Electronic Colloquium on Computational Complexity, Revision 3 of Report No. 139 (2009)

On the Power of Randomized Reductions and the Checkability of SAT

Mohammad Mahmoody[*] and David Xiao[†]

November 5, 2010

Abstract

We prove new results regarding the complexity of various complexity classes under randomized oracle reductions. We first prove that $\mathbf{BPP}^{\mathbf{PrSZK}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$, where **PrSZK** is the class of promise problems having statistical zero knowledge proofs. This strengthens the previously known facts that **PrSZK** is closed under $\mathbf{NC}^1$ truth-table reductions (Sahai and Vadhan, J. ACM '03) and that $\mathbf{P}^{\mathbf{PrSZK}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$ (Vadhan, personal communication). Our results imply that for most cryptographically interesting lattice problems, there is a sharp threshold for the approximation factor below which we do not know if the problems are even in **AM**, while above which the problems are in $\mathbf{AM} \cap \mathbf{coAM}$ not only via Karp reductions but also via randomized oracle reductions.

Then we investigate the power of randomized oracle reductions with relation to the notion of instance checking (Blum and Kannan, J. ACM '95). We observe that a theorem of Beigel implies that if any problem in **TFNP** such as Nash equilibrium is **NP**-hard under randomized oracle reductions, then SAT is checkable.

We also observe that Beigel's theorem can be extended to an average-case setting by relating checking to the notion of program *testing* (Blum et al., JCSS '93). From this, we derive that if one-way functions can be based on **NP**-hardness via a randomized oracle reduction, then SAT is checkable. By showing that **NP** has a *non-uniform* tester, we also show that worst-case to average-case randomized oracle reduction for any relation (or language) $R \in \mathbf{NP}$ implies that R has a non-uniform instance checker. These results hold even for adaptive randomized oracle reductions.

[*]Princeton University, `mohammad@cs.princeton.edu`. Supported by NSF grants CNS-0627526, CCF-0426582 and CCF-0832797.

[†]LRI, Université Paris-Sud, `dxiao@lri.fr`.

ISSN 1433-8092

1 Introduction

In this paper we study various complexity classes under randomized oracle (i.e., Cook) reductions. Such a reduction is an efficient randomized algorithm with access to an oracle, and the algorithm is allowed to ask the oracle multiple questions, even adaptively (using the oracle's answers to generate new questions). The *closure* of a complexity class $\mathcal{C}$ under randomized oracle reductions, denoted $\mathbf{BPP}^{\mathcal{C}}$, is the class of languages (or, more generally, promise problems) that are decidable using efficient randomized algorithms with access to an oracle solving a problem in $\mathcal{C}$.

The closure of many complexity classes under randomized oracle reductions is poorly understood. For example, the following questions remain completely open (i.e., resolving them in either direction would be consistent with standard conjectures about complexity classes, such as $\mathbf{P} \neq \mathbf{NP}$):

1. Let **PrSZK** denote the class of promise problems having statistical zero knowledge proofs. How big is the class $\mathbf{BPP}^{\mathbf{PrSZK}}$? For example, does it contain **NP**-complete languages like SAT?

2. Is it possible to use an algorithm computing Nash equilibrium to solve SAT?

3. Is it possible to base one-way functions on **NP**-hardness?

4. Is it possible to prove that $\mathbf{P} \neq \mathbf{NP}$ implies the existence of a hard-on-average problem in **NP**?

In this paper, we make the following progress on these questions:

1. $\mathbf{BPP}^{\mathbf{PrSZK}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$. Therefore, SAT $\notin \mathbf{BPP}^{\mathbf{PrSZK}}$ unless $\mathbf{PH} = \mathbf{\Sigma}_2$ [11].

2. If SAT $\in \mathbf{BPP}^{\mathbf{TFNP}}$, then SAT has an instance checker. (Recall that computing Nash equilibrium is in **TFNP** [38]).

3. If there is a black-box reduction basing one-way functions on **NP**-hardness, then SAT has an instance checker.

4. If there is a black-box worst-case to average-case reduction for any relation (or language) $R \in \mathbf{NP}$, then R has a non-uniform instance checker.

Our result about **PrSZK** improves on the previously known facts that **PrSZK** is closed under $\mathbf{NC^1}$ truth-table reductions [43] and that $\mathbf{P}^{\mathbf{PrSZK}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$ [45] (where $\mathbf{P}^{\mathbf{PrSZK}}$ denotes the closure of **PrSZK** under *deterministic* oracle reductions, see Theorem A.1).

Regarding our other results, we use the notion of instance checking (also known as program checking or checkability) as defined by Blum and Kannan [7]. Intuitively a problem Π is checkable if there is an efficient randomized algorithm C that uses any program P that purportedly solves Π, such that for every instance x, C^P either decides x correctly or outputs "I don't know" (with very high probability).

One of the open questions posed by Blum and Kannan [7] was whether SAT has an instance checker. This question has remained open for twenty years, and our results imply that, in order to build a reduction that solves SAT using a **TFNP** oracle or using an inverting oracle for a one-way function, one would along the way come up with an instance checker for SAT. Although there is no widely held belief about whether or not SAT is checkable, nevertheless given that this problem has remained unresolved for over twenty years our theorems can be viewed as evidence that solving SAT using a **TFNP** oracle or using an inverting oracle will be "hard to prove" via standard techniques. We credit Holenstein [29] with the observation that connecting the hardness of various problems to the checkability of SAT is a meaningful statement.

1.1 Difficulties dealing with randomized oracle reductions

We first illustrate the difficulties of understanding randomized oracle reductions. Let $\mathbf{AM} \cap \mathbf{coAM}$ denote the class of *promise* problems Π such that both Π and $\overline{\Pi}$ have **AM** proof systems. Suppose we try to prove that $\mathbf{BPP}^{\mathbf{AM} \cap \mathbf{coAM}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$. (Actually this claim is false unless $\mathbf{NP} \subseteq \mathbf{coAM}$, but we discuss it anyway as it illustrates well the issues involved.)[1] Since $\mathbf{BPP}^{\mathbf{AM} \cap \mathbf{coAM}}$ is closed under complement, it suffices to prove the statement that for every $\Pi \in \mathbf{BPP}^{\mathbf{AM} \cap \mathbf{coAM}}$, Π is also contained in **AM**.

The natural strategy: For every $\Pi \in \mathbf{BPP}^{\mathbf{AM} \cap \mathbf{coAM}}$, Π is decidable by an efficient oracle algorithm A with access to an oracle solving some $\Pi' \in \mathbf{AM} \cap \mathbf{coAM}$. To prove $\Pi \in \mathbf{AM}$ the natural idea is to construct an **AM** protocol for Π where on input z, the prover and verifier do the following:

1. The verifier samples random coins ω for A and sends these to the prover.
2. The prover uses ω to emulate the execution of A. The prover answers any oracle queries "$x \in \Pi'$?" that the reduction asks. Since Π' is a promise problem, the prover is allowed to respond "x does not satisfy the promise". The prover then sends back these oracle queries/answers back to the verifier.
3. The verifier checks that, using ω and the oracle answers claimed by the prover, the execution of A is correct.
4. Since $\Pi' \in \mathbf{AM} \cap \mathbf{coAM}$, this means both $\Pi' = (\Pi'_Y, \Pi'_N)$ and its complement $\overline{\Pi'} = (\Pi'_N, \Pi'_Y)$ have **AM** proofs. Therefore, for every query x satisfying the promise, the verifier asks the prover for proofs that $x \in \Pi'_Y$ or $x \in \Pi'_N$, depending on what the prover claimed previously.
5. If all the checks pass, the verifier outputs whatever A outputs, otherwise the verifier rejects.

One could then hopefully use the correctness of the reduction A to prove completeness and soundness of this protocol. However to prove the soundness of such a protocol one must ensure that a cheating prover cannot trick the verifier into accepting NO instances. This is problematic for the following two reasons:

1. **Prover can falsely claim that a query does not satisfy the promise.** Since the prover can respond "x does not satisfy the promise", and since the verifier cannot check whether or not x satisfies the promise, the cheating prover may possibly respond "x does not satisfy the promise" even on queries x that do satisfy the promise.

[1] The claim is false because Even et al. [14] observed that there is even a deterministic oracle reduction A that can decide SAT using only an oracle which decides a promise problem in $\mathbf{NP} \cap \mathbf{co\text{-}NP} \subseteq \mathbf{AM} \cap \mathbf{coAM}$. This implies that $\mathbf{NP} \subseteq \mathbf{BPP}^{\mathbf{AM} \cap \mathbf{coAM}}$. This claim is false precisely because of the issues discussed above.

2. **Prover can generate responses depending on A's random coins.** There may be queries $x \notin \Pi'_Y \cup \Pi'_N$ such that the prover is able to falsely claim both $x \in \Pi'_Y$ and $x \in \Pi'_N$ without being caught. Namely, when the verifier runs the **AM** proof to verify that $x \in \Pi'_Y$ or $y \in \Pi'_N$, the prover is able to make the verifier accept in both cases. In this case, the prover may choose to claim that x is a YES or NO instance depending on the random coins of A. But this means that we cannot use the correctness of the reduction A to argue that the verifier obtains the correct answer, because the correctness of A only holds with respect to oracles whose responses are *independent* of A's random coins.

Our results overcome these difficulties in two ways. For **PrSZK** we use additional structure of **PrSZK** to force the prover first never to answer "x does not satisfy the promise", and second to answer in a way that is independent of A's random coins. Our results for checkability aim for the more modest goal of constructing an instance checker for Π, which turns out to be easier than constructing an **AM** protocol for Π.

1.2 Complexity of real-valued functions verifiable in AM

Our first main theorem (which will imply our result about **PrSZK** as a corollary) studies the power of randomized oracle reductions that use oracle access to a class of real-valued functions that we call $\mathbb{R}$-**TUAM** (denoting "real-valued total unique **AM**", see Definition 2.5). To understand $\mathbb{R}$-**TUAM**, we begin by discussing the well-known class **TFNP** [32]. The class **TFNP** is the following class of search problems: given input x, find y such that (x, y) satisfy a relation R, with the condition that R is efficiently decidable and R is *total*, namely for every x, there exists y such that $(x, y) \in R$.

TFAM is a natural relaxation of **TFNP**. We still require that R is total, but now we allow R also to be just verifiable in **AM** (and not necessarily decidable in **P**). For functions where in addition $y \in \mathbb{R}$ and y is unique up to some small error, we say that the function is in $\mathbb{R}$-**TUAM** (see Definition 3.1).

Although to the best of our knowledge this class of functions is not well-studied, many natural problems can be decided given a $\mathbb{R}$-**TUAM** oracle. For example, consider the problem of Entropy Difference ED, which is complete for the class **PrSZK** [23]. An instance of this problem is a pair of circuits (X_1, X_2) that we think of as samplers: the distribution sampled by the circuit $X : \{0,1\}^m \mapsto \{0,1\}^n$ is given by the output distribution $X_1(U_m)$ on uniform input bits. We write $H(X)$ to denote the Shannon entropy of the distribution sampled by X. A YES instance of ED satisfies $H(X_1) \geq H(X_2) + 1$, while a NO instance satisfies $H(X_2) \geq H(X_1) + 1$. It is clear that being able to approximate the function $f(X_1, X_2) = H(X_1) - H(X_2)$ is sufficient to decide ED. It turns out that the entropy of distributions sampled by circuits can be verified using an **AM** protocol [17, 1] (see Lemma 3.4) and therefore this function f is in the class $\mathbb{R}$-**TUAM**.

Our main theorem about $\mathbb{R}$-**TUAM** is the following:

Theorem 1.1. $\mathbf{BPP}^{\mathbb{R}\text{-}\mathbf{TUAM}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$.

Application to PrSZK. Theorem 1.1 and the above discussion showing that the **PrSZK**-complete problem ED can be decided using an $\mathbb{R}$-**TUAM** oracle imply the following.

Corollary 1.2. $\mathbf{BPP}^{\mathbf{PrSZK}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$.

Problem	Problem is **NP**-hard	Problem in **PrSZK** via Karp reduction
GapSVP_γ	γ any constant [31]	$\gamma \geq \sqrt{n/\log n}$ [21]
GapCVP_γ	$\gamma = n^{1/\log\log n}$ [13]	$\gamma \geq \sqrt{n/\log n}$ [21]
$\mathsf{GapSIVP}_\gamma$	γ any constant [6]	$\gamma \geq \omega(\sqrt{n \log n})$ [41]

Table 1: Hardness of various lattice problems

Proof technique: Theorem 1.1 is proven using the strategy of Section 1.1. A $\mathbb{R}$-**TUAM** function is total so the prover can never respond that a query does not satisfy the promise. To prevent the prover from answering in a way that depends on the random coins chosen by the verifier to emulate the $\mathbf{BPP}^{\mathbb{R}\text{-}\mathbf{TUAM}}$ algorithm, the verifier adds some additional noise to the prover's responses to make them "independent" of the random coins.

Application to basing cryptography on lattice problems: much exciting recent work in cryptography is based on the hardness of various lattice problems [19, 42, 40]. The problems most often studied are GapSVP_γ, GapCVP_γ, and $\mathsf{GapSIVP}_\gamma$, where $\gamma = \gamma(n)$ is an approximation factor. We refer the reader to [33], for example, for precise definitions of these problems.

Here, we simply note that the hardness of these problems depends critically on the value of γ. This is illustrated in Table 1.

Understanding the hardness of these problems is partly motivated by the goal of basing cryptography on **NP**-hardness. Namely, one might hope to show that there is some function γ such that say GapSVP_γ is **NP**-hard, and furthermore that we can construct a one-way function based on the hardness of GapSVP_γ. This would be a great breakthrough in cryptography, since all known cryptosystems are based on much stronger assumptions than **NP**-hardness.

[21] proved negative evidence suggesting that this goal is unattainable. They observed that all known constructions of one-way functions from GapSVP_γ require $\gamma(n) > \sqrt{n/\log n}$. [21] then showed that it is unlikely to prove that GapSVP_γ is **NP**-hard under Karp reductions because since $\mathsf{GapSVP}_\gamma \in \mathbf{PrSZK}$, this would imply $\mathbf{NP} \subseteq \mathbf{PrSZK} \subseteq \mathbf{coAM}$ (which is conjectured to be false since it would imply that $\mathbf{PH} = \Sigma_2$ [11]). However, [21] does not preclude the possibility of proving that GapSVP_γ is **NP**-hard under randomized reductions. [21, 41] provide similar evidence for GapCVP_γ and $\mathsf{GapSIVP}_\gamma$.

Our result strengthens the negative evidence to also rule out the use of randomized reductions: notice that saying that GapSVP_γ is **NP**-hard under randomized oracle reductions is equivalent to saying that $\mathbf{NP} \in \mathbf{BPP}^{\mathsf{GapSVP}_\gamma}$. Since $\mathsf{GapSVP}_\gamma \in \mathbf{PrSZK}$ for $\gamma \geq \sqrt{n/\log n}$, our Corollary 1.2 implies that proving GapSVP_γ is **NP**-hard under randomized oracle reductions for any $\gamma \geq \sqrt{n/\log n}$ would imply that $\mathbf{NP} \subseteq \mathbf{coAM}$, and is therefore unlikely. On the other hand, for $\gamma(n) < \sqrt{n/\log n}$, it is unknown whether $\mathsf{GapSVP}_\gamma \in \mathbf{coAM}$ even if we are considering only Karp reductions.

1.3 Randomized oracle reductions, checkability, and testability

Although we are able to overcome the difficulties discussed in Section 1.1 to prove Theorem 1.1 for the case of $\mathbb{R}$-**TUAM**, we do not know how to bypass both difficulties in general. In the following, we will instead *assume* that the cheating prover is a static, fixed (but possibly faulty) oracle. This by definition prevents the prover from responding in a way that depends on the random coins the verifier sends, and in the settings we study the prover does not benefit by answering "x does not satisfy the promise". The interesting point is that even achieving soundness only against such a severely restricted static prover will imply that the language is "checkable".

Checkability was defined by Blum and Kannan [7] and intuitively guarantees the following: C is an instance checker for a problem Π if, when C is given an instance x and a program P, if P decides Π correctly (for all inputs, not just x), then $C(P, x)$ outputs the correct answer for x with high probability, while if P decides x incorrectly then with high probability $C(P, x)$ either outputs "error" or finds the correct answer despite the incorrectness of $P(x)$. Most instance checkers C require only oracle access to P and not the code of P.

It was already observed by Blum and Kannan [7] that Π is checkable (with C using only oracle access to P) if and only if Π and $\overline{\Pi}$ have interactive proofs where the prover answers only queries of the form "$x \in \Pi$?" and soundness is only required to hold against static cheating provers of the kind described above.

1.3.1 An application of Beigel's theorem

Theorem 1.3 (Beigel, as cited in [7])**.** *Suppose* Π *is decidable by a randomized reduction* A *with oracle access to* Π'*, and conversely* Π' *is decidable by a randomized reduction* A' *with oracle access to* Π*. Then* Π *is checkable if and only if* Π' *is checkable.*

By definition **TFNP** relations are checkable: given an oracle $\mathcal{O}$, check that $(x, \mathcal{O}(x)) \in R$ indeed holds. Because the relation is total, $\mathcal{O}(x)$ can never claim that no solution exists.

TFNP contains important problems such as Nash equilibrium [38]. Megiddo and Papadimitriou [32] observed that $\mathbf{P}^{\mathbf{TFNP}} \subseteq \mathbf{NP} \cap \mathbf{co\text{-}NP}$ and therefore no **TFNP** oracle can be used to decide SAT under deterministic oracle reductions unless $\mathbf{NP} = \mathbf{co\text{-}NP}$. Consequently, it is unlikely that problems such as finding Nash equilibrium are **NP**-hard under deterministic oracle reductions. Since **TFNP** is trivially reducible to **NP**, as a corollary of Theorem 1.3 we observe the following about the hardness of **TFNP**:

Corollary 1.4. *If* $\mathrm{SAT} \in \mathbf{BPP}^{\mathbf{TFNP}}$*, then* SAT *is checkable.*

As discussed above, one consequence is that if finding Nash equilibrium can be used to solve SAT via a *randomized* oracle reduction, then SAT is checkable. Corollary 1.4 can be interpreted as an incomparable version of the theorem of Megiddo and Papadimitriou [32], where our version handles randomized reductions but arrives at a different (weaker) conclusion.

1.3.2 Extending Beigel's theorem to testability

So far we have considered worst-case to worst-case reductions, but equally interesting are worst-case to average-case randomized oracle reductions. Such a reduction is an efficient oracle algorithm A that is guaranteed to decide Π correctly (on all inputs) given oracle access to $\mathcal{O}$ that decides Π' on average over an input distribution D, namely when $\Pr_{x \leftarrow D}[\mathcal{O}(x) \neq \Pi(x)] \leq 1/\operatorname{poly}(n)$.

A notion of checkability for average-case problems also exists which following Blum et al. [9] is called testability. We say that C is a tester for Π' with respect to the input distribution D if the following holds: whenever C is given a program P that correctly decides all x then C must also decide all x. If C is given access to P such that $\Pr_{x \leftarrow D}[P(x) \neq \Pi(x)] \geq \delta$ for some error parameter δ, then C outputs "error" with high probability. This can even be extended to a notion of testability for search problems; see Definition 2.7 for a formal definition. We remark that we allow C to run in time depending on the input length, while the related notion of property testing [9, 24] typically considers testing algorithms that run in time depending only on $1/\delta$. Also, typically C will only use oracle access to P.

Our first observation is that Beigel's theorem can be extended to encompass testability:

Theorem 1.5 (Extended Beigel's theorem, informal)**.** *Suppose* Π *is decidable by a worst-case to average-case randomized reduction* A *with oracle access to* Π'*, and conversely* Π' *is decidable by a (worst-case to worst-case) randomized reduction* A' *with oracle access to* Π*. Then if* Π' *is testable, then both* Π *and* Π' *are checkable.*

Application to basing one-way functions on NP-hardness. One of the main goals of theoretical cryptography is to base the existence of cryptographic primitives on reasonable assumptions. $\mathbf{P} \neq \mathbf{NP}$ is a minimal assumption, and it would be ideal if one could construct one-way functions from this assumption. Such a result might be proved by giving an efficient reduction A that uses an oracle $\mathcal{O}$ that inverts a (candidate) one-way function in order to decide SAT. However, it has been shown in a series of works [15, 10, 3, 39] that such a hope is most likely false if A is non-adaptive, since it would imply that $\mathbf{NP} \subseteq \mathbf{coAM}$.

We know less about whether or not one can base one-way functions on **NP**-hardness via an *adaptive* reduction. There are certain cryptographic [27, 2, 34] and complexity-theoretic [16, 28, 5] settings where adaptivity in the reduction buys more power, and so it is important to understand whether adaptive reductions basing one-way functions on **NP**-hardness are possible. Brassard [12] showed that one-way *permutations* cannot be based on **NP**-hardness unless $\mathbf{NP} \subseteq \mathbf{coAM}$. Pass [39] showed that if (general) one-way functions can be based on **NP**-hardness, then certain *witness-hiding* protocols do not exist. However Haitner et al. [26] showed that it is unlikely that known witness-hiding protocols are of the type studied by [39]. More recently, Haitner et al. [25] study the case of stronger primitives such as collision-resistant hash functions and constant-round statistically hiding commitments that can be broken by recursive collision finders. Their result for arbitrarily adaptive reductions states that if such primitives, say collision-resistant hash functions, can be based on **NP**-hardness, then there exists an interactive proof system for UNSAT where the honest prover strategy can be implemented in $\mathbf{BPP}^{\mathbf{NP}}$. Using the characterization of checkability in terms of interactive proofs given in [7], this then implies that SAT is checkable. However we note that their technique applies only to primitives such as collision-resistant hash functions, and does not extend to one-way functions.

Here we show using different techniques that an adaptive reduction that bases the existence of a one-way function on the hardness of SAT would imply that SAT is checkable. The observation is that a one-way function is testable (for a suitable notion of testability for search problems), and therefore we can apply Theorem 1.5.

Corollary 1.6 (Joint with Thomas Holenstein)**.** *If there is a randomized oracle reduction that uses an oracle* $\mathcal{O}$ *which inverts a one-way function in order to decide* SAT*, then* SAT *is checkable.*

We note that the proof that was joint with Holenstein was direct and did not go through the formalism of testability developed here.

Application to basing hardness of learning on $\mathbf{NP}$*-hardness.* It was shown in [4] that if there exists a reduction uses a PAC learning algorithm to decide SAT (in other words, it bases the hardness of PAC learning on $\mathbf{NP}$-hardness), then there also exists a reduction that uses an algorithm that inverts auxiliary-input one-way functions [37] into an algorithm that solves SAT. In the same way that inverting a one-way function is testable, so is inverting auxiliary-input one-way functions. Therefore, using Theorem 1.5, we can deduce the following:

Corollary 1.7. *If there is a randomized oracle reduction that uses an oracle solving the PAC learning problem in order to decide* SAT*, then* SAT *is checkable.*

Application to worst-case to average-case reductions for SAT Bogdanov and Trevisan [10] prove that non-adaptive worst-case to average-case randomized reductions for $\mathbf{NP}$ imply that $\mathbf{NP} \subseteq \mathbf{co\text{-}NP}/\operatorname{poly}$. The latter implies that $\mathbf{PH} = \Sigma_3$ which is considered implausible. We will consider the same problem but allow adaptive reductions. Using a technique of [15] we observe that:

Theorem 1.8. *Every language* $L \in \mathbf{NP}$ *is testable by a* non-uniform *tester.*

Theorem 4.3 states this formally. Combined with Theorem 1.5 this will imply the following.

Corollary 1.9 (Informal)**.** *Suppose there is a (possibly adaptive) worst-case to average-case oracle reduction for the relation* $R \in \mathbf{NP}$ *(where* R *is* not *necessarily* $\mathbf{NP}$*-complete). Namely the reduction uses an oracle* $\mathcal{O}$ *solving* R *on average in order to decide* R *on all inputs (with high probability). Then* R *has a non-uniform instance checker.*

As pointed to us by one of the anonymous reviewers, the original Nisan's proof to show that Permanent has a two-prover proof system can be used to eliminate the needed advice in Corollary 1.9 for $R = \text{SAT}$ if the worst-case to average-case reduction for R is of a special form. Namely if the language R is downward self-reducible (which is the case for SAT) and also has a worst-case to average-case reduction where the reduction never asks queries y of length $|y| > n$, then R has a *uniform* instance checker (see Observation 4.6.)

1.4 Randomized vs. deterministic oracle reductions

As already noted throughout the introduction, if we restrict our attention to deterministic oracle reductions then all of the results we prove are already known (and indeed in most cases stronger conclusions hold). We remark that, although it is commonly conjectured that $\mathbf{P} = \mathbf{BPP}$ [47, 8, 35, 30], the techniques used to prove derandomization under commonly held hardness assumptions do not necessarily say that for some oracle $\mathcal{O}$, it holds that $\mathbf{P}^{\mathcal{O}} = \mathbf{BPP}^{\mathcal{O}}$. Indeed, there are examples of $\mathcal{O}$ where $\mathbf{P}^{\mathcal{O}} \neq \mathbf{BPP}^{\mathcal{O}}$. Thus one cannot apply the general derandomization theorems above to the previously known results about, say, $\mathbf{P}^{\mathbf{PrSZK}}$ and $\mathbf{P}^{\mathbf{TFNP}}$ to derive our results. We will argue directly about $\mathbf{BPP}^{\mathbf{PrSZK}}$, $\mathbf{BPP}^{\mathbf{TFNP}}$, etc. without relying on derandomization assumptions.

2 Preliminaries

For a random variable X, by $x \leftarrow X$ we mean that x is sampled according to the distribution of X. For a set S, by U_S we mean the random variable with uniform distribution over S. For a set

S, by $x \overset{\mathrm{R}}{\leftarrow} S$ we mean $x \leftarrow U_S$. By U_n we mean the random variable uniformly distributed over $\{0,1\}^n$. For $\alpha \in \mathbb{R}$, by $[\pm\alpha]$ we mean the interval $[-\alpha, +\alpha]$.

As is standard convention, when we say an event occurs with negligible probability we mean it occurs with probability $n^{-\omega(1)} = \mathrm{neg}(n)$ and we say that it occurs with overwhelming probability if it occurs with probability $1 - \mathrm{neg}(n)$. Here, n is the input length. We call an algorithm *efficient* if it runs in time $\mathrm{poly}(n)$.

We assume that the reader is familiar with the idea of promise problems [14] as well as the standard complexity classes **BPP**, **NP**, **PH**, **SZK**, **AM**, etc., which we take to be classes of promise problems (and not just languages). We will write simply **SZK** rather than **PrSZK** from now on with this understanding.

For every promise problem $\Pi = (\Pi_Y, \Pi_N)$ we write $\Pi(x) = 1$ if $x \in \Pi_Y$ and $\Pi(x) = 0$ if $x \in \Pi_N$. A language L is simply a promise problem $(L, \overline{L})$, where $\overline{L}$ is the complement of L.

To extend promise decision problems to search problems, we work with *promise relations*.

Definition 2.1 (Promise relations)**.** *Let $R = (R_Y, R_N)$ where $R_Y \subseteq \{0,1\}^* \times \{0,1\}^*$ and $R_N \subseteq \{0,1\}^* \times \{0,1\}^*$ such that $R_Y \cap R_N = \varnothing$. We call R a* promise relation *and sometimes write for shorthand $(x, y) \in R$ to mean $(x, y) \in R_Y$. We say R is a* standard *relation if $R_Y = \overline{R_N}$.*

For any promise relation $R = (R_Y, R_N)$, define $R(x) = \{y \mid (x, y) \in R_Y\}$. A relation R is *total* if for every x, $R(x) \neq \varnothing$. Note that the decision problem for total relations is trivial, but the search problem may still be hard.

In the following, we will consider $\varnothing$ to also denote a special symbol signifying the empty set, so that search algorithms are allowed to output $\varnothing$ to mean that the algorithm cannot find a solution. We will also abuse notation slightly and allow $A(x) \in R(x)$ to be a true statement if $R(x)$ is empty and $A(x)$ outputs the special symbol "$\varnothing$".

We say an algorithm A solves the language L (resp. the relation R) if for all x, it holds that $A(x) = L(x)$ (resp. $A(x) \in R(x)$). If the algorithm A is randomized, the latter should hold with overwhelming probability (for all x's).

Relations in NP and AM

Definition 2.2 (**NP** Relation)**.** *A standard relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is an* **NP** *relation if the set R is accepted by an efficient algorithm, and for all $(x, y) \in R$, $|y| \leq \mathrm{poly}(|x|)$.*

Definition 2.3 (**TFNP**)**.** *A standard relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is in the class* **TFNP** *if R is an* **NP** *relation and R is total.*

Definition 2.4 (**AM** relations)**.** *For a promise relation $R = (R_Y, R_N)$, define the promise problem $M^R = \{M^R_Y, M^R_N\}$ where $M^R_Y = \{(x, y) \mid (x, y) \in R_Y\}$ and $M^R_N = \{(x, y) \mid (x, y) \in R_N\}$. R is an* **AM** *relation if $M^R \in$* **AM***, and also $\forall (x, y) \in R_Y$ it holds that $|y| \leq \mathrm{poly}(|x|)$.*

Definition 2.5 (**TFAM**)**.** *A promise relation $R = (R_Y, R_N)$ is in* **TFAM** *if R is a total relation and also an* **AM** *relation.*

Since our definition is for promise problems, instances $(x, y) \notin R_Y \cup R_N$ can behave arbitrarily. It follows immediately from the definitions that **TFNP** $\subseteq$ **TFAM**.

When are AM relations interesting? In contrast to **NP**, not every **AM** decision problem has an interesting search version. This is because an **AM** decision problem might not have well-defined witnesses. For example, consider the standard Graph Non-Isomorphism (GNI) protocol, where given (G_1, G_2) the verifier picks random bit b and sends a random permutation of G_b to the prover, who must then respond with b. There is no fixed "witness" of the non-isomorphism, so the natural **AM** relation of Graph Non-Isomorphism is trivial, $((G_1, G_2), y) \in R_{\mathsf{GNI}}$ iff $(G_1, G_2) \in \mathsf{GNI}$, and the value y is ignored. On the other hand, some interesting problems not known to be in **NP** (such as Entropy Difference ED) do have interesting witnesses that are only known to be verifiable using randomized protocols and not deterministically (see Section 3).

Checkability

Definition 2.6 (Checkability)**.** *A relation R is* checkable *if there exists an efficient randomized oracle algorithm A such that for all oracles $\mathcal{O}$, the following holds.*

- *Completeness: Suppose for all x it holds that $\mathcal{O}(x) \in R(x)$. Then for all x, $A^{\mathcal{O}}(x) \in R(x)$ with overwhelming probability.*
- *Soundness: For any x such that $\mathcal{O}(x) \notin R(x)$, $A^{\mathcal{O}}(x)$ with overwhelming probability either outputs some $y \in R(x)$ or outputs a special error symbol $\perp$, which indicates $\mathcal{O}(x)$ may be wrong.*

This definition coincides with the definition of [7] for checkability of promise problems Π if one considers the relation $(x, y) \in R$ iff $\Pi(x) = y$, where $y \in \{0, 1\}$.

It is known [7, 18] that for any $k \geq 2$ and any language L, L is checkable if and only if both L and $\overline{L}$ have an interactive proof system with k provers where the provers are asked only L queries.

An average case notion of checkability was defined by Blum et al. [9], which they called "program testing". (In the following we keep the convention that D_n is a distribution over $\{0, 1\}^n$.)

Definition 2.7 (Testability)**.** *A relation R is δ-*testable *over the ensemble of distributions $D = \{D_n\}$ if there exists an efficient randomized oracle algorithm A such that for all oracles $\mathcal{O}$ the following holds.*

- *Completeness: If $\mathcal{O}(x) \in R(x)$ for all x, then $A^{\mathcal{O}}(1^n)$ accepts with overwhelming probability.*
- *Soundness: If $\Pr_{x \leftarrow D_n}[\mathcal{O}(x) \notin R(x)] > \delta$, then $A^{\mathcal{O}}(1^n)$ outputs $\perp$ with overwhelming probability, indicating that $\mathcal{O}$ may not $(1 - \delta)$-solve R.*

As with checkability, the definition of program testing for decision problems Π follows immediately by considering the relation $(x, y) \in R$ iff $\Pi(x) = y$.

Worst-case to average-case reductions

Let D_n be a distribution over $\{0, 1\}^n$. We say that the ensemble of distributions $D = \{D_n\}$ is *samplable* if there is an efficient randomized algorithm S which the output of $S(1^n)$ is distributed according to D_n. For $\rho = \rho(n)$, we say that an oracle $\mathcal{O}$ ρ-solves the relation R over D if for every n it holds that $\Pr_{x \leftarrow D_n}[\mathcal{O}(x) \in R(x)] \geq \rho$.

Definition 2.8 (Worst-case to average-case reductions.)**.** *Let $\rho = \rho(n)$, and let D be an ensemble of distributions. We say that the relation R reduces to ρ-solving the relation R' over D if there is an efficient randomized oracle algorithm A such that $\Pr[A^{\mathcal{O}}(x) \in R(x)] = 1 - \operatorname{neg}(n)$ for every $x \in \{0,1\}^n$ whenever $\mathcal{O}$ ρ-solves the relation R' over D.*

Definition 2.9 (Worst-case to average-inverting reductions)**.** *Let $f = \{f_n \colon \{0,1\}^n \to \{0,1\}^n\}$ be a family of functions. We say that the relation R reduces to ρ-inverting f if R reduces to ρ-solving the relation R^f over the ensemble of distributions D^f where $R^f = \{(y,x) \colon f(x) = y\}$ and $D^f_n = f(U_n)$.*

Note that in both Definitions 2.8 and 2.9, the reduction is allowed to ask oracle queries y of larger length $|y| = \operatorname{poly}(|x|)$ where x is the input to the reduction.

3 Real-valued Total Functions

3.1 Definitions and Preliminaries

We begin by defining the class of relations $\mathbb{R}$-**TUAM**, which intuitively captures functions $f : \{0,1\}^* \to \mathbb{R}$ such that given (x,y), it is possible to verify using an **AM** protocol that $|y - f(x)|$ is small.

Definition 3.1 ($\mathbb{R}$-**TUAM**)**.** *A function $f : \{0,1\}^* \to \mathbb{R}$ is in $\mathbb{R}$-**TUAM** (denoting "real-valued total unique **AM**") if for every $\varepsilon \geq 1/\operatorname{poly}(n)$, the following relation $R = (R_Y, R_N)$ is in **AM**:*

1. *$R_Y = \{(x, f(x)) \mid x \in \{0,1\}^*\}$*
2. *$R_N = \{(x,y) \mid x \in \{0,1\}^*, y \in \mathbb{R}$ s.t. $|y - f(x)| > \varepsilon\}$*

We let $\mathbf{BPP}^{\mathbb{R}\text{-}\mathbf{TUAM}}$ denote the class of promise problems that are decidable by a randomized oracle algorithm given oracle access to a real-valued function whose output is verifiable in **AM**, formalized as follows.

Definition 3.2 ($\mathbf{BPP}^{\mathbb{R}\text{-}\mathbf{TUAM}}$)**.** *$\Pi \in \mathbf{BPP}^{\mathbb{R}\text{-}\mathbf{TUAM}}$ if there exists an $f \in \mathbb{R}$-**TUAM**, an oracle algorithm A, and an $\varepsilon(n) = 1/\operatorname{poly}(n)$ such that for all oracles $\mathcal{O}$ satisfying $\forall x,\ |f(x) - \mathcal{O}(x)| \leq \varepsilon(|x|)$, it holds for all $z \in \Pi_Y \cup \Pi_N$ that $A^{\mathcal{O}}(z) = \Pi(z)$ with overwhelming probability over the random coins of A.*

Our main interest in studying $\mathbb{R}$-**TUAM** is its relationship to **SZK**. We will characterize **SZK** by its complete problem: Entropy Difference ED. See e.g., [46] for an introduction and other definitions of **SZK**.

Definition 3.3 (**SZK** and ED, [23])**.** *A promise problem Π is in **SZK** if and only if it is Karp-reducible to the following promise problem $\mathsf{ED} = (\mathsf{ED}_Y, \mathsf{ED}_N)$. Instances of ED are pairs of circuits (X_1, X_2) where each circuit $X : \{0,1\}^m \mapsto \{0,1\}^n$ is identified by its output distribution $X(U_m)$. Let $H(X) = H(X(U_m))$ denote the Shannon entropy of $X(U_m)$. Then:*

1. *$(X_1, X_2) \in \mathsf{ED}_Y$ iff $H(X_1) \geq H(X_2) + 1$.*
2. *$(X_1, X_2) \in \mathsf{ED}_N$ iff $H(X_2) \geq H(X_1) + 1$.*

We first recall that the entropy can be estimated using an **AM** protocol, whose proof we give in the appendix.

Lemma 3.4 ([17, 1, 22])**.** *For every $\varepsilon > 1/\operatorname{poly}(n)$, there is an* **AM** *protocol that on input (X, y) accepts if $H(X) = y$ and rejects if $|H(X) - y| > \varepsilon$.*

Proposition 3.5. $\mathbf{BPP}^{\mathbf{SZK}} \subseteq \mathbf{BPP}^{\mathbb{R}\text{-}\mathbf{TUAM}}$

Proof. We show how to implement an ED oracle using a $\mathbb{R}$-**TUAM** oracle for any $\varepsilon < 1$. Consider the function $f(X_1, X_2) = H(X_1) - H(X_2)$. This function is in $\mathbb{R}$-**TUAM** because by Lemma 3.4 the entropy of a circuit can be approximated using an **AM** protocol.

We claim that given any oracle $\mathcal{O}$ that solves (the search problem of) the relation R with up to ε error, one can decide ED: given (X_1, X_2), query (X_1, X_2) from $\mathcal{O}$ to get $y = \mathcal{O}(X_1, X_2)$ and accept iff $y > 0$. Since we are guaranteed that $|(H(X_1) - H(X_2)) - y| \leq \varepsilon$, it therefore holds that if $H(X_1) > H(X_2) + 1$ then $y > 1 - \varepsilon > 0$ and if $H(X_1) < H(X_2) - 1$ then $y < -1 + \varepsilon < 0$. $\square$

3.2 Power of $\mathbb{R}$-TUAM

We now prove Theorem 1.1, namely $\mathbf{BPP}^{\mathbb{R}\text{-}\mathbf{TUAM}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$.

Proof of Theorem 1.1. Fix any $\Pi \in \mathbf{BPP}^{\mathbb{R}\text{-}\mathbf{TUAM}}$, then by definition there exists $f \in \mathbb{R}\text{-}\mathbf{TUAM}$, an efficient oracle algorithm A, and a parameter $\varepsilon \geq 1/\operatorname{poly}(n)$ satisfying Definition 3.2. We will follow the natural strategy outlined in Section 1.1, but in order to make it work we need to overcome the two difficulties outlined there. The first difficulty is overcome simply because $\mathbb{R}$-**TUAM** is total, and therefore the prover can never respond that a query x does not satisfy the promise. To overcome the second difficulty, we will exploit the fact that $\mathbb{R}$-**TUAM** is a *unique* relation. By adding some noise to the prover's responses, which we can check is close to the unique true answer using the **AM** proof for the $\mathbb{R}$-**TUAM** relation, we will prevent it from making its answers dependent on the verifier's random coins. We now explain how this is done.

Define the (randomized) oracle $\mathcal{O}_\varepsilon$ as follows. $\mathcal{O}_\varepsilon(x)$ first chooses a uniformly random $\alpha_x \overset{\mathrm{R}}{\leftarrow} [-\varepsilon/2, +\varepsilon/2]$ which we call the "randomizer" of the query x and takes $y = f(x) + \alpha_x$. Then $\mathcal{O}_\varepsilon(x)$ will round y and output $\lfloor y \rceil_\varepsilon$ where $\lfloor y \rceil_\varepsilon$ denotes the integer multiple of ε that is closest to y. Note that it always holds that $\mathcal{O}_\varepsilon \in [f(x) \pm \varepsilon]$, and so by the definition of A and $\mathcal{O}_\varepsilon$, it holds that

$$\Pr_{\mathcal{O}_\varepsilon, A}[A^{\mathcal{O}_\varepsilon}(z) = \Pi(z)] \geq 1 - n^{-\omega(1)} \tag{1}$$

for all $z \in \Pi_Y \cup \Pi_N$ (where $\Pi(z) = 1$ if $z \in \Pi_Y$ and $\Pi(z) = 0$ if $z \in \Pi_N$). The reason is that one can choose and fix the randomness of $\mathcal{O}_\varepsilon$ first and then Inequality 1 holds by the definition of A. In the following, let $p(n) = \operatorname{poly}(n)$ be an upper bound on the number of oracle queries made by A and let ω denote random coins used to run A. Without loss of generality we assume that A does not ask any query x more than once. We also write $\mathcal{O}_\varepsilon(x)$ more explicitly as $\mathcal{O}_\varepsilon(x, \alpha_x)$ to denote the value of the randomizer α_x used for the answer to the query x.

To prove that $\Pi \in \mathbf{AM}$ it suffices to show an **AM** protocol where either the verifier catches the prover cheating and rejects or (if the prover is honest) the output of the verifier is statistically close to the output of $A^{\mathcal{O}_\varepsilon}$. This way the verifier either catches the cheating prover or gets a good emulation of $A^{\mathcal{O}_\varepsilon}$ which she can use to take her final decision and choose to accept or reject.

The intuition. The idea is that the verifier will select random coins ω for the execution of A along with real numbers $\alpha_1, \ldots, \alpha_{p(n)}$ drawn uniformly at random from $[\pm\varepsilon/2]$ and send these to the prover, and the prover will use ω to run A responding oracle queries according to $\mathcal{O}_\varepsilon$ while using α_i as the randomizer for the i'th query x_i. The prover sends back all the $f(x_i)$ to the verifier, who checks for all $i \in [p(n)]$ using $f(x_i)$ and α_i that the responses $\lfloor f(x_i) + \alpha_i \rceil_\varepsilon$ give a consistent and accepting execution of the reduction. Furthermore, we assumed that $f \in \mathbb{R}$-**TUAM**, so let R be the corresponding relation guaranteed by Definition 3.1. Since R is an **AM** relation, we will also require that the prover give a proof that all $f(x_i)$'s are correct up to some error δ.

Completeness of this strategy is clear because the honest prover can always prove that $(x, f(x)) \in R$, and so each oracle query x is answered with the same distribution as $A^{\mathcal{O}_\varepsilon}$. Soundness follows as well. The reason is that the only time the prover can bias the value $\lfloor f(x_i)+\alpha_i \rceil_\varepsilon$ is when $(f(x_i)+\alpha_i \mod \varepsilon) \approx \varepsilon/2$ for some i, i.e., by slightly perturbing $f(x_i)$ the prover can cause $f(x_i) + \alpha_i$ to be rounded either up or down. But it is easy to see that this bad situation is unlikely, namely for each query, $\Pr_{\alpha_i}[(f(x_i) + \alpha_i \mod \varepsilon) \in [\varepsilon/2 \pm \delta]] = 2\delta/\varepsilon$. By taking $\delta/\varepsilon \ll 1/p(n)$, where $p(n)$ is the total number of queries, the verifier gets an emulation of the reduction which is statistically close to an honest emulation and therefore the soundness of the protocol follows from the definition of the reduction.

The AM protocol for the problem Π

Protocol 3.6. *Common input: instance z.*

1. *The verifier V_A sends a random seed ω that will be used to execute A. V_A also sends random numbers $\alpha_1, \ldots, \alpha_{p(n)} \xleftarrow{R} [-\varepsilon/2, +\varepsilon/2]$.*

2. *The prover P_A emulates the execution of A using random coins ω, where the prover answers the i'th oracle query x_i with $\lfloor y_i + \alpha_i \rceil_\varepsilon$. The prover sends back to the verifier the values (x_i, y_i) for all $i \in [p(n)]$. (An honest prover sets $y_i = f(x_i)$.)*

3. *In parallel for all $i \in [p(n)]$, the verifier engages the prover in the* **AM** *protocol that $f(x_i) = y_i$ with approximation error $\delta = \frac{\varepsilon}{np(n)}$. If any of these protocols reject, then the verifier rejects.*

4. *The verifier checks for $i = 1, \ldots, p(n)$ that emulating A with the i'th query x_i answered by $\lfloor y_i + \alpha_i \rceil_\varepsilon$ leads to A asking the $i+1$'st query x_{i+1}, and so on, and accepts iff A accepts.*

Since $\mathbf{BPP}^{\mathbb{R}\text{-}\mathbf{TUAM}}$ is closed under complement, proving that the above protocol decides Π suffices to prove Theorem 1.1.

Completeness. For any $z \in \Pi_Y$ and for any query x_i, the honest prover computes $y_i = f(x_i)$ and uses $\lfloor y_i + \alpha_i \rceil_\varepsilon$ as the response to x_i. Therefore the oracle answers are distributed identically to $\mathcal{O}_\varepsilon(x_i, \alpha_i)$, and so by Inequality 1 the verifier accepts with overwhelming probability.

Soundness. Fix any $z \in \Pi_N$, and let P' be a possibly cheating prover. For each query x_i let y_i be the claim of P' for $f(x_i)$. If for any $i \in [p(n)]$ it holds that $|y_i - f(x_i)| > \frac{\varepsilon}{np(n)}$, then with overwhelming probability one of the **AM** protocols in Step 3 will fail by the soundness condition of the $\mathbb{R}$-**TUAM** relation, and so V_A will reject.

So let us suppose that the strategy of P' is restricted to always claim some y_i such that $|y_i - f(x_i)| \le \frac{\varepsilon}{np(n)}$. Now look at the oracle answer $\lfloor y_1 + \alpha_1 \rceil_\varepsilon$ used for the first query x_1 in the emulation of the reduction. If it holds that $(f(x_1) + \alpha_1 \mod \varepsilon) \notin [\varepsilon/2 \pm \delta]$, then for all y_1 satisfying $|y_1 - f(x_1)| \le \delta$, it holds that $\lfloor y_1 + \alpha_1 \rceil_\varepsilon = \lfloor f(x_1) + \alpha_1 \rceil_\varepsilon$ and so in this case the prover is unable to control the value of $\lfloor y_1 + \alpha_1 \rceil_\varepsilon$. But since α_1 was chosen at random from $[\pm\varepsilon/2]$, it holds that:

$$\Pr_{\alpha_1 \overset{\mathrm{R}}{\leftarrow} [\pm\varepsilon/2]} [(f(x_1) + \alpha_1 \mod \varepsilon) \in [\varepsilon/2 \pm \delta]] = 2\delta/\varepsilon \ .$$

It means that with probability $\geq 1 - 2\delta/\varepsilon$ over the choice of $\alpha_1 \overset{\mathrm{R}}{\leftarrow} [\pm\varepsilon/2]$ the prover does not have any control over the first oracle answer used in the emulation of the reduction. The same argument holds for the i'th query for any $i \in [p(n)]$.

It follows by the union bound that the total statistical distance between the set of query/answer pairs generated by $A^{\mathcal{O}_\varepsilon}$ and the query/answer pairs generated by (P', V_A) is bounded by at most $\frac{2}{n}$. Therefore, the probability that (P', V_A) accepts is bounded by at most $\leq 2/n + n^{-\omega(1)}$. By repeating the overall protocol in parallel one can reduce this error to be negligible. $\square$

Theorem 1.1 and Proposition 3.5 yield Corollary 1.2, namely $\mathbf{BPP}^{\mathbf{SZK}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$.

4 Reductions that Imply Checkability of SAT

We saw in Section 1.3.1 that a result of Beigel as cited in [7] (Theorem 1.3) along with the fact that **TFNP** is trivially checkable implies the following theorem (see Definition 2.3 for a formal definition of **TFNP**).

Theorem 4.1. *For any relation $R \in$* **TFNP**, *if* SAT *can be reduced to (the search problem of) R through a randomized reduction, then* SAT *is checkable.*

4.1 Extending Beigel's theorem to testability

We will prove Theorem 4.2 a variant of Theorem 1.3 for the case in which one of the reductions in the hypothesis has the extra feature of being worst-case to average case (see Definition 2.8). From this stronger assumption we conclude a stronger consequence as follows.

Theorem 4.2 (Restating Theorem 1.5, formally)**.** *Let $\delta = \delta(n)$, let R and R' be two relations, and let D be an ensemble of distributions. Suppose that solving R reduces to $(1-\delta)$-solving R' over D, and also suppose that there is a randomized (worst-case to worst-case) reduction from R' to R. Then if R' is δ-testable over D (resp. non-uniformly), then R and R' are both checkable (resp. non-uniformly).*

Proof. (of Theorem 4.2) We first prove the theorem for the uniform case when there is no advice.

Let A_R be the randomized oracle reduction from solving R to $(1-\delta)$-solving R' over D and let $\ell = \mathrm{poly}(n)$ be an upper-bound on the length of the oracle queries of A_R. Notice that, in particular, A_R is also a worst-case to worst-case reduction from R to R'. Let $A_{R'}$ be the randomized (worst-case to worst-case) reduction from R' to R. By Theorem 1.3 and the fact that A_R is also a worst-case to worst-case reduction from R to R', it follows that R is checkable if and only if R' is checkable. Therefore in the following it suffices to show that R is checkable.

One can think of $\mathcal{O}' = A_{R'}^{\mathcal{O}}$ as an oracle which hopefully solves the relation R' correctly. Given a perfect oracle $\mathcal{O}$ for R, by running the reduction $A_{R'}$ with oracle access to $\mathcal{O}$ one can efficiently simulate the oracle $\mathcal{O}'$ which solves the relation R'. Suppose *w.l.o.g.* that $\ell = \mathrm{poly}(n)$ is also an upper-bound on the length of the oracle queries of $A_{R'}$.

Suppose that $T_{R'}$ is a δ-tester for R'. Given the input x, and having access to the oracle $\mathcal{O}$ (which is supposed to solve R') the instance checker $C_R^{\mathcal{O}}(x)$ for the relation R does the following.

- For all $i \in [\ell(n)]$, C_R runs the δ-tester $T_{R'}$ over the oracle $\mathcal{O}' = A_{R'}^{\mathcal{O}}$ to make sure that it decides R' with probability at least $1-\delta$ over D_i for the input length i. $C_R^{\mathcal{O}}(x)$ outputs $\perp$ if $T_{R'}^{\mathcal{O}'}(1^i)$ outputs $\perp$ for any $i \in [\ell]$.
- Otherwise, $C_R(x)$ outputs whatever $A_R^{\mathcal{O}'}(x)$ outputs.

The completeness of C_R is immediate, because if $\mathcal{O}$ is a perfect oracle for R' then for each query y, $A_{R'}^{\mathcal{O}}(y) \in R'(y)$ holds with overwhelming probability. In this case with overwhelming probability $T_{R'}$ will accept also and C_R will output $z \in R(x)$.

The soundness of C_R holds by the soundness of $T_{R'}$ and the definition of A_R. Namely either $\mathcal{O}' = A_{R'}^{\mathcal{O}}$ $(1-\delta)$-solves R' over D_i for $i \leq \ell(n)$, or else $T_{R'}$ outputs $\perp$ with overwhelming probability. In the first case, by the correctness of A_R, the checker C_R gives a correct output with overwhelming probability.

For non-uniform testers, it is clear that the above reduction still holds except one needs to hardwire into C_R the advice strings of $T_{R'}$ for all input lengths $i \leq \ell(n)$. $\square$

Theorem 4.3 (Formal statement of Theorem 1.8)**.** *Let R be any* **NP** *relation, $\delta = 1/\operatorname{poly}(n)$, and D_n be any ensemble of samplable distributions. Then R can be δ-tested over D_n given $\lceil \log \frac{2}{\delta} \rceil = O(\log n)$ bits of advice. In particular the advice for length n inputs can be any s_n such that $s_n \leq \Pr_{y\leftarrow D_n}[R(y) \neq \varnothing] \leq s_n + \delta/2$.*

The intuition. Suppose that we are given $s_n \approx \Pr_{y\leftarrow D_n}[R(y) \neq \varnothing]$ (the approximate fraction of YES instances of R over D_n). If we sample enough points $y_1, \dots, y_k \leftarrow D_n$, then by Chernoff we anticipate roughly a s_n fraction of y_i's to satisfy $R(y_i) \neq \varnothing$. As the first step the checker for R simply verifies that $s_n \approx |\{i \mid \mathcal{O}(y_i) \neq \varnothing\}|/k$ holds. This yet does not mean that with high probability over $y \leftarrow D_n$, $\mathcal{O}(y)$ returns the right answer. But since R is an **NP**-relation we can always make sure that if $\mathcal{O}(y)$ returns $z \neq \varnothing$, then $(y,z) \in R$. By enforcing this extra check over the solutions $z_i = \mathcal{O}(y_i)$, the oracle $\mathcal{O}$ can be wrong only in "one direction": to return $\varnothing$ for a query y that $R(y) \neq \varnothing$. But if $\mathcal{O}$ does so significantly, then it will change its bias $\Pr_{y\leftarrow D_n}[\mathcal{O}(y) \neq \varnothing] \ll s_n$ and it can be detected. A very similar trick is used in [15, 10, 3]. The difference between our setting and [15, 10, 3] is that they deal with *provers* (rather than oracles) that are stateful and might cheat more intelligently by answering their queries depending on all the queries that they are asked. That makes the job of [15, 10, 3] potentially harder, but they bypass this difficulty by putting strong restrictions on the adaptivity of the reduction (which is not the case for our result). Also [36, 44] use this technique in another setting where the advice $s_n \approx \Pr_{y\leftarrow D_n}[\mathcal{O}(y) \neq \varnothing]$ is used to construct a non-uniform reduction that $(1 - 1/\operatorname{poly}(n))$-solves R over D_n given access to any oracle that solves R correctly only with probability $\geq 1/2 + n^{-1/3+\varepsilon}$.

Proof. (of Theorem 4.3) Let $\mathcal{O}$ be the oracle that is going to be tested for the relation R. The tester $T_R^{\mathcal{O}}$ acts as follows.

1. Let $k = n/\delta^2$. For $i \in [k]$ sample $x_i \leftarrow D_n$.
2. For $i \in [k]$, ask x_i from $\mathcal{O}$ to get $y_i = \mathcal{O}(x_i)$.

3. If for any $i \in [k]$, it holds that $y_i \neq \varnothing$ but $(x_i, y_i) \notin R$ (which can be checked efficiently), then output $\perp$.

4. If $\frac{|\{i\colon \mathcal{O}(x_i) \in R(x)\}|}{k} < s_n - \frac{\delta}{6}$ then output $\perp$, otherwise accept.

Completeness. If $\mathcal{O}(x) \in R(x)$ for all $x \in \{0,1\}^n$, then T_R never outputs $\perp$ in Step 3. We show that the probability of outputting $\perp$ in Step 4 is negligible. Let $p = \Pr_{x \leftarrow D_n}[R(x) \neq \varnothing]$ and $p' = \frac{|\{i\colon \mathcal{O}(x_i) \in \mathbb{R}(x)\}|}{k}$ be the empirical estimate of p. By Chernoff, it holds that $\Pr_{x_1,\dots,x_k}[|p' - p| > \varepsilon] < 2e^{-2k\varepsilon^2}$. Now since it is guaranteed that $s_n \leq p$, therefore it holds that

$$\begin{aligned}\Pr[p' < s_n - \delta/6] \leq \Pr[p' < p - \delta/6] \leq \Pr[|p' - p| > \delta/6] \\ < 2e^{-2k(\delta/6)^2} = 2e^{-n/18} = \operatorname{neg}(n).\end{aligned}$$

Soundness. Suppose $\Pr_{x \leftarrow D_n}[\mathcal{O}(x) \notin R(x)] \geq \delta$. Then either it holds that $\Pr_{x \leftarrow D_n}[R(x) = \varnothing \wedge \mathcal{O}(x) \neq \varnothing] \geq \delta/6$ or it holds that $\Pr_{x \leftarrow D_n}[R(x) \neq \varnothing \wedge \mathcal{O}(x) \notin R(x)] \geq 5\delta/6$. We show that in both cases T_R outputs $\perp$ with overwhelming probability.

If $\Pr_{x \leftarrow D_n}[R(x) = \varnothing \wedge \mathcal{O}(x) \neq \varnothing] \geq \delta/6$ then with probability at least $1 - (1-\delta/6)^k = 1 - \operatorname{neg}(n)$ one of x_i's is sampled such that $R(x_i) = \varnothing$ and $\mathcal{O}(x_i) \neq \varnothing$. In this case $(x_i, \mathcal{O}(x_i)) \notin R$ and so T_R outputs $\perp$ in Step 3.

On the other hand if $\Pr_{x \leftarrow D_n}[R(x) \neq \varnothing \wedge \mathcal{O}(x) \notin R(x)] \geq 5\delta/6$, because $p \leq s_n + \delta/2$ then

$$\begin{aligned}&\Pr_{x \leftarrow D_n}[\mathcal{O}(x) \in R(x)] \leq \Pr_{x \leftarrow D_n}[R(x) \neq \varnothing \wedge \mathcal{O}(x) \in R(x)] \leq \\ &\Pr_{x \leftarrow D_n}[R(x) \neq \varnothing] - 5\delta/6 = p - 5\delta/6 \leq s_n + \delta/2 - 5\delta/6 \\ &= s_n - \delta/3.\end{aligned}$$

So for each i it holds that $\Pr_{x_i \leftarrow D_n}[\mathcal{O}(x_i) \in R(x_i)] \leq s_n - \delta/3$, and thus by Chernoff it holds that

$$\begin{aligned}\Pr[p' \geq s_n - \delta/6] = \Pr[p' \geq (s_n - \delta/3) + \delta/6] < 2e^{-2k(\delta/6)^2} \\ = 2e^{-n/18} = \operatorname{neg}(n).\end{aligned}$$

But if $p' < s_n - \delta/6$, then T_R outputs $\perp$ in Step 4.

$\square$

Worst-case to average-case reductions for NP

For definitions of worst-case to average-case reductions, see Definition 2.8.

Corollary 4.4 (Formal statement of Corollary 1.9)**.** *Let R be an* **NP***-relation, let $\delta = 1/\operatorname{poly}(n)$ and let D be any efficiently samplable ensemble of distributions. If R reduces to $(1-\delta)$-solving R over D, then R has a non-uniform instance checker.*

Corollary 4.5 (Formal statement of Corollary 1.6)**.** *Let $\delta = 1/\operatorname{poly}(n)$, and let f be an efficiently computable family of functions. If* SAT *reduces to $(1-\delta)$-inverting f, then* SAT *is* uniformly *checkable.*

Proof. (of Corollaries 4.4 and 4.5) Since SAT is **NP** complete, there is a (worst-case to worst-case) randomized reduction from the **NP** relation R of Theorem 4.3 to the relation SAT. Therefore Corollary 4.4 follows from Theorem 4.2 and Theorem 4.3 immediately.

Corollary 4.5 also follows from Theorem 4.2 and Theorem 4.3 similarly by using $R = \text{SAT}$ and using the Cook-Levin reduction (to reduce inverting f to solving a SAT instance). But this time we do not need the advice because of the following. Even though R^f might not be a total relation, but it still holds that

$$\Pr_{y \leftarrow D_i^f}[R^f(y) \neq \varnothing] = \Pr_{y \leftarrow f(U_i)}[y \text{ is invertible}] = 1.$$

Thus here we can use the constants $s_i = 1$ as the advice for all the query lengths $i \leq \text{poly}(n)$ that reduction might ask and apply Theorem 4.3, eliminating the need for advice. □

One of the anonymous reviewers indicated to us that for downward self-reducible relations R (which are not necessarily in **NP**) the Corollary 4.4 can be improved to get a *uniform* instance checker if the worst-case to average case reduction is of a restricted form:

Observation 4.6 (Nisan, observed by anonymous reviewer)**.** *Let R be a downward self-reducible relation, let $\delta = 1/\text{poly}(n)$ and let D be any efficiently samplable ensemble of distributions. If R reduces to $(1-\delta)$-solving R over D by oracle reduction A, and if the reduction A never asks queries of length $\ell(n) > n$ from its oracle, then R has a* uniform *instance checker.*

For a sketch of the proof of Observation 4.6 see Appendix A.4

5 Acknowledgements

We thank Thomas Holenstein for great discussions and his contribution in Corollary 1.6. We thank Salil Vadhan for sharing his insight into zero knowledge and promise problems, as well as giving pointers to the literature on promise problems. We also thank Boaz Barak for interesting discussions. We also thanks the anonymous reviewer of CCC '10 who told us about the Observation 4.6.

References

[1] W. Aiello and J. Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *Journal of Computer and System Sciences*, 42(3):327–345, 1991. Preliminary version in *FOCS'87*.

[2] M. Ajtai. The worst-case behavior of schnorr's algorithm approximating the shortest nonzero vector in a lattice. In *STOC '03*, pages 396–406, New York, NY, USA, 2003. ACM. ISBN 1-58113-674-9. doi: http://doi.acm.org/10.1145/780542.780602.

[3] A. Akavia, O. Goldreich, S. Goldwasser, and D. Moshkovitz. On basing one-way functions on np-hardness. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 701–710, 2006.

[4] B. Applebaum, B. Barak, and D. Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Proc. FOCS '08*, pages 211–220, 2008.

[5] L. Babai and S. Laplante. Stronger separations for random-self-reducibility, rounds, and advice. In *In IEEE Conference on Computational Complexity*, pages 98–104, 1999.

[6] J. Blömer and J.-P. Seifert. On the complexity of computing short linearly independent vectors and short bases in a lattice. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 711–720, New York, NY, USA, 1999. ACM. ISBN 1-58113-067-8. doi: http://doi.acm.org/10.1145/301250.301441.

[7] M. Blum and S. Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, 1995. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/200836.200880.

[8] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984. ISSN 0097-5397. doi: http://dx.doi.org/10.1137/0213053.

[9] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.

[10] A. Bogdanov and L. Trevisan. On worst-case to average-case reductions for np problems. *SIAM Journal on Computing*, 36(4):1119–1159, 2006.

[11] R. B. Boppana, J. Håstad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987.

[12] G. Brassard. Relativized cryptography. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 383–391. IEEE Computer Society, 1979.

[13] I. Dinur, G. Kindler, R. Raz, and S. Safra. Approximating cvp to within almost-polynomial factors is np-hard. *Combinatorica*, 23(2):205–243, 2003. ISSN 0209-9683. doi: http://dx.doi.org/10.1007/s00493-003-0019-y.

[14] S. Even, A. L. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Inf. Control*, 61(2):159–173, 1984. ISSN 0019-9958. doi: http://dx.doi.org/10.1016/S0019-9958(84)80056-X.

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

[16] J. Feigenbaum, L. Fortnow, C. Lund, and D. Spielman. The power of adaptiveness and additional queries in random-self-reductions. *Computational Complexity*, 4:338–346, 1994.

[17] L. Fortnow. The complexity of perfect zero-knowledge. *Advances in Computing Research: Randomness and Computation*, 5:327–343, 1989.

[18] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science*, 134(2):545–557, 1994.

[19] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[20] O. Goldreich. On promise problems (a survey in memory of Shimon Even [1935-2004]). Technical Report TR05–018, Electronic Colloquium on Computational Complexity, February 2005.

[21] O. Goldreich and S. Goldwasser. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540–563, 2000. ISSN 0022-0000. 30th Annual ACM Symposium on Theory of Computing (Dallas, TX, 1998).

[22] O. Goldreich and S. Vadhan. Comparing entropies in statistical zero-knowledge with applications to the structure of SZK. Technical Report TR98-063, Electronic Colloquium on Computational Complexity, 1998.

[23] O. Goldreich and S. P. Vadhan. Comparing entropies in statistical zero knowledge with applications to the structure of SZK. In *IEEE Conference on Computational Complexity*, pages 54–73. IEEE Computer Society, 1999.

[24] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/285055.285060.

[25] I. Haitner, M. Mahmoody-Ghidary, and D. Xiao. A new sampling protocol with applications to basing cryptographic primitives on the hardness of **NP**, 2009. CCC, 2010. Also available as ECCC TR-867-09.

[26] I. Haitner, A. Rosen, and R. Shaltiel. On the (im)possibility of arthur-merlin witness hiding protocols. In *Theory of Cryptography, Fourth Theory of Cryptography Conference, TCC 2009*, 2009.

[27] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. Preliminary versions in *STOC'89* and *STOC'90*.

[28] E. Hemaspaandra, A. V. Naik, M. Ogihara, and A. L. Selman. P-selective sets and reducing search to decision vs. self-reducibility. *J. Comput. Syst. Sci.*, 53(2):194–209, 1996. ISSN 0022-0000. doi: http://dx.doi.org/10.1006/jcss.1996.0061.

[29] T. Holenstein. Private communication. 2009.

[30] R. Impagliazzo and A. Wigderson. $\mathbf{P} = \mathbf{BPP}$ if $\mathbf{E}$ requires exponential circuits: Derandomizing the XOR lemma. In *Proc.* 29*th STOC*, pages 220–229. ACM, 1997.

[31] S. Khot. Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5): 789–808, 2005. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/1089023.1089027.

[32] N. Megiddo and C. H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theor. Comput. Sci.*, 81(2):317–324, 1991. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/0304-3975(91)90200-L.

[33] D. Micciancio. Lattice based cryptography. In H. C. A. van Tilborg, editor, *Encyclopedia of Cryptography and Security*. Springer, 2005. ISBN 978-0-387-23473-1.

[34] D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. In *FOCS '04*, pages 372–381, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2228-9. doi: http://dx.doi.org/10.1109/FOCS.2004.72.

[35] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, Oct. 1994. Preliminary version in FOCS' 88.

[36] R. O'Donnell. Hardness amplification within np. *J. Comput. Syst. Sci.*, 69(1):68–94, 2004. ISSN 0022-0000. doi: http://dx.doi.org/10.1016/j.jcss.2004.01.001.

[37] R. Ostrovsky and A. Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Proceedings of the 2nd Israel Symposium on Theory of Computing Systems*, pages 3–17. IEEE Computer Society, 1993.

[38] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994. ISSN 0022-0000. doi: http://dx.doi.org/10.1016/S0022-0000(05)80063-7.

[39] R. Pass. Parallel repetition of zero-knowledge proofs and the possibility of basing cryptography on np-hardness. In *IEEE Conference on Computational Complexity*, pages 96–110, 2006.

[40] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proc. 41st STOC*, STOC '09, pages 333–342, New York, NY, USA, 2009.

[41] C. Peikert and V. Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In *CRYPTO 2008: Proceedings of the 28th Annual conference on Cryptology*, pages 536–553, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-85173-8. doi: http://dx.doi.org/10.1007/978-3-540-85174-5_30.

[42] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proc. 37th STOC*, pages 84–93, New York, NY, USA, 2005.

[43] A. Sahai and S. Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM*, 50(2):196–249, 2003. Preliminary version in *FOCS'97*.

[44] Trevisan. On uniform amplification of hardness in NP. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2005.

[45] S. Vadhan. Private communication., 2009.

[46] S. P. Vadhan. *A Study of Statistical Zero-Knowledge Proofs*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1999.

[47] A. C. Yao. Protocols for secure computations. pages 160–164, 1982.

A Omitted Proofs

A.1 AM protocol for entropy

Proof of Lemma 3.4. We give a simple proof for completeness. Recall the following facts. For two circuits Y_1 and Y_2 sampling distributions, let their concatenation Y_1Y_2 be the joint circuit sampling the product distribution. For all $\varepsilon > 0$, it is possible given any $y > 0$ to construct in $\text{poly}(n, 1/\varepsilon)$ time a circuit Z_y such that $|H(Z_y) - y| < \varepsilon/100$.

It is known that ED can be reformulated so that the gap between X_1 and X_2 is $\varepsilon/2$ rather than 1, and still the problem remains complete for **SZK**, so let us assume this. Since $\mathsf{ED} \in \mathbf{SZK} \subseteq \mathbf{AM}$ [1, 22], therefore given input (X, y), the verifier can run in parallel the **AM** protocol for ED on inputs (XZ_1, Z_y) and (Z_{y+1}, X) and accept iff both executions accept. □

A.2 Proof sketch of Beigel's Theorem (Theorem 1.3)

Let C_Π be a checker for Π. In order to check Π', given the input x, the checker algorithm $C_{\Pi'}$ first runs the reduction $A'(x)$ and for any query like y that $A'(x)$ wants to ask from its R oracle, $C_{\Pi'}$ does the following. $C_{\Pi'}$ runs the checker algorithm $C_\Pi(y)$ over the "oracle" $A^{\mathcal{O}}$. If it leads to reject, C_Π rejects as well, but if it did not reject and returned the output z, $C_{\Pi'}$ safely uses the answer z for the query y and continues running the reduction $A'(x)$. It is easy to see C_Π's completeness and soundness of follow from those of $C_{\Pi'}$ and the definition of reductions A and A'.

A.3 Proof sketch of $\mathbf{P}^{\mathbf{SZK}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$

Theorem A.1 (Vadhan, Theorem 5.4 in [20])**.** *Let $\Pi = (\Pi_Y, \Pi_N)$ be such that there exist sets S_Y, S_N satisfying $S_Y \cup S_N = \{0,1\}^*$, $\Pi_Y \subseteq S_Y$, $\Pi_N \subseteq S_N$, $(S_Y, \Pi_N) \in \mathbf{NP}$, $(S_N, \Pi_Y) \in \mathbf{NP}$. Then it holds that $\mathbf{P}^\Pi \subseteq \mathbf{NP} \cap \mathbf{co\text{-}NP}$.*

This result can be proven the following way (this differs from the proof given in [20]). Consider problems Π that "extend" to a **TFNP** search problem in the following way: there exist disjoint S_Y, S_N containing Π_Y, Π_N respectively such that the relation

$$(x, bw) \in R \;\Leftrightarrow\; (b = 1 \wedge (x, w) \in S_Y) \vee (b = 0 \wedge (x, w) \in S_N)$$

is in **TFNP**. This is equivalent to the hypothesis of Theorem A.1, and solving R immediately implies solving Π. Now combined with the fact that $\mathbf{P}^{\mathbf{TFNP}} \subseteq \mathbf{NP} \cap \mathbf{co\text{-}NP}$ one obtains Theorem A.1.

This definition naturally generalizes to problems Π that "extend" to **TFAM** problems. As we argued in Proposition 3.5, the **SZK**-complete problem entropy difference can be extended to the **TFAM** problem of computing $f(X_1, X_2) = H(X_1) - H(X_2)$. For the same reason that $\mathbf{P}^{\mathbf{TFNP}} \subseteq \mathbf{NP} \cap \mathbf{co\text{-}NP}$, it also holds that $\mathbf{P}^{\mathbf{TFAM}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$ (Vadhan [45] proved this using a proof along the lines of the proof of Theorem A.1 given in [20]). Therefore, one concludes that

$$\mathbf{P}^{\mathbf{SZK}} = \mathbf{P}^{\mathsf{ED}} \subseteq \mathbf{P}^{\mathbf{TFAM}} \subseteq \mathbf{AM} \cap \mathbf{coAM}$$

A.4 Proof sketch of Observation 4.6

The proof is along the lines of Nisan's proof that Permanent has a two-prover proof system (which is equivalent to a proof system where the prover is stateless and behaves like an oracle).

We show by induction over the input length $k \in [n]$ that R has a δ-tester T_R and an instance checker C_R. By a padding argument we can assume without loss of generality that the worst-case to average-case reduction A always asks queries of the same length as the input length n. Also, if A is the worst-case to average-case reduction, then the following is a checker (as in the proof of Theorem 4.2) on inputs of length n: $C_R^{\mathcal{O}}$ first runs T_R to check that $\mathcal{O}$ $(1-\delta)$-solves R on inputs of length n, and if this passes then run $A^{\mathcal{O}}(x)$ and output whatever $A^{\mathcal{O}}$ outputs.

It suffices therefore to construct a tester T_R. Such a tester trivially exists for input length 1. By induction, we define how T_R behaves on input length n assuming we already have a tester for input lengths $n-1$.

1. First run the tester on $\mathcal{O}$ for input length $n-1$, and reject if it rejects.
2. For $i \in [n/\delta^2]$ sample $x_i \leftarrow D_i$ uniformly.
3. For each sampled x_i, run the downward self-reduction of R over x_i which will (perhaps adaptively) generate queries x_{ij} of length $|x_{i,j}| < |x|$ for $j \le \text{poly}(n)$.
4. Run $A^{\mathcal{O}}$ on each of the queries x_{ij} (which are of length $\le k-1$).
5. Use the resulting answers along with the downward self-reduction in order to compute the answer for x_i. Call this answer y_i.
6. Reject if there exists any x_i such that either of the following holds:
   (a) $\mathcal{O}(x_i) \neq \varnothing$ and $(x_i, \mathcal{O}(x_i)) \notin R$.
   (b) $\mathcal{O}(x_i) = \varnothing$ but $(x_i, y_i) \in R$ and $y_i \neq \varnothing$.

   Otherwise, accept.

The analysis of the tester T_R is similar to that of Theorem 4.2. Let $\mathsf{TIME}_{T_R}(n)$ denote the running time of T_R on inputs of length n. Then it is clear that $\mathsf{TIME}_{T_R}(n) \le \mathsf{TIME}_{T_R}(n-1) + \text{poly}(n)$, which implies that $\mathsf{TIME}_{T_R}(n) \le \text{poly}(n)$. The reason is that we only run the tester once for each input length.