcal{L}$ all the functions f_S obtained from $S \in \mathcal{S}_n$ such that $\ell = \ell(n)$. The argument above shows that $\mathcal{L}$ is useful against circuits of size $2^{(1-\delta)\cdot\ell}$, and its non-triviality follows since the NPRG works on all input lengths. Finally, the truth-tables of f_S's included in $\mathcal{L}$ can be recognized in non-deterministic time $\tilde{O}(N^{1+2\epsilon}) < N^{1+3\epsilon}$, by computing the NPRG. ■

Indeed, the proof above works also with the weaker notion of non-deterministic HSG (NHSG), and we formulated it using NPRGs merely since the latter notion is more well-known.

B Proof of Lemma 7.5.3.4

We now restate Lemma 7.5.3.4 from the proof of Theorem 7.5 and prove it.

Lemma B.1. *Let* $\mathbf{x}, \mathbf{y}$ *be two random variables taking values from* $[0,1]$ *and* $\epsilon \in (0,1)$ *such that* $\epsilon^{-1} \in \mathbb{N}$, *and let* $\tau = 4/\epsilon$. *If* $\mathbb{E}[\mathbf{x}] - \mathbb{E}[\mathbf{y}] > \epsilon$, *then there exists* $\ell \in [\tau]$ *such that*

$$\Pr[\mathbf{x} \geq (\ell+1)/\tau] - \Pr[\mathbf{y} \geq \ell/\tau] > \epsilon/2\,.$$

Proof. For every $i \in \{0, 1 \ldots, \tau+1\}$, let

$$p_i = \Pr\Big[\mathbf{x} \in [i/\tau, (i+1)/\tau)\Big] \quad \text{and} \quad q_i = \Pr\Big[\mathbf{y} \in [i/\tau, (i+1)/\tau)\Big]\,.$$

Note that $p_{\tau+1} = q_{\tau+1} = 0$ (we define them purely for notational convenience), and that $\sum_{i=0}^{\tau} p_i = \sum_{i=0}^{\tau} q_i = 1$. Also, we have that

$$\mathbb{E}[\mathbf{x}] \in \left[\sum_{i=0}^{\tau} p_i \cdot (i/\tau), \tau^{-1} + \sum_{i=0}^{\tau} p_i \cdot (i/\tau)\right)\,,$$
$$\mathbb{E}[\mathbf{y}] \in \left[\sum_{i=0}^{\tau} q_i \cdot (i/\tau), \tau^{-1} + \sum_{i=0}^{\tau} q_i \cdot (i/\tau)\right)\,,$$

which by our hypothesis $\mathbb{E}[\mathbf{x}] - \mathbb{E}[\mathbf{y}] > \epsilon$ implies that

$$\left[\sum_{i=0}^{\tau} p_i \cdot (i/\tau)\right] - \left[\sum_{i=0}^{\tau} q_i \cdot (i/\tau)\right] > \epsilon - \tau^{-1}\,.$$

Now, denote $p_{\geq\ell} = \sum_{j=\ell}^{\tau} p_j$ and $q_{\geq\ell} = \sum_{j=\ell}^{\tau} q_j$. Then, we have that

$$\sum_{i=0}^{\tau} p_i \cdot (i/\tau) = \sum_{\ell=1}^{\tau} p_{\geq\ell}/\tau = \mathop{\mathbb{E}}_{\ell\in[\tau]}[p_{\geq\ell}]\,,$$
$$\sum_{i=0}^{\tau} q_i \cdot (i/\tau) = \mathop{\mathbb{E}}_{\ell\in[\tau]}[q_{\geq\ell}]\,,$$

and therefore

$$\mathop{\mathbb{E}}_{\ell\in[\tau]}\Big[p_{\geq\ell} - q_{\geq\ell}\Big] > \epsilon - \tau^{-1}\,.$$

Next, observe that

$$\mathop{\mathbb{E}}_{\ell\in[\tau]}[p_{\geq \ell+1}] = \mathop{\mathbb{E}}_{\ell\in[\tau]}[p_{\geq \ell}] - \mathop{\mathbb{E}}_{\ell\in[\tau]}[p_{\ell}] \geq \mathop{\mathbb{E}}_{\ell\in[\tau]}[p_{\geq \ell}] - 1/\tau \ .$$

Putting them together, we have

$$\mathop{\mathbb{E}}_{\ell\in[\tau]}\left[p_{\geq \ell+1} - q_{\geq \ell}\right] > \epsilon - 2\tau^{-1} = \epsilon/2 \ ,$$

and hence there exists $\ell \in [\tau]$ such that $p_{\geq \ell+1} - q_{\geq \ell} > \epsilon/2$. ■