Electronic Colloquium on Computational Complexity, Report No. 25 (2019)

On Nonadaptive Reductions to the Set of Random Strings and Its Dense Subsets

Shuichi Hirahara*
The University of Tokyo

Osamu Watanabe†
Tokyo Institute of Technology

February 28, 2019

Abstract

We investigate the computational power of an arbitrary distinguisher for (not necessarily computable) hitting set generators as well as the set of Kolmogorov-random strings. This work contributes to (at least) two lines of research. One line of research is the study of the limits of black-box reductions to some distributional NP problem. We show that a black-box nonadaptive randomized reduction to any distinguisher for (not only polynomial-time but even) exponential-time computable hitting set generators can be simulated in $\mathsf{AM} \cap \mathsf{coAM}$; we also show an upper bound of $\mathsf{S}_2^{\mathsf{NP}}$ even if there is no computational bound on a hitting set generator. These results further strengthen the evidence that the recent worst-case to average-case reductions within NP shown by Hirahara (2018, FOCS) are inherently non-black-box. As an application, we show that GapMCSP $\in \mathsf{P/poly}$ implies that GapMCSP is low for $\mathsf{S}_2^{\mathrm{p}}$, which is proved by combining our proof techniques with the non-black-box reductions.

Another line of research concerns the computational power of nonadaptive deterministic polynomial-time reductions to the set of Kolmogorov-random strings. It was conjectured by Allender (CiE, 2012) and others that the computational power is exactly characterized by BPP, intuitively because nonadaptive deterministic reductions could only make use of Kolmogorov-random strings as a source of pseudorandomness.

We present strong evidence *against* this conjecture by showing that every language in the exponential-time hierarchy is reducible to the set of Kolmogorov-random strings under PH reductions; in particular, the conjecture is false unless the exponential-time hierarchy collapses to BPEXP. Moreover, our reduction cannot be regarded as a black-box reduction to avoiding hitting set generators (unless the exponential-time hierarchy collapses to the second level), thereby showing that nonadaptive deterministic efficient reductions can exploit the power of Kolmogorov-random strings not just as a distinguisher for hitting set generators.

1 Introduction

Kolmogorov complexity enables us to quantify how a finite string looks "random" in terms of compressibility. For a string $x \in \{0,1\}^*$, its Kolmogorov complexity is the length of the shortest program d such that running d prints x. More specifically, we fix an arbitrary universal Turing machine U, and the Kolmogorov complexity of x is defined as $\mathrm{K}_U(x) := \min\{\, |d| \mid U(d) = x \,\}$. A

*hirahara@is.s.u-tokyo.ac.jp
†watanabe@is.titech.ac.jp

ISSN 1433-8092

string x is called *random* (with threshold s) if $\mathrm{K}_U(x) \geq s$, i.e., x cannot be compressed into a short program.

In this paper, we extensively study what can be reduced to the set of random strings, and its dense subset. As a consequence, we contribute to several lines of research of complexity theory – including average-case complexity and black-box reductions, hitting set generators, the Minimum Circuit Size Problem, and the computational power of the set of random strings. Perhaps most surprisingly, we present new hardness results for the set of Kolmogorov-random strings, and thereby we refute Allender's conjecture [All12] stating that polynomial-time nonadaptive reductions to the set of Kolmogorov-random string can be simulated by BPP in some sense.

The underlying theme that unifies these research lines is Kolmogorov complexity. While Kolmogorov complexity is not computable, by either imposing a time constraint on U or taking another "decoder" U, we are led to several important concepts of complexity theory mentioned above. Below, we review these concepts through the lens of Kolmogorov complexity.

An important motivation for this work is the case when a decoder U is defined as a circuit interpreter G^{int}: Let G^{int} denote the function that takes a description of a Boolean circuit C, and outputs the truth table of the function computed by C. Here a *truth table* of a function $f\colon \{0,1\}^n \to \{0,1\}$ is the string of length 2^n that can be obtained by concatenating $f(x)$ for every input $x \in \{0,1\}^n$, and we often identify a function with its truth table. Taking $U = G^{\mathsf{int}}$, the Kolmogorov complexity $\mathrm{K}_{G^{\mathsf{int}}}(f)$ is approximately equal to the minimum circuit size for computing f. Therefore, a circuit lower bound question can be seen as a question of finding a random string f with respect to $\mathrm{K}_{G^{\mathsf{int}}}$. For example, one of the central open questions in complexity theory, $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{\epsilon n})$ for some constant $\epsilon > 0$, can be equivalently rephrased as the question whether there exists a polynomial-time algorithm that, on input 1^N, finds a "random" string f of length N such that $\mathrm{K}_{G^{\mathsf{int}}}(f) = N^{\Omega(1)}$ for infinitely many N. The problem of computing $\mathrm{K}_{G^{\mathsf{int}}}(f)$ on input f is called the Minimum Circuit Size Problem (MCSP) [KC00], which is intensively studied recently.

A dense subset of random strings (with respect to $\mathrm{K}_{G^{\mathsf{int}}}$) is also one of the important concepts in complexity theory, named as a natural property. In the influential work of Razborov and Rudich [RR97], they introduced the notion of natural proof, and explained the limits of current proof techniques for showing circuit lower bounds. A *natural property* $R \subseteq \{0,1\}^*$ is a polynomial-time computable $1/\mathsf{poly}(\ell)$-dense subset of random strings with respect to $\mathrm{K}_{G^{\mathsf{int}}}$. Here, a set is called γ-*dense* if $\Pr_{x \in_R \{0,1\}^\ell}[x \in R] \geq \gamma(\ell)$ for every $\ell \in \mathbb{N}$. It is known that a natural property is equivalent to an errorless average-case algorithm for MCSP [HS17].

More generally, a dense subset of random strings with respect to K_G can be seen as an adversary for a hitting set generator G. We consider a family of functions $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell \in \mathbb{N}}$. A *hitting set generator* (HSG) is the notion that is used to derandomize one-sided-error randomized algorithms. For a set $R \subseteq \{0,1\}^*$, we say that G is a hitting set generator (with parameter γ) for R if $\Pr_{r \in_R \{0,1\}^\ell}[r \in R] \geq \gamma(\ell)$ implies $R \cap \mathrm{Im}(G_\ell) \neq \varnothing$, for every $\ell \in \mathbb{N}$. Conversely, R is said to γ-*avoid* G if G is not a hitting set generator for R, that is, (1) $\Pr_{r \in_R \{0,1\}^\ell}[r \in R] \geq \gamma(\ell)$ for all $\ell \in \mathbb{N}$ (i.e., R is γ-dense), and (2) $R \cap \mathrm{Im}(G_\ell) = \varnothing$ (i.e., R does not intersect with the image $\mathrm{Im}(G_\ell)$ of G_ℓ). Since $\mathrm{Im}(G_\ell)$ contains all the non-random strings with respect to K_{G_ℓ}, this definition means that R is a γ-dense subset of random strings with respect to K_G.

Now we proceed to reviewing each research line. We start with average-case complexity and black-box reductions.

1.1 Limits of Black-box Reductions

The security of modern cryptography is based on average-case hardness of some computational problems in NP. It is, however, a challenging question to find a problem in NP that is hard with respect to a random input generated efficiently. The fundamental question of average-case complexity is to find a problem in NP whose average-case hardness is based on the worst-case complexity of an NP-complete problem.

A line of work was devoted to understanding why resolving this question is so difficult. Given our poor understanding of unconditional lower bounds, the most prevailing proof technique in complexity theory for showing intractability of a problem is by means of reductions. Moreover, almost all reduction techniques are *black-box* in the sense that, given two computational problems A and B, a reduction R solves A given any oracle (i.e., a black-box algorithm) solving B. The technique of reductions led to the discovery of tons of NP-complete problems computationally equivalent to each other — in the *worst-case* sense. On the other hand, it turned out that the power of black-box reductions is limited for the purpose of showing intractability of average-case problems based on worst-case problems.

Building on the work of Feigenbaum and Fortnow [FF93], Bogdanov and Trevisan [BT06] showed that if a worst-case problem L is reducible to some average-case problem in NP via a nonadaptive black-box randomized polynomial-time reduction, then L must be in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. This in particular shows that the hardness of any average-case problem in NP cannot be based on the worst-case hardness of an NP-complete problem via such a reduction technique (unless the polynomial-time hierarchy collapses [Yap83]). Akavia, Goldreich, Goldwasser and Moshkovitz [AGGM06, AGGM10] showed that, in the special case of a nonadaptive reduction to the task of inverting a one-way function, the upper bound of [BT06] can be improved to $\mathsf{AM} \cap \mathsf{coAM}$, thereby removing the advice "/poly". Bogdanov and Brzuska [BB15] showed that even a general (i.e. adaptive) reduction to the task of inverting a size-verifiable one-way function cannot be used for any problem outside $\mathsf{AM} \cap \mathsf{coAM}$. Applebaum, Barak, and Xiao [ABX08] studied black-box reductions to PAC learning, and observed that the technique of [AGGM06] can be applied to (some restricted type of) a black-box reduction to the task of inverting an auxiliary-input one-way function.

1.2 Non-black-box Reductions

It was very recent that the first worst-case to average-case reductions from worst-case problems conjectured to be outside coNP to some average-case problems in NP were found: Hirahara [Hir18] showed that approximation versions of the minimum time-bounded Kolmogorov complexity problem (MINKT [Ko91]) and MCSP admit worst-case to average-case reductions. These problems ask, given a string x and a threshold s, whether x can be compressed by certain types of algorithms of size s. For example, MCSP asks whether x can be compressed as a truth table of a circuit of size at most s. For a constant $\epsilon > 0$, its approximation version $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is the problem of approximating the minimum circuit size for a function $f\colon \{0,1\}^n \to \{0,1\}$ (represented as its truth table) within a factor of $2^{(1-\epsilon)n}$. Specifically, the YES instances of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ consists of (f, s) such that $\mathsf{size}(f) \leq s$, and the NO instances of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ consists of (f, s) such that $\mathsf{size}(f) > 2^{(1-\epsilon)n}s$. MCSP can be defined as $\mathrm{Gap}_1\mathrm{MCSP}$. It is easy to see that $\mathrm{MCSP} \in \mathsf{NP}$ and $\mathrm{MINKT} \in \mathsf{NP}$, but these are important examples of problems for which there is currently neither a proof of NP-completeness nor evidence against NP-completeness. Allender and Das [AD17] showed that MCSP is SZK-hard, but this hardness result is unlikely to be improved to NP-hardness using "oracle-independent" reduction

techniques: Hirahara and Watanabe [HW16] showed that a one-query randomized polynomial-time reduction to MCSP^A for every oracle A can be simulated in $\mathsf{AM} \cap \mathsf{coAM}$. Nonetheless, MCSP and MINKT are (indirectly) conjectured to be outside $\mathsf{coNP/poly}$ by Rudich [Rud97] based on some assumptions of average-case complexity: He conjectured that there exists a (certain type of) hitting set generator secure even against nondeterministic polynomial-size circuits. We also mention that the approximation version of MINKT is harder than Random 3SAT, which is conjectured by Ryan O'Donnell (cf. [HS17]) to not be solvable by coNP algorithms.

One of the main questions addressed in this paper is whether the technique used in [Hir18] is inherently non-black-box or not. As mentioned above, there are several results and techniques developed in order to simulate black-box reductions by $\mathsf{AM} \cap \mathsf{coAM}$ algorithms. Why can't we combine these techniques with the (seemingly non-black-box) reductions of [Hir18] in order to prove $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \mathsf{coAM}$ and refute Rudich's conjecture? Note that refuting Rudich's conjecture would significantly change our common belief about average-case complexity and the power of nondeterministic algorithms. We emphasize that while the proof of [Hir18] seems to yield only non-black-box reductions, it does not necessarily mean that there is no alternative proof that yields a black-box reduction.

In order to address the question, we aim at improving our understanding of the limits of black-box reductions. We summarize a landscape around average-case complexity in Figure 1.

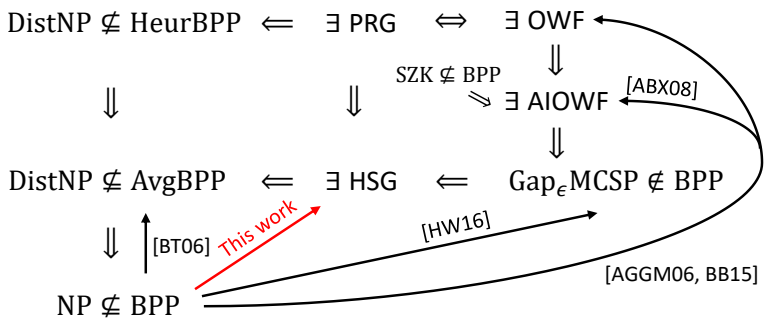

Figure 1: Average-case complexity and limits of black-box reductions. "$A \to B$" means that there is no black-box (or oracle-independent) reduction technique showing "$A \Rightarrow B$" under reasonable complexity theoretic assumptions. The security of all cryptographic primitives is with respect to an almost-everywhere polynomial-time randomized adversary.

A couple of remarks about implications written in Figure 1 are in order: First, the equivalence between the existence of a pseudorandom generator (PRG) and a one-way function (OWF) is due to [HILL99]. Second, the implication from the existence of an auxiliary-input one-way function (AIOWF) to $\mathrm{Gap}_\epsilon\mathrm{MCSP} \notin \mathsf{BPP}$ was implicitly proved in [ABK+06b] and explicitly in [AH17], based on [HILL99, GGM86, RR97]. The implication from $\mathsf{SZK} \not\subseteq \mathsf{BPP}$ to the existence of an auxiliary-input one-way function is due to Ostrovsky [Ost91] (see also [Vad06]). Third, building on [CIKK16, HS17], it was shown in [Hir18, Theorem VI.5] that $\mathrm{Gap}_\epsilon\mathrm{MCSP} \notin \mathsf{BPP}$ implies the nonexistence of natural properties, which yields a hitting set generator $G^{\mathsf{int}} = \{G_{2^n} \colon \{0,1\}^{\widetilde{O}(2^{\epsilon' n})} \to \{0,1\}^{2^n}\}_{n \in \mathbb{N}}$ defined as a "circuit interpreter": a function that takes a description of a circuit of size $2^{\epsilon' n}$ and outputs its truth table (cf. [Hir18, Definition V.3]). The existence of a hitting set generator naturally induces a hard problem in DistNP with respect to AvgBPP algorithms (cf. [Hir18, Lemma VI.4]). Therefore, the reduction of [Hir18] can be regarded as a non-black-box (in

fact, nonadaptive) reduction to a distinguisher for the hitting set generator G^{int}.

We thus continue the study of the limits of black-box reductions to a distinguisher for a hitting set generator, initiated by Gutfreund and Vadhan [GV08]. Motivated by the question on whether derandomization is possible under uniform assumptions (cf. [IW01, TV07]), they investigated what can be reduced to any oracle avoiding a hitting set generator in a black-box way.[1] They showed that any polynomial-time randomized nonadaptive black-box reductions to any oracle avoiding an exponential-time computable hitting set generator G can be simulated in $\mathsf{BPP}^{\mathsf{NP}}$, which is a trivial upper bound when G is polynomial-time computable.

1.3 Our Results

We significantly improve this upper bound to $\mathsf{AM} \cap \mathsf{coAM}$, thereby putting the study of hitting set generators into the landscape of black-box reductions within NP (Figure 1). We also show a uniform upper bound of $\mathsf{S}_2^{\mathsf{NP}}$ even if G is not computable.

Theorem 1 (Main; informal)**.** *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell \in \mathbb{N}}$ be any (not necessarily computable) hitting set generator such that $s(\ell) \leq (1 - \Omega(1))\ell$ for all large $\ell \in \mathbb{N}$. Let $\mathsf{BPP}_{\|}^R$ denote the class of languages solvable by a randomized polynomial-time nonadaptive machine with oracle access to R. (The subscript $\|$ stands for parallel queries.) Then,*

$$\bigcap_R \mathsf{BPP}_{\|}^R \subseteq \mathsf{NP/poly} \cap \mathsf{coNP/poly} \cap \mathsf{S}_2^{\mathsf{NP}},$$

where the intersection is taken over all oracles R that $(1 - 1/\mathsf{poly}(\ell))$-avoid G. Moreover, if G_ℓ is computable in $2^{O(\ell)}$, then we also have

$$\bigcap_R \mathsf{BPP}_{\|}^R \subseteq \mathsf{AM} \cap \mathsf{coAM}.$$

Compared to the line of work showing limits of black-box reductions within NP, a surprising aspect of Theorem 1 is that it generalizes to any function G that may not be computable. Indeed, almost all the previous results [FF93, BT06, AGGM06, ABX08] crucially exploit the fact that a verifier can check the correctness of a certificate for an NP problem; thus a dishonest prover can cheat the verifier only for one direction, by not providing a certificate for a YES instance. In our situation, a verifier cannot compute G and thus cannot prevent dishonest provers from cheating in this way. At a high level, our technical contributions are to overcome this difficulty by combining the ideas of Gutfreund and Vadhan [GV08] with the techniques developed in [FF93, BT06].

Moreover, we present a new $\mathsf{S}_2^{\mathrm{p}}$-type algorithm for simulating reductions to an oracle R avoiding G. Indeed, at the core of Theorem 1 is the following two types of algorithms simulating reductions: One is an $\mathsf{S}_2^{\mathrm{p}}$ algorithm that simulates any query $q \stackrel{?}{\in} R$ of length at most $\Theta(\log n)$, and the other is an $\mathsf{AM} \cap \mathsf{coAM}$ algorithm that simulates any query $q \stackrel{?}{\in} R$ of length at least $\Theta(\log n)$. In particular, when G is exponential-time computable, the $\mathsf{S}_2^{\mathrm{p}}$ algorithm can be replaced with a polynomial-time algorithm and obtain the $\mathsf{AM} \cap \mathsf{coAM}$ upper bound.

[1] As a *black-box* reduction to any distinguisher for G, it is required in [GV08] that there exists a *single* machine that computes a reduction to every oracle avoiding G. On the other hand, as stated in Theorem 1, we allow reductions to depend on oracles, which makes our results stronger.

We remark that Theorem 1 improves all the previous results mentioned before in some sense. Compared to [BT06], our results show that the advice "/poly" is not required in order to simulate black-box reductions to any oracle avoiding an exponential-time computable hitting set generator. Compared to [AGGM06, ABX08], our results "conceptually" improve their results because the existence of one-way functions imply the existence of hitting set generators; on the other hand, since the implication goes through the *adaptive* reduction (from the task of inverting a one-way function to a distinguisher for a PRG) of [HILL99], technically speaking, our results are incomparable with their results.[2] Similarly, our results conceptually improve the result of [HW16], but these are technically incomparable, mainly because the implication goes through the non-black-box reduction of [Hir18].

1.4 Why are the Reductions of [Hir18] Non-black-box?

Based on Theorem 1, we now argue that the reductions of [Hir18] are inherently non-black-box in a certain formal sense, without relying on any unproven assumptions: The reason is that the idea of [Hir18] can be applied to not only time-bounded Kolmogorov complexity but also any other types of Kolmogorov complexity, including resource-unbounded Kolmogorov complexity. Therefore, if this generalized reduction could be made black-box, then (as outlined below) by Theorem 1 we would obtain a finite algorithm $\mathsf{S}_2^{\mathsf{NP}}$ that approximates resource-unbounded Kolmogorov complexity, which is a contradiction, *unconditionally*.

To give one specific example, we briefly outline how the reductions of [Hir18] can be generalized to the case of Levin's Kt-complexity [Lev84]: Fix any efficient universal Turing machine U, and the Kt-complexity of a string x is defined as

$$\mathrm{Kt}(x) := \min\{|d| + \log t \mid U(d) \text{ outputs } x \text{ within } t \text{ steps } \}.$$

We define a hitting set generator $G = \{G_\ell : \{0,1\}^{\ell/2} \to \{0,1\}^\ell\}_{\ell\in\mathbb{N}}$ as $G_\ell(d,t) := U(d)$ for $(d,t) \in \{0,1\}^{\ell/2}$ when $|U(d)| = \ell$ and $U(d)$ halts within t steps, which is computable in exponential time. Note that $\mathrm{Im}(G)$ contains all strings with low Kt-complexity. Given an efficient algorithm D that γ-avoids G, we can approximate $\mathrm{Kt}(x)$ by the following algorithm: Fix any input x. Take any list-decodable code Enc, and let $\mathrm{NW}^{\mathrm{Enc}(x)}(z)$ denote the Nisan-Wigderson generator [NW94] instantiated with $\mathrm{Enc}(x)$ as the truth table of a hard function, where z is a seed of the generator. Then check whether the distinguishing probability $|\mathbb{E}_{z,w}[D(\mathrm{NW}^{\mathrm{Enc}(x)}(z)) - D(w)]|$ is large or small by sampling, whose outcome tells us whether $\mathrm{Kt}(x)$ is small or large, respectively. Indeed, if the distinguishing probability is large, then by using the security proof of the Nisan-Wigderson generator, we obtain a short description (with oracle access to D) for x. Conversely, if $\mathrm{Kt}(x)$ is small, then since D γ-avoids G, the distinguishing probability is at least γ. Now, if we could make this analysis work for any oracle that γ-avoids G, then by Theorem 1 we would put a problem of approximating $\mathrm{Kt}(x)$ in AM, which is not possible unless $\mathsf{EXP} = \mathsf{PH}$. (Note that the minimization problem of Kt is EXP-complete under NP reductions [ABK+06b].)

[2] We emphasize that we concern the nonadaptivity of reductions used in the security proof of pseudorandom generators. Several simplified constructions of pseudorandom generators G^f from one-way functions f (e.g., [Hol06, HRV13]) are nonadaptive in the sense that G^f can be efficiently computed with nonadaptive oracle access to f; however, the security reductions of these constructions are adaptive because of the use of Holenstein's uniform hardcore lemma [Hol05]. Similarly, the reduction of [HILL99, Lemma 6.5] is adaptive. (We note that, in the special case when the degeneracy of a one-way function is efficiently computable, the reduction of [HILL99] is nonadaptive.)

1.5 Applications: Lowness of GapMCSP

As explained in the previous subsection, the non-black-box reductions of [Hir18] cannot be combined with Theorem 1 *unconditionally*. In contrast, we show that the proof ideas of Theorem 1 can be combined with the non-black-box reductions *conditionally*, which leads us to a new structural property of GapMCSP. Specifically, assuming GapMCSP is easy with respect to non-uniform algorithms, we show that GapMCSP is low for $\mathsf{S}_2^{\mathrm{p}}$.

Theorem 2. *For any oracle A, any constant $\alpha > 0$, if $\mathrm{Gap}_\alpha\mathrm{MCSP}^A \in \mathsf{P/poly}$ then $\mathsf{S}_2^{\mathrm{Gap}_\beta\mathrm{MCSP}^A} \subseteq \mathsf{S}_2^{\mathrm{p}}$ for some constant $\beta > 0$.*

The main idea of Theorem 2 is to combine the $\mathsf{S}_2^{\mathrm{p}}$-type algorithm for simulating reductions to an oracle avoiding G^{int} (Theorem 1) with the non-black-box reductions. Recall that $\mathsf{S}_2^{\mathrm{p}}$ is a proof system where two competing provers, one of which is guaranteed to be honest, try to convince a polynomial-time verifier. In our $\mathsf{S}_2^{\mathrm{p}}$ simulation algorithm, for each $i \in \{0,1\}$, the ith prover sends as a polynomial-size circuit an oracle R_i avoiding G^{int}. Then a verifier obtains an oracle $R_0 \cap R_1$ that avoids G^{int} (assuming that either R_0 or R_1 avoids G^{int}). We then invoke the non-black-box reductions of [CIKK16, Hir18] to solve $\mathrm{Gap}_\beta\mathrm{MCSP}^A$ using $R_0 \cap R_1$ as an oracle.

Unfortunately, we were not able to combine our $\mathsf{AM} \cap \mathsf{coAM}$ algorithm of Theorem 1 with the non-black-box reductions under similar conditions. We leave it as an interesting open question.

Open Question 3. Prove that $\mathsf{NP} \subseteq \mathsf{P/poly}$ (or $\mathrm{MCSP} \in \mathsf{P/poly}$) implies $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \mathsf{coAM}$ for some constant $\epsilon > 0$.

1.6 Our Techniques

We outline our proof strategy for Theorem 1 below. Suppose that we have some reduction from L to any oracle R that avoids a hitting set generator G. Let $\mathcal{Q}$ denote the query distribution that a reduction makes. We focus on the case when the length of each query is larger than $\Theta(\log n)$, and explain the ideas of the $\mathsf{AM} \cap \mathsf{coAM}$ simulation algorithms.

As a warm-up, consider the case when the support $\mathsf{supp}(\mathcal{Q})$ of $\mathcal{Q}$ is small (i.e., $|\mathsf{supp}(\mathcal{Q}) \cap \{0,1\}^\ell| \ll 2^\ell$ for any length $\ell \in \mathbb{N}$). In this case, we can define an oracle R_1 so that $R_1 := \{0,1\}^* \setminus \mathsf{supp}(\mathcal{Q}) \setminus \mathrm{Im}(G)$; this is a dense subset and avoids the hitting set generator G. Therefore, we can simulate the reduction by simply answering all the queries by saying "No"; hence such a reduction can be simulated in BPP.

In general, we cannot hope that $\mathsf{supp}(\mathcal{Q})$ is small enough. To generalize the observation above, let us recall the notion of α-heaviness [BT06]: We say that a query q is *α-heavy* (with respect to $\mathcal{Q}$) if the query q is α times more likely to be sampled under $\mathcal{Q}$ than the uniform distribution on $\{0,1\}^{|q|}$; that is, $\Pr_{w\sim\mathcal{Q}}[w = q] \geq \alpha 2^{-|q|}$. Now we define our new oracle $R_2 := \{0,1\}^* \setminus \{\, q \in \{0,1\}^* \mid q\colon \alpha\text{-heavy} \,\} \setminus \mathrm{Im}(G)$, which can be again shown to avoid G because the fraction of α-heavy queries is at most $1/\alpha$ $(\ll 1)$.

The problem now is that it is difficult to simulate the new oracle R_2; it appears that, given a query q, we need to test whether $q \stackrel{?}{\in} \mathrm{Im}(G)$, which is not possible in $\mathsf{AM} \cap \mathsf{coAM}$. However, it turns out that we do not need to test it, as we explain next: Observe that the size of $\mathrm{Im}(G)$ is very small; it is at most $2^{s(\ell)}$ $\left(\ll 2^\ell\right)$. Thus, the probability that a query q is in $\mathrm{Im}(G)$ and q is not α-heavy (i.e., q is rarely queried) is at most $\alpha \cdot 2^{s(\ell)-\ell}$, where ℓ is the length of q. As a consequence, the reduction cannot "distinguish" the oracle R_2 and a new oracle $R_3 := \{0,1\}^* \setminus \{\, q \in \{0,1\}^* \mid q\colon \alpha\text{-heavy} \,\}$;

hence we can simulate the reduction if, given a query q, we are able to decide whether $q \stackrel{?}{\in} R_3$ in $\mathsf{AM} \cap \mathsf{coAM}$.

This task, however, still appears to be difficult for $\mathsf{AM} \cap \mathsf{coAM}$; indeed, at this point, Gutfreund and Vadhan [GV08] used the fact that the approximate counting is possible in $\mathsf{BPP}^{\mathsf{NP}}$, and thereby simulated the oracle R_3 by $\mathsf{BPP}^{\mathsf{NP}}$.

Our main technical contribution is to develop a way of simulating the reduction to R_3. First, note that the lower bound protocol of Goldwasser and Sipser [GS86] enables us to give an AM certificate for α-heaviness; we can check, given a query q, whether q is $\alpha(1+\epsilon)$-heavy or α-light for any small error parameter $\epsilon > 0$. Thus, we have an AM protocol for $\{0,1\}^* \setminus R_3$ for every query q (except for $\alpha(1 \pm \epsilon)$-heavy and light queries).

If, in addition, we had an AM protocol for R_3, then we would be done; unfortunately, it does not seem possible in general. The upper bound protocol of Fortnow [For89] does a similar task, but the protocol can be applied only for a limited purpose: we need to keep the randomness used to generate a query $q \sim \mathcal{Q}$ from being revealed to the prover. When the number of queries of the reduction is limited to 1, we may use the upper bound protocol in order to give an AM certificate for R_3; on the other hand, if the reduction makes two queries $(q_1, q_2) \sim \mathcal{Q}$, we cannot simultaneously provide AM certificates of the upper bound protocol for *both* of q_1 and q_2, because the fact that q_1 and q_2 are sampled *together* may reveal some information about the private randomness. To summarize, the upper bound protocol works only for the *marginal* distribution of each query, but does not work for the *joint* distribution of several queries.

That is, what we can obtain by using the upper bound protocol is information about *each* query. For example, the heavy-sample protocol of Bogdanov and Trevisan [BT06] (which combines the lower and upper bound protocol and sampling) estimates, in $\mathsf{AM} \cap \mathsf{coAM}$, the probability that a query q sampled from $\mathcal{Q}$ is α-heavy.

Our idea is to overcome the difficulty above by generalizing the Feigenbaum-Fortnow protocol [FF93]. Feigenbaum and Fortnow developed an $\mathsf{AM} \cap \mathsf{coAM}$ protocol that simulates a nonadaptive reduction to an NP oracle R, given as advice the probability that a query is a positive instance of R. We generalize the protocol in the case when the oracle $\{0,1\}^* \setminus R_3$ is solvable by AM on average (which can be done by the lower bound protocol [GS86]), and given as advice the probability that a query q is in $\{0,1\}^* \setminus R_3$ (which can be estimated by the heavy-sample protocol [BT06]):

Theorem 4 (Generalized Feigenbaum-Fortnow Protocol; informal)**.** *Suppose that M is a randomized polynomial-time nonadaptive reduction to oracle R whose queries are distributed according to $\mathcal{Q}$, and that R is solvable by AM on average (that is, there exists an AM protocol Π_R such that, with probability $1 - 1/\mathsf{poly}(n)$ over the choice of $q \sim \mathcal{Q}$, the protocol Π_R computes R on input q). Then, there exists an $\mathsf{AM} \cap \mathsf{coAM}$ protocol Π_M such that, given a probability $p^* \approx \Pr_{q \sim \mathcal{Q}}[q \in R]$ as advice, the protocol Π_M simulates the reduction M with probability at least $1 - 1/\mathsf{poly}(n)$.*

On the Case of Adaptive Reductions. We mention that Theorem 1 cannot be extended to the case of *adaptive* reductions. Indeed, Trevisan and Vadhan [TV07] constructed an exponential-time computable pseudorandom generator based on the intractability of some PSPACE-complete problem, and its security reduction is black-box in the sense of Theorem 1 and adaptive. If Theorem 1 could be extended to the case of adaptive reductions, we would obtain $\mathsf{PSPACE} = \mathsf{AM}$, which is unlikely to be true.

1.7 Lower Bounds

Theorem 1 leads us to the natural question whether the upper bound is tight. We present evidence that our two types of simulation algorithms are nearly tight.

1.7.1 On the $\mathsf{AM} \cap \mathsf{coAM}$-type Simulation Algorithms

In Appendix A, we observe that the SZK-hardness of MCSP [AD17] also holds for an average-case version of MCSP:

Theorem 5. *Let $\epsilon > 0$ be any constant, and R be any oracle $\frac{1}{2}$-avoiding $G^{\mathsf{int}} = \{G_n^{\mathsf{int}} : \{0,1\}^{n^\epsilon} \to \{0,1\}^n\}_{n\in\mathbb{N}}$. Then, $\mathsf{SZK} \subseteq \mathsf{BPP}^R$.*

The reduction of Theorem 5 is adaptive because of the use of [HILL99]. We conjecture that $\mathsf{SZK} \subseteq \bigcap_R \mathsf{BPP}_{\|}^R$, which implies that the $\mathsf{AM} \cap \mathsf{coAM}$ upper bound of Theorem 1 cannot be significantly improved. While we were not able to obtain nonadaptive reductions in the general case, we show in Appendix A that the problem of factoring the product of two primes is nonadaptively reducible to R.

Theorem 6. *Let $\epsilon > 0$ be any constant, and R be any oracle $\frac{1}{2}$-avoiding $G^{\mathsf{int}} = \{G_n^{\mathsf{int}} : \{0,1\}^{n^\epsilon} \to \{0,1\}^n\}_{n\in\mathbb{N}}$. Then, factoring the product of two primes can be done in $\mathsf{ZPP}_{\|}^R$.*

1.7.2 On the $\mathsf{S}_2^{\mathrm{p}}$-type Simulation Algorithms

Our $\mathsf{S}_2^{\mathrm{p}}$-type simulation algorithm is in fact completely tight in a certain setting. Let G be a universal Turing machine. We consider an exponential-time analogue of Theorem 1 when the reduction can make only short queries. Specifically, for an oracle R, denote by $\mathsf{EXP}^{R^{\leq \mathsf{poly}}}$ the class of languages that can be computed by a $2^{n^{O(1)}}$-time algorithm that can query $q \overset{?}{\in} R$ of length $\leq n^{O(1)}$, on inputs of length n. (We note that all the queries of polynomial length can be asked by an exponential-time reduction, and thus the adaptivity does not matter here.) We show that the computational power of $\mathsf{EXP}^{R^{\leq \mathsf{poly}}}$ where R is an arbitrary dense subset of Kolmogorov-random strings (i.e., R avoids G) is *exactly* equal to the exponential-time analogue of $\mathsf{S}_2^{\mathrm{p}}$.

Theorem 7 (informal)**.** *Fix any universal Turing machine U. Then we have*

$$\bigcap_{R\colon \frac{1}{2}\text{-avoids } U} \mathsf{EXP}^{R^{\leq \mathsf{poly}}} = \bigcap_{R\colon \frac{1}{2}\text{-avoids } U} \mathsf{BPEXP}^{R^{\leq \mathsf{poly}}} = \mathsf{S}_2^{\mathrm{exp}}.$$

Here $R^{\leq \mathsf{poly}}$ means that the length of queries is restricted to be at most a polynomial in the input length. We also have $\mathsf{EXP}^{\mathsf{NP}} \subseteq \bigcap_R \mathsf{S}_2^R \subseteq \mathsf{S}_2^{\mathrm{exp}}$.

Previously, Allender, Friedman and Gasarch [AFG13] showed that black-box BPP reductions to any avoiding oracle can be simulated in $\mathsf{EXPSPACE}$. Theorem 7 significantly improves their upper bound to $\mathsf{S}_2^{\mathrm{exp}}$.

Despite the fact that the set of Kolmogorov-random strings is not computable, Theorem 7 gives a surprising connection between efficient reductions to any dense subset of Kolmogorov-random strings and a complexity class. Such a characterization was sought after in yet another line of work, as we review below.

1.8 Reduction to the Set of Kolmogorov-Random Strings

There is a clear relationship between Kolmogorov-randomness and pseudorandomness, as explored by Allender, Buhrman, Koucký, van Melkebeek, and Ronneburger [ABK+06b]: Any string generated by a computable process has low Kolmogorov complexity, and thus the set of Kolmogorov-random strings serves as a distinguisher for any computable hitting set generator; in particular, they proved a curious inclusion that $\mathsf{PSPACE} \subseteq \mathsf{P}^{R_{\mathrm{K}_U}}$. Here, $R_{\mathrm{K}_U} := \{\, x \in \{0,1\}^* \mid K_U(x) \geq |x|/2 \,\}$ is the set of random strings with respect to K_U. Similarly, Allender, Buhrman, Koucký [ABK06a] showed that $\mathsf{NEXP} \subseteq \mathsf{NP}^{R_{\mathrm{K}_U}}$; Buhrman, Fortnow, Koucký, and Loff [BFKL10] showed that $\mathsf{BPP} \subseteq \mathsf{P}_{\parallel}^{R_{\mathrm{K}_U}}$.

Note that since the set of Kolmogorov-random strings is not computable, it is absurd to hope that efficiently computable complexity classes could be characterized in terms of efficient reductions to R_{K_U}. Surprisingly, it was shown by Allender, Friedman, and Gasarch [AFG13] (and later improved by Cai, Downey, Epstein, Lempp, and Miller [CDE+14]) that when the intersection is taken over all prefix-free universal Turing machines U, there is a computable upper bound:

Theorem 8 ([BFKL10, AFG13, CDE+14])**.** $\mathsf{BPP} \subseteq \bigcap_U \mathsf{P}_{\parallel}^{R_{\mathrm{K}_U}} \subseteq \mathsf{PSPACE}$*, where the intersection is taken over all prefix-free universal Turing machines.*

For some technical reasons, the upper bounds of [AFG13, CDE+14] are known to hold only for *prefix-free* universal Turing machines. However, a similar upper bound can be obtained for usual universal Turing machines by imposing a super-constant minimum query length restriction on reductions [HK18].

At this point, a natural question arises: What is the exact computational power of R_{K_U} under polynomial-time nonadaptive reductions? Intuitively, any polynomial-time nonadaptive reduction cannot make any use of the set of Kolmogorov-random strings of length larger than $O(\log n)$, because the Kolmogorov complexity of any query that the reduction can make on input 1^n is at most $O(\log n)$. It was argued in [ABFL14] that, intuitively, short queries to Kolmogorov-random strings could only be used as a source of pseudorandomness. Allender [All12] thus explicitly conjectured that the lower bound of Theorem 8 is exactly tight, and then a fair amount of effort has been made to verify the conjecture.

Conjecture 9 ([BFKL10, All12, ADF+13, ABFL14, HK18])**.** $\mathsf{BPP} = \bigcap_U \mathsf{P}_{\parallel}^{R_{\mathrm{K}_U}}$.

Beyond its curiosity, such a characterization might enable us to study BPP by using the techniques about Kolmogorov complexity. Moreover, Conjecture 9 is interesting from the point of view of the study of MCSP: In some technical sense, Kolmogorov complexity can be seen as the minimum size of a circuit with oracle access to the halting problem (cf. [ABK+06b]); thus R_{K_U} can be regarded as a computability-theoretic analogue of MCSP. In light of this, since Conjecture 9 shows non-NP-hardness results about R_{K_U} under nonadaptive polynomial-time reductions (unless $\mathsf{NP} \subseteq \mathsf{BPP}$), it would give us some new insights about NP-hardness of MCSP, which has been a focus of recent work on MCSP (e.g. [MW17, HW16, HP15, AHK17, AH17]). We refer the reader to the survey of Allender [All17] for detailed background on MCSP and R_{K_U}.

1.9 Evidence against Conjecture 9

In this work, we disprove Conjecture 9 under the assumption that the exponential-time hierarchy does not collapse to BPEXP. Our main new insight is to consider a sparse language in $\bigcap_U \mathsf{P}_{\parallel}^{R_{\mathrm{K}_U}}$,

or in other words, an exponential time analogue of Theorem 8; we obtain the following (the upper bound is from [AFG13, CDE+14]).

Theorem 10. $\mathsf{EXPH} \subseteq \bigcap_U \mathsf{PH}^{R_{\mathrm{K}_U}} \subseteq \mathsf{EXPSPACE}$.

In particular, by a simple padding argument, it implies that $\bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}} \neq \mathsf{BPP}$ unless $\mathsf{EXPH} = \mathsf{BPEXP}$. We emphasize that Theorem 10 crucially makes use of the fact that the set of Kolmogorov-random strings serves not just as a distinguisher for hitting set generators; indeed, otherwise, by Theorem 7, we would obtain the unexpected collapse of EXPH to $\mathsf{S}_2^{\mathrm{exp}}$. In particular, Theorem 10 shows that the intuition that nonadaptive polynomial-time reductions could exploit R_{K_U} only as a source of pseudorandomness is wrong.

We mention that by translating Theorem 10 into a polynomial-time analogue, we obtain that any language in PH-uniform P/poly is computable by a *quasipolynomial-time* algorithm with oracle access to R_{K_U}. We leave it as an open question to improve the running time of this reduction to polynomial time.

Organization. The rest of this paper is organized as follows. After reviewing necessary background in Section 2, we show that a reduction only with short queries can be simulated by $\mathsf{S}_2^{\mathrm{p}}$ in Section 3. We present the generalized Feigenbaum-Fortnow protocol in Section 4; then we show a proof of Theorem 1 in Section 5. In Section 6, we show the lowness property of GapMCSP. We investigate the computational power of the set of Kolmogorov-random strings in Section 7.

2 Preliminaries

By a *string* we mean a binary string, i.e., an element of $\{0,1\}^*$, and by a *language* we mean a set of strings. We identify a language $L \subseteq \{0,1\}^*$ with its characteristic function. For a language A and $\ell \in \mathbb{N}$, let $A^{=\ell} := A \cap \{0,1\}^\ell$. Let $[n] := \{1,\ldots,n\}$ for $n \in \mathbb{N}$.

Interactive Proof Systems. AM is the class of languages L that can be accepted by some polynomial-time two-round Arthur-Merlin protocol; that is, there exists a polynomial-time Turing machine V (called an AM verifier) such that

- (Completeness) if $x \in L$ then $\Pr_r[V(x,y,r) = 1 \text{ for some } y] \geq \frac{2}{3}$, and
- (Soundness) if $x \notin L$ then $\Pr_r[V(x,y,r) = 1 \text{ for some } y] \leq \frac{1}{3}$.

For our purpose, it is convenient to use the following characterization of $\mathsf{AM} \cap \mathsf{coAM}$.

Fact 11. *Let $L \subseteq \{0,1\}^*$. Then $L \in \mathsf{AM} \cap \mathsf{coAM}$ if and only if there exists a randomized polynomial-time verifier V of a private-coin constant-round interactive proof system such that, for any input $x \in \{0,1\}^*$,*

- *(Completeness) there exists a prover P such that V outputs $L(x)$ by communicating with P with probability at least $\frac{2}{3}$, and*
- *(Soundness) for any prover P, V outputs $L(x)$ or $\perp$ with P with probability at least $\frac{2}{3}$.*

That is, the verifier outputs a correct answer $L(x)$ when interacting with an honest prover, and it does not output the wrong answer $1 - L(x)$ for any cheating prover, with high probability. The fact

above follows from the transformation from public coin protocols to private coin protocols [GS86], and the AM hierarchy collapses [Bab85].

Circuits. We will use a circuit in order to make it easy to compose several protocols[3]. Usually, a circuit takes a string of some fixed length as input and outputs a string of some fixed length. For our purpose, it is convenient to extend this usual notion of circuit: By using some encoding, we regard circuits as taking a string of length *at most* l for some $l \in \mathbb{N}$, and outputting a string (not necessarily of fixed length). We regard a circuit as computing a function from $\{0,1\}^*$ to $\{0,1\}^* \cup \{\text{"undefined"}\}$ such that the function outputs "undefined" on inputs of length $> l$.

Nonadaptive Reductions. A polynomial-time *nonadaptive* reduction is a polynomial-time oracle Turing machine whose possible queries can be computed without access to an oracle in polynomial time. For simplicity, we assume without loss of generality that, for all inputs of the same length, the reduction makes the same number m of queries (by adding dummy queries if necessary). For any oracle $R \subseteq \{0,1\}^*$, we denote by $\mathsf{BPP}_{\|}^R$ the class of languages from which there exists a randomized polynomial-time nonadaptive reduction. For a nonadaptive reduction M, we denote by $M^R(x)$ the output of the reduction given an oracle $R \subseteq \{0,1\}^*$ and input $x \in \{0,1\}^*$.

Query Distribution. We can modify a randomized nonadaptive reduction M so that the marginal distribution of each query of M is identical; that is, for any query $q \in \{0,1\}^*$, the probability that q is sampled as the ith query of M is the same for all $i \in [m]$. To achieve this, we simply modify M as follows: it generates a permutation $\pi\colon [m] \to [m]$ uniformly at random, runs $M(x)$ to make m queries $q_1, \ldots, q_m$, asks $q_{\pi(i)}$ as the ith query to get an answer a_i from an oracle, and resumes the computation of $M(x)$ to get the decision on x by supplying $a_{\pi^{-1}(1)}, \ldots, a_{\pi^{-1}(m)}$ as oracle answers. It is then easy to see that in the new query machine the ith query distribution is identical for all $i \in [m]$. By the modification above, we can take a single query distribution $\mathcal{Q}_x$ such that each query of $M(x)$ is distributed according to $\mathcal{Q}_x$.

Let $\mathcal{Q}$ be a distribution over $\{0,1\}^*$. We use the notation $q \sim \mathcal{Q}$ to indicate that a random variable q is sampled from $\mathcal{Q}$. For a set A, we use the notation $a \in_R A$ to indicate that a random variable a is sampled from the uniform distribution over A. For a string $x \in \{0,1\}^*$, let $\mathcal{Q}(x)$ denote $\Pr_{q\sim\mathcal{Q}}[q = x]$. For any $\alpha > 0$, a string $q \in \{0,1\}^*$ of length $\ell \in \mathbb{N}$ is called *α-heavy* (with respect to $\mathcal{Q}$) if $\mathcal{Q}(q) \geq \alpha 2^{-\ell}$; otherwise (i.e., $\mathcal{Q}(q) < \alpha 2^{-\ell}$), it is called *$\alpha$-light*.

Concentration Inequality. We will use a standard concentration inequality:

Lemma 12 (Hoeffding's inequality [Hoe63])**.** *For any independent random variables $X_1, \ldots, X_n \in [0,1]$ and any $t \geq 0$, we have* $\Pr\left[\left|\sum_{i=1}^n (X_i - \mathbb{E}[X_i])\right| \geq nt\right] \leq 2\exp(-2nt^2)$.

Description Interpreter (Hitting Set Generator). Throughout this paper, we use ℓ to denote a size parameter for discussing oracles that avoid simple strings. We consider any family of functions $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell\in\mathbb{N}}$ that is regarded as a generator of strings from their shorter descriptions; here, $s\colon \mathbb{N} \to \mathbb{N}$ denotes a function that determines the seed length of the generator. We assume that $s(\ell) \leq \ell$ and, for simplicity, that $\ell - s(\ell)$ is nondecreasing. We call such family G of functions a *description interpreter*. We measure the time complexity of G with respect to its output length; that is, we say that G is $O(t_G(\ell))$-time computable if there exists a deterministic algorithm that computes, for any given description $u \in \{0,1\}^{s(\ell)}$ and $\ell \in \mathbb{N}$, a length ℓ string

[3]The reader may simply regard a circuit as a Turing machine with an appropriate description to which one can embed some additional information.

$w = G_\ell(u)$ in time $O(t_G(\ell))$. Let $\mathrm{Im}(G_\ell)$ denote the range of G_ℓ; that is, $\mathrm{Im}(G_\ell) = \{\, w \in \{0,1\}^\ell \mid w = G_\ell(u) \text{ for some } u \in \{0,1\}^{s(\ell)} \,\}$. Also let $\mathrm{Im}(G) := \cup_{\ell\in\mathbb{N}}\mathrm{Im}(G_\ell)$. We regard a description interpreter G as a hitting set generator, and define the standard notion of breaking G as follows.

Definition 13. *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell\in\mathbb{N}}$ be any description interpreter. For any parameter $\gamma\colon \mathbb{N} \to [0,1)$, a set R of strings is called a* γ-avoiding *set for G (or, R γ-avoids G) if*

1. *$R \cap \mathrm{Im}(G) = \varnothing$, and*
2. *$\Pr_{w\in R\{0,1\}^\ell}[\, w \in R\,] \geq \gamma(\ell)$ for all $\ell \in \mathbb{N}$.*

Similarly, for any fixed length $\ell \in \mathbb{N}$, we say that R is a γ-avoiding set at length ℓ for G *if $R^{=\ell} \cap \mathrm{Im}(G) = \varnothing$ and $\Pr_{w\in R\{0,1\}^\ell}[w \in R] \geq \gamma(\ell)$.*

For simplicity, we always assume that a parameter γ satisfies $\gamma(\ell) \leq 1 - 2^{s(\ell)-\ell}$ for all $\ell \in \mathbb{N}$, as otherwise γ-avoiding sets may not exist. A set R satisfying the second condition above is called γ-*dense*. Thus, a γ-*avoiding* set for G is simply a subset of $\{0,1\}^* \setminus \mathrm{Im}(G)$ that is γ-dense. In the context where G is fixed, we omit specifying G and simply say that R is a γ-avoiding set.

Definition 14 (Black-box reduction to γ-avoiding oracles [GV08])**.** *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell\in\mathbb{N}}$ be any description interpreter. Let $L \subseteq \{0,1\}^*$ be a language. A randomized nonadaptive oracle machine M is called a* black-box reduction *from L to any γ-avoiding oracle of G if, for any γ-avoiding oracle R for G and any $x \in \{0,1\}^*$, we have*

$$\Pr\left[\, M^R(x) = L(x) \,\right] \;\geq\; \frac{2}{3}, \tag{1}$$

where the probability is taken over the internal randomness of M.

Remark. By the standard amplification technique, the success probability $\frac{2}{3}$ in Definition 14 can be boosted to $1 - 2^{-|x|^c}$ for any constant c.

We stress that the definition above requires that there exists a *single* machine that works for *every* γ-avoiding oracle; on the other hand, Theorem 1 states that, if, for every γ-avoiding oracle R, there exists a nonadaptive reduction M_R from L to R, then $L \in \mathsf{AM} \cap \mathsf{coAM}$. That is, the order of the quantifier is reversed; nonetheless, a diagonalization argument enables us to establish the equivalence:

Proposition 15. *Let G be any description interpreter, $\gamma\colon \mathbb{N} \to [0,1)$ be any parameter, and $L \subseteq \{0,1\}^*$ be a language. The following are equivalent:*

1. $L \in \bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{BPP}_{\|}^R$.
2. *There exists a randomized polynomial-time nonadaptive black-box reduction from L to any γ-avoiding oracle of G.*

Proof. The direction from the second item to the first item is obvious. We prove below the contrapositive of the other direction.

Suppose that, for any randomized nonadaptive oracle machine M, there exists some γ-avoiding oracle R_M of G such that $\Pr_M\left[M^R(x) = L(x)\right] < \frac{2}{3}$ for some $x \in \{0,1\}^*$. We claim that there exists some *single* γ-avoiding oracle R of G such that $L \notin \mathsf{BPP}_{\|}^R$.

To this end, let $\{M_e\}_{e\in\mathbb{N}}$ be the set of all randomized nonadaptive oracle machines. We will construct some γ-avoiding oracle R_e and input x_e (and $\ell_e \in \mathbb{N}$) by induction on $e \in \mathbb{N}$, so that M_e given oracle R_{e+1} fails to compute L on input x_e; then we will define $R := \bigcup_{e\in\mathbb{N}} R_e$. Let us start with $R_0 := \varnothing$ and $\ell_0 := 0$.

At stage $e \in \mathbb{N}$, we claim that there exists some γ-avoiding oracle $R'_{e+1} \subseteq \{0,1\}^*$ and some input $x_e \in \{0,1\}^*$ such that

- $\Pr\left[M_e^{R'_{e+1}}(x_e) = L(x_e)\right] < \frac{2}{3}$, and
- $q \in R_e$ if and only if $q \in R'_{e+1}$ for any string q of length $< \ell_e$.

Indeed, for any oracle Q, let $Q' := \{\, q \in Q \mid |q| \geq \ell_e \,\} \cup \{\, q \in R_e \mid |q| < \ell_e \,\}$. Consider a randomized nonadaptive oracle machine M'_e such that $M_e'^{Q}$ simulates $M_e^{Q'}$; that is, M'_e is hardwired with the set $\{\, q \in R_e \mid |q| < \ell_e \,\}$, and simulates M_e and answer any query q of length $< \ell_e$ by using the hardwired information. By our assumption, there exists some γ-avoiding oracle $\hat{R}_{e+1}$ of G such that $\Pr\left[M_e'^{\hat{R}_{e+1}}(x_e) = L(x_e)\right] < \frac{2}{3}$ for some $x_e \in \{0,1\}^*$; by the definition of M'_e, we obtain $\Pr\left[M_e^{R'_{e+1}}(x_e) = L(x_e)\right] < \frac{2}{3}$ for $R'_{e+1} := \{\, q \in \hat{R}_{e+1} \mid |q| \geq \ell_e \,\} \cup \{\, q \in R_e \mid |q| < \ell_e \,\}$, which is again γ-avoiding G. This completes the proof of the claim above. Now define $\ell_{e+1} \in \mathbb{N}$ as a large enough integer so that $\ell_{e+1} \geq \ell_e$ and the machine M_e on input x_e does not query any string of length $\geq \ell_{e+1}$, and define an oracle $R_{e+1} := \{\, q \in R'_{e+1} \mid |q| < \ell_{e+1} \,\}$, which completes the construction of stage $e \in \mathbb{N}$.

Define $R := \bigcup_{e\in\mathbb{N}} R_e$, which γ-avoids G by the construction above. By the choice of $(\ell_e)_{e\in\mathbb{N}}$, we have

$$\Pr\left[M_e^R(x_e) = L(x_e)\right] = \Pr\left[M_e^{R_{e+1}}(x_e) = L(x_e)\right] < \frac{2}{3},$$

for every randomized nonadaptive oracle machine M_e. Thus $L \notin \mathsf{BPP}_{\parallel}^R$. □

3 Simulating Short Queries by Competitive Prover Systems

We first show that a reduction that makes only short queries can be simulated by $\mathsf{S}_2^{\mathrm{p}}$.

Theorem 16 ($\mathsf{S}_2^{\mathrm{p}}$ Simulation of Short Queries)**.** *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell\in\mathbb{N}}$ be any description interpreter and $\gamma\colon \mathbb{N} \to [0,1)$ be a parameter such that $\gamma(\ell) \leq 1 - 2^{s(\ell)-\ell+1}$ for all large $\ell \in \mathbb{N}$. Suppose that there exists a randomized polynomial-time black-box reduction M from a language L to any γ-avoiding oracle for G such that the length of any query of M is at most $O(\log n)$ for every input of length n. Then $L \in \mathsf{S}_2^{\mathrm{p}}$.*

Proof. The idea is that two competitive provers send the image $\mathrm{Im}(G)$ of G as a certificate. Given two possible images $I_0, I_1 \subseteq \{0,1\}^*$, $R := \{0,1\}^* \setminus I_0 \setminus I_1$ is an avoiding set for G. Moreover, since $|\mathrm{Im}(G)|$ is small, the set R is dense enough. We then derandomize a BPP computation by using the power of $\mathsf{S}_2^{\mathrm{p}}$.[4] (Recall that $\mathsf{BPP} \subseteq \mathsf{S}_2^{\mathrm{p}}$ [Can96, RS98].) Details follow.

Let $c \log n$ be an upper bound on the length of queries that M makes. Our $\mathsf{S}_2^{\mathrm{p}}$ algorithm is as follows: Fix any input x of length n. We number the two competitive provers 0 and 1. The ith prover

[4] Alternatively, we may use the result of Russell and Sundaram [RS98] showing that $\mathsf{S}_2 \cdot \mathsf{BP} \cdot \mathsf{P} = \mathsf{S}_2^{\mathrm{p}}$ in a black-box way.

($i \in \{0,1\}$) sends, for each $\ell \leq c \log n$, a subset $I_{i,\ell} \subseteq \{0,1\}^\ell$ of size at most $2^{s(\ell)}$; an honest prover sets $I_{i,\ell} := \mathrm{Im}(G_\ell)$. Define $I_i := \bigcup_{\ell \leq c \log n} I_{i,\ell}$. Note that such subsets can be encoded as a string of polynomial length. Each prover also sends a list of randomness $r_1^i, \cdots, r_t^i \in \{0,1\}^m$ to be used by the reduction M, where t is a parameter chosen later, and m is the length of a coin flip used by M. The verifier sets $R := \{0,1\}^* \setminus I_0 \setminus I_1$, and accept if and only if $\Pr_{j,k \in_R [t]}[M^R(x; r_j^0 \oplus r_k^1) = 1] > \frac{1}{2}$, where $M^R(x;r)$ denotes the output of the reduction when its coin flip is r, and $\oplus$ denotes the bit-wise XOR. Note that the running time of the verifier is at most a polynomial in n and t. Below we establish the correctness of this algorithm for some $t = \mathsf{poly}(n)$.

We focus on the case when the 0th prover is honest; thus $I_{0,\ell} := \mathrm{Im}(G_\ell)$ for each $\ell \leq c \log n$. Since $|I_{1,\ell}| \leq 2^{s(\ell)}$, the number of strings of length ℓ in $R(I_1) := \{0,1\}^* \setminus I_0 \setminus I_1$ is at least $2^\ell - |I_{0,\ell}| - |I_{1,\ell}| \geq 2^\ell \gamma(\ell)$ (here we write $R(I_1)$ instead of R to emphasize that R depends on I_1); thus $R(I_1)$ is a γ-avoiding oracle. By the definition of the reduction, for every I_1 we have $\Pr_{r \in_R \{0,1\}^m}[M^{R(I_1)}(x;r) = L(x)] \geq \frac{2}{3}$.

We use the notion of cover introduced by Canetti [Can96]: A sequence $r_1, \cdots, r_t \in \{0,1\}^m$ is called a cover of a subset $A \subseteq \{0,1\}^m$ if for all $r \in \{0,1\}^m$, $\Pr_{j \in_R [t]}[r_j \oplus r \in A] > \frac{1}{2}$. Define $A(I_1) := \{\, r \in \{0,1\}^m \mid M^{R(I_1)}(x;r) = L(x) \,\}$. We claim that by a probabilistic argument there exists a sequence $r_1, \cdots, r_t \in \{0,1\}^m$ that covers $A(I_1)$ for every I_1: Fix any I_1 and $r \in \{0,1\}^m$. Pick $r_1, \cdots, r_t \in_R \{0,1\}^m$. For any $j \in [t]$, the probability that $r_j \oplus r \in A(I_1)$ is at least $\frac{2}{3}$. Thus by a concentration bound (Lemma 12), the probability that at most a $\frac{1}{2}$-fraction of $j \in [t]$ satisfies $r_j \oplus r \in A(I_1)$ is at most $\exp(-\Omega(t))$. By the union bound over all r, the probability that a sequence $r_1, \cdots r_t$ does not cover $A(I_1)$ is at most $2^m \cdot \exp(-\Omega(t))$. By the union bound over all I_1, the probability that there exists some I_1 such that $A(I_1)$ is not covered by $r_1, \cdots, r_t$ is at most $2^{n^c + m} \cdot \exp(-\Omega(t))$. Therefore, for $t := \Theta(n^c + m)$, there exists a sequence $r_1, \cdots, r_t$ that covers $A(I_1)$ for every I_1. The 0th honest prover sends this sequence $r_1, \cdots, r_t$ to the verifier as $r_1^0, \cdots, r_t^0$, in which case the verifier outputs $L(x)$ correctly because $\Pr_{j,k \in_R [t]}[M^{R(I_1)}(x; r_j^0 \oplus r_k^1) = L(x)] > \frac{1}{2}$, for every I_1 and every $r_1^1, \cdots, r_t^1$. □

By a simple padding argument, Theorem 16 gives us the upper bound claimed in Theorem 7. We defer a proof of the matching lower bound to Section 7.

Theorem 17. *Take any description interpreter G and any function γ satisfying the hypothesis of Theorem 16. Then,*

$$\bigcap_{R:\ \gamma\text{-avoids } G} \mathsf{BPEXP}^{R^{\leq \mathsf{poly}}} \quad \subseteq \quad \mathsf{S}_2^{\mathrm{exp}}.$$

Here $R^{\leq \mathsf{poly}}$ means that the length of queries is restricted to be at most a polynomial in an input length.

Proof. Let $L \in \bigcap_{R:\ \gamma\text{-avoids } G} \mathsf{BPEXP}^{R^{\leq \mathsf{poly}}}$. We first note that, as in Proposition 15, the order of quantifiers can be swapped; indeed, the proof of Proposition 15 does not rely on any specific property of $\mathsf{BPP}_\parallel$ reductions; hence, the same proof works for other notions of reduction. Thus, there exists some randomized $t(n)$-time black-box reduction M from a language L to any γ-avoiding oracle for G such that the length of any query that M makes on input of length n is at most $\log t(n)$, for some $t(n) = 2^{n^{O(1)}}$. Let $L' := \{\, x01^{t(n)} \mid x \in L \,\}$ be a padded version of L. Applying Theorem 16 to L', we obtain $L' \in \mathsf{S}_2^{\mathrm{p}}$, from which it follows that $L \in \mathsf{S}_2^{\mathrm{exp}}$. □

4 Generalized Feigenbaum-Fortnow Protocol

In this section, we present one of the main building blocks of our proof. Our protocol is inspired by the protocol of Feigenbaum and Fortnow [FF93] (and its description by Bogdanov and Trevisan [BT06]) for simulating some type of randomized nonadaptive reduction M to an NP problem R. Suppose that for a given input x, M makes m nonadaptive queries $q_1, \ldots, q_m$ under a certain distribution $\mathcal{Q}$. In the Feigenbaum-Fortnow protocol, a verifier asks a prover to give witnesses to all positive instances among them. The prover cannot give a witness to a negative instance (hence, it cannot cheat the verifier by saying "yes" to a negative instance) while it may try to cheat the verifier by not giving a witness to some of the positive instances of $q_1, \ldots, q_m$. If, however, the verifier knows the proportion p^* of positive instances among queries under the distribution $\mathcal{Q}$, then it may detect wrong negative answers from the prover if the number of positive answers is much smaller than p^*m. More specifically, the Feigenbaum-Fortnow protocol runs as follows. It first generates K tuples of m nonadaptive queries $\{(q_{k1}, \ldots, q_{km})\}_{1 \le k \le K}$ by running $M(x)$ independently K times. By a concentration inequality, the number of positive instances among all Km queries should be in the range of $m \cdot \left(p^*K \pm O(\sqrt{K})\right)$ with high probability; thus, if the prover gives "yes" answers (with witnesses) much smaller than $m \cdot \left(p^*K - O(\sqrt{K})\right)$, then the verifier stops the computation immediately, suspecting that the prover is not honest. On the other hand, if the number of positive answers to those queries is close to p^*Km, then the number of positive instances on which the prover can cheat is at most $O(m\sqrt{K})$, with high probability. We choose K large enough so that $O(m\sqrt{K}) \ll K$; then the majority of K tuples are answered correctly by the oracle, and we can use them to determine the result of $M^R(x)$ by taking the majority vote of the results of $M(x)$ computed by using prover's answers to each tuple of queries $(q_{k1}, \ldots, q_{km})$.[5]

We generalize the Feigenbaum-Fortnow protocol so that a new protocol is capable of dealing with a reduction to a *distributional* AM problem R; that is, we show that, given any nonadaptive reduction to some AM problem solvable on average and the proportion p^* of positive instances as advice, one can simulate the reduction in $\mathsf{AM} \cap \mathsf{coAM}$. In our protocol, we use Adleman's trick (for proving $\mathsf{BPP} \subseteq \mathsf{P/poly}$ [Adl78]) to "derandomize" AM oracle so that we obtain a new NP oracle, and then run the original Feigenbaum-Fortnow protocol. The following is the specification of the generalized Feigenbaum-Fortnow protocol:

Inputs. A tuple (C, V, δ, p^*) such that:

- A randomized nonadaptive reduction C is given as a probabilistic circuit such that each query of C is identically distributed to some distribution $\mathcal{Q}$ over $\{0,1\}^*$, and the reduction always makes exactly m queries. (We assume that an input to a reduction is hardwired into the circuit C; thus C does not take any input other than random bits.)
- An AM verifier V is given as a circuit.
- An error parameter $\delta \in \left(0, \frac{1}{2}\right)$ is given in unary, and a probability $p^* \in [0,1]$ is given in binary.

Promise. We assume that there exist some answer $a \in \{0,1\}$, some oracle $R \subseteq \{0,1\}^*$, and some error parameters $\epsilon_0, \epsilon_1, \epsilon_2 \in [0,1]$ satisfying the following:

[5] We note that taking the majority is not necessary; instead, it suffices to pick $k \in_R [K]$ and use the result.

- $\Pr_C[C^R = a] \geq 1 - \epsilon_0$. (That is, a is supposed to be the answer of the reduction C to the oracle R.)
- The advice p^* satisfies $|p^* - \Pr_{q\sim\mathcal{Q}}[q \in R]| \leq \epsilon_1$.
- The distributional problem $(R, \mathcal{Q})$ is "solvable by AM on average": that is, define[6]

$$V_{\text{YES}} := \{\, q \in \{0,1\}^* \mid \Pr_r [V(q, y, r) = 1 \text{ for some } y] \geq 3/4 \,\}, \text{ and}$$
$$V_{\text{NO}} := \{\, q \in \{0,1\}^* \mid \Pr_r [V(q, y, r) = 0 \text{ for all } y] \geq 3/4 \,\};$$

  then we assume that $\Pr_{q\sim\mathcal{Q}}[q \in V_{\text{YES}} \cup V_{\text{NO}}] \geq 1 - \epsilon_2$ and $V_{\text{YES}} \subseteq R \subseteq \{0,1\}^* \setminus V_{\text{NO}}$.

Protocol.

1. (Preprocess of Verifier: Adleman's trick) Let s be sufficiently large so that $s > 20|V|$ and $s \geq 20\log(1/\delta)$ where $|V|$ denotes the circuit size of V. Pick $r_1, \ldots r_s$ uniformly at random, and share the random bits with the prover. Define a new circuit W by

$$W(x, y_1, \ldots, y_s) := \operatorname*{majority}_{i\in[s]} V(x, y_i, r_i).$$

   In what follows, we call $\bar{y} := (y_1, \ldots, y_s)$ a certificate for W.

2. (Verifier) Let $K := m^2(1/\delta)^2 \log(m/\delta)$. Run C independently K times and obtain queries $(q_{k1}, \ldots, q_{km})$ for each kth run of C ($k \in [K]$). Send these queries to the prover.

3. (Prover) For each $(k, i) \in [K] \times [m]$, send a certificate $\bar{y}_{ki}$ for W; an honest prover sends, if any, some certificate $\bar{y}_{ki}$ such that $W(q_{ki}, \bar{y}_{ki}) = 1$.

4. (Verifier) Let $a^*_{ki} := W(q_{ki}, \bar{y}_{ki}) \in \{0, 1\}$ for each $(k, i) \in [K] \times [m]$. Verify that

$$\sum_{\substack{1\leq k\leq K \\ 1\leq i\leq m}} a^*_{ki} \;\geq\; mp^*K \;-\; m\left((\epsilon_1 + \epsilon_2)K + \sqrt{K\log(m/\delta)}\right), \tag{2}$$

   and if not, output $\perp$ and halt. Otherwise, pick $k \in_R [K]$ uniformly at random and output the kth run of the reduction of C assuming that the answers from the oracle are $(a_{k1}, \ldots, a_{km})$.

Theorem 18 (Correctness of the Generalized Feigenbaum-Fortnow Protocol)**.** *Suppose that the protocol above is given inputs satisfying the promise listed above. Then, the protocol satisfies the completeness and soundness for error $\epsilon := \epsilon_0 + 2m\epsilon_1 + 3m\epsilon_2 + 3\delta$, described below:*

- *(Completeness) There exists a prover such that the verifier outputs a with probability at least $1 - \epsilon$.*
- *(Soundness) For any prover, the verifier outputs a or $\perp$ with probability at least $1 - \epsilon$.*

We prove this theorem by a sequence of claims below. The following claim follows from a standard fact about amplification of the success probability of a randomized machine.

[6] As a circuit V outputs "undefined" if an input (q, y, r) is too long, the sets $V_{\text{YES}}, V_{\text{NO}}$ of strings are finite.

Claim 19 (Amplification and Adleman's trick)**.** *With probability at least* $1-\delta$ *over the choice of* $r_1, \ldots, r_s$*, the following holds. For any query* $q \in V_{\mathrm{YES}} \cup V_{\mathrm{NO}}$*,*

- *if* $q \in V_{\mathrm{YES}}$ *then* $W(q, \bar{y}) = 1$ *for some certificate* $\bar{y} := (y_1, \ldots, y_s)$*, and*
- *if* $q \in V_{\mathrm{NO}}$ *then* $W(q, \bar{y}) = 0$ *for any certificate* $\bar{y} := (y_1, \ldots, y_s)$*.*

Proof. First, note that $|V_{\mathrm{YES}} \cup V_{\mathrm{NO}}|$ is at most $2^{s/20}$. Indeed, the circuit size of V is less than $s/20$, and hence for any input q of length $\geq s/20$, V is not defined; thus $q \notin V_{\mathrm{YES}} \cup V_{\mathrm{NO}}$. That is, the length of every query in $V_{\mathrm{YES}} \cup V_{\mathrm{NO}}$ is less than $s/20$.

Fix any q such that $q \in V_{\mathrm{YES}}$ or $q \in V_{\mathrm{NO}}$; in the former case, let $a_q := 1$ and $a_q := 0$ otherwise. Our claim is that $\max_{\bar{y}} W(q, \bar{y}) = a_q$. For each $i \in [s]$, let $X_i \in \{0,1\}$ be the random variable (over the random choice of $r_1, \ldots, r_s$) such that $X_i := 1$ iff $V(q, y_i, r_i) = 1$ for some y_i; in other words, $X_i := \max_{y_i} V(q, y_i, r_i) \in \{0,1\}$. Observe that

$$\max_{\bar{y}} W(q, \bar{y}) = \max_{\bar{y}} \operatorname*{majority}_{i \in [s]} V(q, y_i, r_i) = \operatorname*{majority}_{i \in [s]} \max_{y_i} V(q, y_i, r_i) = \operatorname*{majority}_{i \in [s]} X_i.$$

By the assumption on V, we have $|\mathbb{E}[X_i] - a_q| \leq \frac{1}{4}$ for any $i \in [s]$; hence, $\operatorname{majority}_{i \in [s]} X_i \neq a_q$ implies $|\frac{1}{s}\sum_i (X_i - \mathbb{E}[X_i])| \geq \frac{1}{4}$. By Hoeffding's inequality (Lemma 12),

$$\Pr[\max_{\bar{y}} W(q, \bar{y}) \neq a_q] \leq \Pr\left[\left|\sum_{i=1}^{s} (X_i - \mathbb{E}[X_i])\right| \geq \frac{s}{4}\right] \leq 2\exp\left(-2s\,(1/4)^2\right) \leq 2^{-s/10}.$$

Now, by the union bound over all $q \in V_{\mathrm{YES}} \cup V_{\mathrm{NO}}$, the probability that there exists some $q \in V_{\mathrm{YES}} \cup V_{\mathrm{NO}}$ such that $\max_{\bar{y}} W(q, \bar{y}) \neq a_q$ is at most $|V_{\mathrm{YES}} \cup V_{\mathrm{NO}}| \cdot 2^{-s/10} \leq 2^{-s/20} \leq \delta$. □

For each $(k, i) \in [K] \times [m]$, define $a_{ki} \in \{0,1\}$ as $a_{ki} := 1$ if and only if $W(q_{ki}, \bar{y}) = 1$ for some certificate $\bar{y}$. The honest prover sends a certificate for W (if any), and thus $a_{ki} = a^*_{ki}$; on the other hand, when communicating with a cheating prover, we have only $a^*_{ki} \leq a_{ki}$. The next claim shows the sum of (a_{ki}) concentrates around its mean.

Claim 20 (Concentration)**.** *Under the event of Claim 19, with probability at least* $1-\delta$*, the following holds:*

$$\left|\sum_{\substack{1 \leq k \leq K \\ 1 \leq i \leq m}} a_{ki} - mKp^*\right| \leq m\left((\epsilon_1 + \epsilon_2)K + \sqrt{K \log(m/\delta)}\right).$$

Proof. Fix any $i \in [m]$. By the assumption on C, the queries $q_{1i}, \ldots, q_{Ki}$ are independent and identically distributed according to $\mathcal{Q}$; hence, $a_{1i}, \ldots, a_{Ki} \in \{0,1\}$ are also independent random variables. The expectation $\mathbb{E}[a_{ki}]$ of these random variables is equal to $\Pr_{q \sim \mathcal{Q}}[W(q, \bar{y}) = 1 \text{ for some } \bar{y}]$.

We claim that, for any $(k, i) \in [K] \times [m]$, the expectation $\mathbb{E}[a_{ki}]$ is equal to p^* up to additive error $\epsilon_1 + \epsilon_2$. Under the event of Claim 19, we have $q \in V_{\mathrm{YES}} \implies \exists \bar{y}.\ W(q, \bar{y}) = 1 \implies q \notin V_{\mathrm{NO}}$ for any $q \in \{0,1\}^*$; hence,

$$\Pr[q \in V_{\mathrm{YES}}] \leq \Pr[\exists \bar{y}.\ W(q, \bar{y}) = 1] \leq \Pr[q \notin V_{\mathrm{NO}}]. \tag{3}$$

Similarly, since $V_{\text{YES}} \subseteq R \subseteq \{0,1\}^* \setminus V_{\text{NO}}$, we have

$$\Pr[q \in V_{\text{YES}}] \leq \Pr[q \in R] \leq \Pr[q \notin V_{\text{NO}}]. \tag{4}$$

Combining (3) and (4), we obtain

$$|\Pr[q \in R] - \mathbb{E}[a_{ki}]| \leq \Pr[q \notin V_{\text{NO}}] - \Pr[q \in V_{\text{YES}}] \leq \epsilon_2.$$

Therefore, $|\mathbb{E}[a_{ki}] - p^*| \leq |\mathbb{E}[a_{ki}] - \Pr[q \in R]| + |\Pr[q \in R] - p^*| \leq \epsilon_2 + \epsilon_1$.

By Hoeffding's inequality (Lemma 12), for each $i \in [m]$,

$$\Pr\left[\left|\sum_{k=1}^{K}(a_{ki} - \mathbb{E}[a_{ki}])\right| \geq \sqrt{K\log(m/\delta)}\right] \leq 2\exp(-2K\log(m/\delta)/K) \leq \delta/m.$$

By the union bound over all $i \in [m]$, with probability at least $1-\delta$, we have

$$\begin{aligned}\left|\sum_{i=1}^{m}\sum_{k=1}^{K} a_{ki} - mKp^*\right| &\leq \left|\sum_{i=1}^{m}\sum_{k=1}^{K}(a_{ki} - \mathbb{E}[a_{ki}])\right| + \left|\sum_{i=1}^{m}\sum_{k=1}^{K}(\mathbb{E}[a_{ki}] - p^*)\right| \\ &\leq \sum_{i=1}^{m}\left|\sum_{k=1}^{K}(a_{ki} - \mathbb{E}[a_{ki}])\right| + mK(\epsilon_1+\epsilon_2) \\ &\leq m\sqrt{K\log(m/\delta)} + mK(\epsilon_1+\epsilon_2).\end{aligned}$$

□

Now we are ready to bound the probability of completeness and soundness. Let E denote any event (which is supposed to be the event that completeness or soundness does not hold); using Claim 19 and 20, we will bound the probability in the following way:

$$\Pr[E] \;\leq\; 2\delta + \Pr[\,E \wedge (\text{the event of Claim 19 holds}) \wedge (\text{the concentration of Claim 20 occurs})\,].$$

That is, assuming that the events of Claim 19 and 20 happens, we will analyze the probability of completeness and soundness.

Claim 21 (Completeness)**.** *The verifier outputs a with probability at least $1-\epsilon$ when interacting with the honest prover.*

Proof. The verifier *does not* output a only if

- the inequality (2) does not hold, or
- for a random $k \in_R [K]$, the kth run of C is not correct.

The honest prover sets $a^*_{ki} := a_{ki}$ for any $(k,i) \in [K] \times [m]$. Thus, under the assumption that the concentration of Claim 20, the inequality (2) is satisfied; that is, the verifier does not output $\perp$, and hence it remains to bound the probability that, for a random $k \in_R [K]$, the kth run of C is not correct.

The kth run of C is not correct only if the reduction itself makes a mistake, or there exists some $i \in [m]$ such that $a^*_{ki} \neq R(q_{ki})$ (which happens only if $q_{ki} \notin V_{\text{YES}} \cup V_{\text{NO}}$ for the honest prover). The former probability is at most ϵ_0, and the latter is at most $m\epsilon_2$.

Overall, the verifier outputs a with probability at least $1 - 2\delta - \epsilon_0 - m\epsilon_2 \geq 1 - \epsilon$. □

Claim 22 (Soundness)**.** *For any cheating prover, the verifier outputs a or $\perp$ with probability at least $1-\epsilon$.*

Proof. The verifier outputs the wrong answer $1-a$ only if for a random $k \in_R [K]$, the kth run of C is not correct.

Recall that we have $a^*_{ki} \le a_{ki}$ for any $(k,i) \in [K] \times [m]$ no matter how a prover tries to cheat. The main difference between the proof of Claim 21 is that, for a random choice $k \in_R [K]$ of the verifier, a prover may be cheating so that $a^*_{ki} < a_{ki}$ for some $i \in [m]$; as a consequence, the kth run of C is more likely to be wrong. On the other hand, the number of $(k,i) \in [K] \times [m]$ such that $a^*_{ki} < a_{ki}$ is small: Indeed, under the event that the verifier does not output $\perp$, the inequality (2) holds, and we also have the concentration of Claim 20; hence, we obtain $\sum_{k=1}^{K} \sum_{i=1}^{m} (a_{ki} - a^*_{ki}) \le 2m\left((\epsilon_1+\epsilon_2)K + \sqrt{K \log(m/\delta)}\right)$. Thus, the probability that $a^*_{ki} < a_{ki}$ for some $i \in [m]$ over the random choice of $k \in_R [K]$ is at most $2m(\epsilon_1+\epsilon_2) + m\sqrt{\log(m/\delta)/K} \le 2m(\epsilon_1+\epsilon_2) + \delta$.

Overall, the probability that the verifier outputs the wrong answer is at most $(2\delta + \epsilon_0 + m\epsilon_2) + (2m(\epsilon_1+\epsilon_2)+\delta) \le \epsilon$. □

Proof of Theorem 18. Immediate from Claim 21 and 22. □

Remark. The generalized Feigenbaum-Fortnow protocol above also works for any reduction that does not necessarily output a Boolean value (e.g., a reduction solving a search problem), with a suitable modification on the completeness and soundness.

5 Simulating Long Queries by $\mathsf{AM} \cap \mathsf{coAM}$

Using the generalized Feigenbaum-Fortnow protocol, we prove our main result:

Theorem 23 (Main; the formal version of Theorem 1)**.** *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell \in \mathbb{N}}$ be any (not necessarily computable) description interpreter and $\gamma \colon \mathbb{N} \to [0,1)$ be a parameter such that*

- *there exists a constant $\epsilon > 0$ such that $s(\ell) \le (1-\epsilon)\ell$ for all large $\ell \in \mathbb{N}$, and*
- *there exists a constant $c > 0$ such that $\gamma(\ell) \le 1 - \ell^{-c}$ for all large $\ell \in \mathbb{N}$.*

Then,

$$\bigcap_{R\colon \gamma\text{-}avoids\ G} \mathsf{BPP}^R_{\|} \quad \subseteq \quad \mathsf{NP/poly} \cap \mathsf{coNP/poly} \cap \mathsf{S}_2^{\mathsf{NP}}.$$

Moreover, if G can be computed in $2^{O(\ell)}$ time, then we also have

$$\bigcap_{R\colon \gamma\text{-}avoids\ G} \mathsf{BPP}^R_{\|} \quad \subseteq \quad \mathsf{AM} \cap \mathsf{coAM}.$$

As shown in Section 3, reductions that make only short queries can be simulated in $\mathsf{S}_2^{\mathrm{p}}$. On the other hand, in this section, we show that reductions that make only long queries can be simulated in $\mathsf{AM} \cap \mathsf{coAM}$. The advice in Theorem 23 is used in order to give the characteristic function of $\mathrm{Im}(G)$ for all strings of length $O(\log n)$, under which situation the rest of reductions can be simulated in

$\mathsf{AM} \cap \mathsf{coAM} \subseteq \mathsf{NP/poly} \cap \mathsf{coNP/poly}$. If a hitting set generator is computable in exponential time, then the advice can be computed in polynomial time and thus can be removed. Without any advice and without any computational bound on a hitting set generator, the reduction can be simulated in $\mathsf{S}_2^{\mathsf{NP}}$, which is a complexity class that can simulate $\mathsf{AM} \cap \mathsf{coAM}$ and $\mathsf{S}_2^{\mathrm{p}}$.

Therefore, the main ingredient of the main theorem is to simulate long queries in $\mathsf{AM} \cap \mathsf{coAM}$:

Theorem 24 (AM Simulation of Long Queries)**.** *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell \in \mathbb{N}}$ be any description interpreter. Let $t, \theta, \alpha_0 \colon \mathbb{N} \to \mathbb{N}$ be efficiently computable functions. Suppose that there exists a randomized $t(n)$-time nonadaptive black-box reduction M from a language L to any γ-avoiding oracle for G such that the length of any query that M makes on input length n is at least $\theta(n)$. Suppose also that, for all large $n \in \mathbb{N}$,*

- $\alpha_0(n) \leq \frac{1}{16\mathrm{e}^3 t(n)^2} 2^{\theta(n) - s(\theta(n))}$, *and*
- $\alpha_0(n) \cdot \left(1 - 2^{s(\ell)-\ell} - \gamma(\ell)\right) \geq 1$ *for all $\ell \in \mathbb{N}$ such that $\theta(n) \leq \ell \leq t(n)$.*

Then, there exists an $\mathsf{AM} \cap \mathsf{coAM}$ protocol running in $t(n)^{O(1)}$ time that decides L.

We first observe that this is sufficient to prove Theorem 23.

Proof of Theorem 23 from Theorem 24. Take any language $L \in \bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{BPP}_{\|}^R$. By Proposition 15, we have a randomized $t(n)$-time nonadaptive black-box reduction M from L to any γ-avoiding oracle for G, where $t(n) = n^{O(1)}$. By the assumption on the seed length s, we have $\epsilon\ell \leq \ell - s(\ell)$ for all large $\ell \in \mathbb{N}$. For any $\ell \in \mathbb{N}$ between $\theta(n)$ and $t(n)$, we have $1 - 2^{s(\ell)-\ell} - \gamma(\ell) \geq \ell^{-c} - 2^{-\epsilon\ell} \gg \ell^{-c}/2$; hence, by defining $\alpha_0(n) := 2t(n)^c$, we obtain $\alpha_0(n) \geq \left(1 - 2^{s(\ell)-\ell} - \gamma(\ell)\right)^{-1}$ for all ℓ between $\theta(n) \leq \ell \leq t(n)$. On the other hand, $2^{\theta(n)-s(\theta(n))}/t(n)^2 \geq 2^{\epsilon\theta(n)}/t(n)^2$; thus, for $\theta(n) := ((c+2+1)\log t(n))/\epsilon$, we obtain $\alpha_0(n) \ll 2^{\theta(n)-s(\theta(n))}/16\mathrm{e}^3 t(n)^2$ for all large n.

The assumptions about parameters of Theorem 24 are thus satisfied for $\theta(n) = O(\log t(n))$. In particular, we can encode the characteristic function of the set $\cup_{\ell \leq \theta(n)} \mathrm{Im}(G_\ell)$ as an advice string of length $t(n)^{O(1)}$. Given such an advice, we can modify the reduction M so that M does not make any query q of length at most $\theta(n)$: Indeed, if the original reduction makes a query q of length $\leq \theta(n)$, then we modify the reduction so that q is answered according to whether $q \in \{0,1\}^* \setminus \mathrm{Im}(G_{|q|})$, which can be decided by using the advice. After this modification, by using Theorem 24, M can be simulated in $\mathsf{AM} \cap \mathsf{coAM}$. We thus obtain $L \in \mathsf{AM/poly} \cap \mathsf{coAM/poly} = \mathsf{NP/poly} \cap \mathsf{coNP/poly}$.

Moreover, if G is computable in $2^{O(\ell)}$, then the advice can be computed in polynomial time: Indeed, by an exhaustive search, one can compute $\mathrm{Im}(G_\ell)$ in time $2^{O(s(\ell))} \cdot 2^{O(\ell)} = 2^{O(\ell)} \leq t(n)^{O(1)}$ for all $\ell \leq \theta(n)$.

Finally, we sketch an $\mathsf{S}_2^{\mathsf{NP}}$ algorithm for deciding L when G is not necessarily computable and no advice is given: As in Theorem 16, each competitive prover sends a verifier all the strings in $\mathrm{Im}(G)$ of length at most $\theta(n)$. Let $I_0, I_1 \subseteq \{0,1\}^*$ be the claimed image of G. Then we modify the reduction M so that any short query is answered according to $\{0,1\}^* \setminus I_0 \setminus I_1$. Now by applying Theorem 24, we obtain an AM algorithm deciding L. In particular, there exists an NP machine V such that $\Pr_r[V(x,r) = L(x)] \geq \frac{2}{3}$ for every input x. This randomized computation can be derandomized as in Theorem 16, by requesting each competitive prover to send a sequence of coin flips $r_1, \cdots, r_s$. Thus we obtain $L \in \mathsf{S}_2^{\mathsf{NP}}$. □

In the rest of this section, we show how to simulate long queries by a constant-round interactive proof system (i.e., a proof of Theorem 24). For simplicity, we focus on the case when $t(n) = n^{O(1)}$. Let M be a randomized $t(n)$-time nonadaptive black-box reduction to any γ-avoiding oracle for G. Let $x \in \{0,1\}^*$ be an input of length n.

We first modify the reduction so that we can assume useful properties. By the modifications explained in Section 2, we may assume that the number of queries that M makes is exactly $m(n)$ $(\leq t(n))$ on inputs of length n. We may also assume that each query of M is identically distributed; Let $\mathcal{Q}_x$ be the query distribution of M on input x.

As explained in the introduction, one of the keys of our proof is that we can replace a γ-avoiding oracle for G by an oracle defined based only on the query distribution $\mathcal{Q}_x$. Here we introduce such oracles and justify the replacement. For any $\alpha > 0$, define L_α, H_α and R_α by

$$\begin{aligned} L_\alpha &:= \{\, q \in \{0,1\}^* \mid q \text{ is } \alpha\text{-light with respect to } \mathcal{Q}_x \,\}, \\ H_\alpha &:= \{0,1\}^* \setminus L_\alpha \;=\; \{\, q \in \{0,1\}^* \mid q \text{ is } \alpha\text{-heavy with respect to } \mathcal{Q}_x \,\}, \text{ and} \\ R_\alpha &:= L_\alpha \setminus \mathrm{Im}(G). \end{aligned}$$

For large enough $\alpha > 0$, we can easily show that R_α γ-avoids G.

Claim 25. *For any $\gamma\colon \mathbb{N} \to [0,1)$ and $\alpha > 0$ and for any length $\ell \in \mathbb{N}$, if*

$$\gamma(\ell) + 1/\alpha \leq 1 - 2^{s(\ell)-\ell},$$

then R_α is a γ-avoiding set at length ℓ for G.

Proof. Since $R_\alpha \subseteq \{0,1\}^* \setminus \mathrm{Im}(G)$, it suffices to show that $\Pr_{w \in_R \{0,1\}^\ell}[w \notin R_\alpha] \leq 1 - \gamma(\ell)$. Note that $w \notin R_\alpha$ if either $w \in \mathrm{Im}(G_\ell)$ or w is α-heavy. The probability of the former case is at most $2^{-\ell} \cdot |\mathrm{Im}(G_\ell)| \leq 2^{s(\ell)-\ell}$. Similarly, the probability of the latter case is bounded above by $2^{-\ell} \cdot |H_\alpha^{=\ell}|$, where $H_\alpha^{=\ell} = \{\, q \in \{0,1\}^\ell \mid q \text{ is } \alpha\text{-heavy} \,\}$. On the other hand, we have

$$|H_\alpha^{=\ell}| \cdot \alpha 2^{-\ell} \;\leq\; \sum_{q \in H_\alpha^{=\ell}} \Pr_{w \sim \mathcal{Q}_x}[\, w = q \,] \;=\; \Pr_{w \sim \mathcal{Q}_x}[\, w \in H_\alpha^{=\ell} \,] \;\leq\; 1.$$

Hence, $|H_\alpha^{=\ell}| \leq 2^\ell/\alpha$. Thus,

$$\Pr_{w \in_R \{0,1\}^\ell}[w \notin R_\alpha] \;\leq\; 2^{s(\ell)-\ell} + 1/\alpha \;\leq\; 1 - \gamma(\ell),$$

proving that $\Pr_{w \in_R \{0,1\}^\ell}[w \in R_\alpha] \geq \gamma(\ell)$. □

Since the reduction M does not make any query q such that $|q| \notin [\theta(n), t(n)]$, Claim 25 guarantees that the reduction M works by using R_α on inputs x of length n if $\gamma(\ell) + 1/\alpha \leq 1 - 2^{s(\ell)-\ell}$ for all $\ell \in \mathbb{N}$ such that $\theta(n) \leq \ell \leq t(n)$. As this condition is satisfied by our assumptions of Theorem 24 for any $\alpha \geq \alpha_0(n)$ and any input x of length n, we have

$$\Pr_M\left[M^{R_\alpha}(x) = L(x)\right] \;\geq\; \frac{15}{16}. \tag{5}$$

On the other hand, we can show below that $M(x)$ cannot distinguish R_α and L_α when α is small enough.

Claim 26. *For any $\alpha > 0$ and input $x \in \{0,1\}^*$ of length n, and for $\epsilon := \alpha 2^{s(\theta(n))-\theta(n)} \cdot m(n)t(n)$,*

$$\Pr_M \left[M^{L_\alpha}(x) \neq M^{R_\alpha}(x) \right] \ \leq \ \epsilon. \tag{6}$$

Proof. Recall that $R_\alpha = L_\alpha \setminus \mathrm{Im}(G)$. Thus, $M(x)$ may find the difference between L_α and R_α only if it makes a query in $L_\alpha \cap \mathrm{Im}(G)$ in one of its $m(n)$ nonadaptive queries. This probability is at most $m(n) \cdot \Pr_{w \sim \mathcal{Q}_x}[w \in L_\alpha \cap \mathrm{Im}(G)]$ by a union bound.

Here we have

$$\begin{aligned} \Pr_{w \sim \mathcal{Q}_x} [\, w \in L_\alpha \cap \mathrm{Im}(G) \,] \ &= \ \sum_{q \in L_\alpha \cap \mathrm{Im}(G)} \Pr_{w \sim \mathcal{Q}_x} [\, q = w \,] \\ &\leq \ \sum_{q \in \mathsf{supp}(\mathcal{Q}_x) \cap \mathrm{Im}(G)} \alpha \cdot 2^{-|q|}, \end{aligned}$$

where $\mathsf{supp}(\mathcal{Q}_x)$ is the set of all possible queries asked by $M(x)$. By our assumption on M, we have $\theta(n) \leq |q| \leq t(n)$ for any $q \in \mathsf{supp}(\mathcal{Q}_x)$. Then it follows

$$\sum_{q \in \mathsf{supp}(\mathcal{Q}_x) \cap \mathrm{Im}(G)} \alpha \cdot 2^{-|q|} \ \leq \ \sum_{\theta(n) \leq \ell \leq t(n)} \alpha \cdot 2^{-\ell} \cdot |\mathrm{Im}(G_\ell)| \ \leq \ \sum_{\theta(n) \leq \ell \leq t(n)} \alpha \cdot 2^{s(\ell)-\ell}$$

because $|\mathrm{Im}(G_\ell)| \leq 2^{s(\ell)}$.

Since we assumed that $\ell - s(\ell)$ is nondecreasing for $\ell \in \mathbb{N}$, we have

$$\sum_{\theta(n) \leq \ell \leq t(n)} \alpha 2^{s(\ell)-\ell} \ \leq \ t(n) \cdot \alpha 2^{s(\theta(n))-\theta(n)} \ = \ \epsilon / m(n).$$

This bound is sufficient to get the desired error bound. □

By Claim 26 and our assumptions on $\alpha_0(n)$ of Theorem 24, for any $\alpha \leq \mathrm{e}^3 \alpha_0(n)$, we have

$$\Pr_M \left[M^{L_\alpha}(x) \neq M^{R_\alpha}(x) \right] \leq \frac{1}{16}. \tag{7}$$

From the inequalities (5) and (7), we immediately obtain the following:

Corollary 27. *For any input $x \in \{0,1\}^*$ of length n and any $\alpha \in [\alpha_0(n), \mathrm{e}^3 \alpha_0(n)]$,*

$$\Pr_M \left[M^{L_\alpha}(x) = L(x) \right] \ \geq \ \frac{7}{8}.$$

In light of this, our task is now to simulate $M^{L_\alpha}(x)$ for some $\alpha \in [\alpha_0(n), \mathrm{e}^3 \alpha_0(n)]$ (in fact, we will choose the threshold α randomly, as explained later). To this end, we combine the generalized Feigenbaum-Fortnow protocol, the lower bound protocol of Goldwasser and Sipser [GS86], and the heavy-sample protocol of Bogdanov and Trevisan [BT06]. Here we review the last two protocols. Since these protocols are explained carefully and in detail in the paper [BT06], we simply review their specifications and use them as a black-box tool.

Lower Bound Protocol. Recall that $q \notin L_\alpha$ if and only if q is α-heavy. The lower bound protocol of Goldwasser and Sipser [GS86] can be used to give an AM-type witness to any α-heavy instance.

It is an AM protocol for showing that a given set of strings has more than s elements for a given threshold s. The specification of the lower bound protocol is as follows.

Inputs. A set of strings is given as a circuit C on $\{0,1\}^m$, which specifies the set as $C^{-1}(1) := \{\, r \in \{0,1\}^m \mid C(r) = 1 \,\}$. A threshold $s \in \mathbb{N}$ such that $0 \leq s \leq 2^m$. Parameters $\delta, \epsilon \in [0,1]$ represented in unary.

Promise.

- Yes instances: $|C^{-1}(1)| \geq s$.
- No instances: $|C^{-1}(1)| \leq (1-\epsilon)s$.

Sketch of the Protocol.

1. (Verifier) Send a random hash function $h : \{0,1\}^m \to \{0,1\}^{m'}$ for some appropriate parameter m'.
2. (Prover) Send a string $y \in \{0,1\}^m$ claiming that $y \in C^{-1}(1)$ and $h(y) = 0^{m'}$ hold. (Such an y is called *an* AM*-type witness.*)
3. (Verifier) Check the correctness of the prover's claim on the witness y.

Theorem 28 (Correctness of the lower bound protocol [GS86]; see [BT06] for a proof)**.** *The lower bound protocol stated above satisfies the following:*

- *(Completeness) Given an yes instance, there exists a prover that makes the verifier accept with probability at least* $1 - \delta$.
- *(Soundness) Given a no instance, for any prover, the verifier accepts with probability at most* δ.

By using the protocol above, it is easy to construct an AM verifier V that checks whether a given query q is α-heavy.

Claim 29. *For any parameter* $\epsilon(n) \geq 1/\mathsf{poly}(n)$*, there exists an* AM *verifier* V *such that, for any input* $x \in \{0,1\}^*$ *of length* n *and any query* $q \in \{0,1\}^*$*,*

1. *if* q *is* α*-heavy with respect to* $\mathcal{Q}_x$*, then* $\Pr_h[V(x,q,h,y) = 1 \text{ for some } y] \geq \frac{3}{4}$*, and*
2. *if* q *is* $(1-\epsilon(n))\alpha$*-light with respect to* $\mathcal{Q}_x$*, then* $\Pr_h[V(x,q,h,y) = 1 \text{ for some } y] \leq \frac{1}{4}$.

Proof. Let Q_x be the circuit that samples the query distribution $\mathcal{Q}_x$; that is, on input $r \in \{0,1\}^m$, the circuit $Q_x(r)$ outputs q so that $\Pr_{r \in_R \{0,1\}^m}[Q_x(r) = q] = \Pr_{w \sim \mathcal{Q}_x}[w = q]$ for any $q \in \{0,1\}^*$. Given a string $q \in \{0,1\}^*$ as input, construct a circuit C_q such that $C_q(r) := 1$ iff $Q_x(r) = q$, on input $r \in \{0,1\}^m$. Now use the lower bound protocol for the circuit C_q, the threshold $s := \alpha 2^{-|q|} 2^m$, and parameters $\delta := \frac{1}{4}$ and $\epsilon := \epsilon(n)$. The lower bound protocol gives an AM certificate for the yes instances such that $|C_q^{-1}(1)| \geq s$, which is equivalent to saying that $\Pr_r[Q_x(r) = q] \geq \alpha 2^{-|q|}$, that is, q is α-heavy. On the other hand, if q is $(1-\epsilon)\alpha$-light, then we have $|C_q^{-1}(1)| < (1-\epsilon)s$; thus with high probability there is no AM-type witness by the correctness of the lower bound protocol (Theorem 28). □

Note that there is a gap between yes instances and no instances; that is, if the probability that q is sampled from $\mathcal{Q}_x$ is between α and $(1-\epsilon)\alpha$, then the behavior of the lower bound protocol is undefined. To circumvent this, we pick the threshold α randomly in the same way with Bogdanov and Trevisan (cf. [BT06, Claim 3.2]): Consider the following set $\mathcal{A}_{\alpha_0,\epsilon}$ of thresholds defined by parameters $\alpha_0, \epsilon > 0$, and choose the threshold α uniformly at random from $\mathcal{A}_{\alpha_0,\epsilon}$.

$$\mathcal{A}_{\alpha_0,\epsilon} \;:=\; \{\, \alpha_0(1+3\epsilon)^i \mid 0 \le i \le 1/\epsilon \,\}.$$

Observe that $\mathcal{A}_{\alpha_0,\epsilon} \subseteq [\alpha_0, \mathrm{e}^3\alpha_0]$. Moreover, the following holds.

Lemma 30. *For every $\alpha_0 > 0$ and $0 < \epsilon < \frac{1}{3}$ and any constant $c > 0$, and for any distribution $\mathcal{Q}$, with probability at least $1 - 1/c$ over the choice of $\alpha \in_R \mathcal{A}_{\alpha_0,\epsilon}$,*

$$\Pr_{q\sim\mathcal{Q}}\Big[\mathcal{Q}(q) \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})\Big] \le c\epsilon. \tag{8}$$

(Recall that $\mathcal{Q}(q) := \Pr_{w\sim\mathcal{Q}}[w = q]$.)

Proof. For any $\epsilon \in \left(0, \frac{1}{3}\right)$ and $q \in \{0,1\}^*$, the intervals $((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})$ are pairwise disjoint for all $\alpha \in \mathcal{A}_{\alpha_0,\epsilon}$; hence for any real $p \in \mathbb{R}$, the probability that $p \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})$ is at most $1/|\mathcal{A}_{\alpha_0,\epsilon}| \le \epsilon$ over the choice of $\alpha \in_R \mathcal{A}_{\alpha_0,\epsilon}$. In particular, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\alpha\in_R\mathcal{A}_{\alpha_0,\epsilon}}\left[\Pr_{q\sim\mathcal{Q}}\Big[\mathcal{Q}(q) \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})\Big]\right] \\
&= \mathop{\mathbb{E}}_{q\sim\mathcal{Q}}\left[\Pr_{\alpha\in_R\mathcal{A}_{\alpha_0,\epsilon}}\Big[\mathcal{Q}(q) \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})\Big]\right] \;\le\; \epsilon.
\end{aligned}$$

Therefore, by Markov's inequality, the probability that $\Pr_{q\sim\mathcal{Q}}\left[\mathcal{Q}(q) \in ((1\pm\epsilon)\alpha 2^{-|q|})\right] \ge c\epsilon$ is at most $\epsilon/(c\epsilon) = 1/c$. □

In our simulation protocol for M, we start with picking $\alpha \in_R \mathcal{A}_{\alpha_0,\epsilon}$ randomly. By Lemma 30, except for probability $1/O(1)$, the heaviness of almost all queries q sampled from $\mathcal{Q}_x$ is not close to the threshold α. As a consequence, the distributional problem $(L_\alpha, \mathcal{Q}_x)$ is solvable by coAM on average; indeed, with probability at least $1 - O(\epsilon)$ over the choice of $q \sim \mathcal{Q}_x$, the lower bound protocol of Claim 29 solves L_α.

Heavy-sample Protocol. Next we review the heavy-sample protocol of Bogdanov and Trevisan [BT06], which is an AM protocol for estimating $\Pr_{q\sim\mathcal{Q}_x}[q \text{ is } \alpha\text{-heavy}]$.

Inputs. A circuit Q which samples a string according to a distribution $\mathcal{Q}$ on $\{0,1\}^*$. A probability $p \in [0,1]$ represented in binary. Parameters $c > 0$ and $0 < \epsilon < \frac{1}{3}$ represented in unary. A threshold $\alpha > 0$ represented in binary.

Promise.

- Yes instances: $\Pr_{q\sim\mathcal{Q}}[\mathcal{Q}(q) \ge \alpha 2^{-|q|}] = p$.
- No instances: $\left|\Pr_{q\sim\mathcal{Q}}[\mathcal{Q}(q) \ge \alpha 2^{-|q|}] - p\right| > 16c\epsilon$.
- We assume the condition (8). That is,

$$\Pr_{q\sim\mathcal{Q}}\Big[\mathcal{Q}(q) \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})\Big] \le c\epsilon.$$

Sketch of the Protocol.

1. (Verifier) Generate random queries $q_1, \ldots, q_k$ from the distribution $\mathcal{Q}$ for some sufficiently large k, and send these queries to the prover.
2. (Prover) For each query q_i, tell the verifier whether q_i is α-heavy.
3. (Verifier and Prover) To check the prover's claim, run the lower bound protocol of Goldwasser and Sipser [GS86] and the upper bound protocol of Fortnow [For89] in parallel.

Theorem 31 (Correctness of the heavy-sample protocol [BT06])**.** *The heavy-sample protocol specified above satisfies following:*

- *(Completeness) Given any yes instance, there exists a prover that makes the verifier accept with probability at least* $1 - O(\epsilon)$*.*
- *(Soundness) Given any no instance, for any prover, the verifier accepts with probability at most* $O(\epsilon)$*.*

Protocol for Simulating $M^{L_\alpha}(x)$. Using the protocols reviewed above, we can now simulate the reduction M to an oracle L_α on input $x \in \{0,1\}^*$. Below we explain how to choose the inputs $(C, V, \delta := \frac{1}{100}, p^*)$ for the generalized Feigenbaum-Fortnow protocol.

The generalized Feigenbaum-Fortnow protocol requires an AM protocol for solving an oracle on average (instead of coAM). By negating answers from the oracle, we can define a new machine $\overline{M}^X$ which simulates the computation of $M^{\{0,1\}^* \setminus X}$ for any given oracle X. We thus use the generalized Feigenbaum-Fortnow protocol for simulating $\overline{M}^{H_\alpha}(x)$ with oracle $H_\alpha := \{0,1\}^* \setminus L_\alpha$, which is the set of α-heavy queries with respect to $\mathcal{Q}_x$; more specifically, let C be the circuit that simulates the reduction $\overline{M}$ on input x (where the input x is hardwired into the circuit) and we give the circuit C to the protocol as input.

To solve H_α on average by an AM protocol, we use the lower bound protocol. That is, we build a circuit V_x that simulates the AM verifier stated in Claim 29 on input x and on all the queries $q \in \{0,1\}^*$ that $M(x)$ can make. Then we give the circuit V_x to the protocol as input.

We also need to give as advice a probability p^* that approximates $\Pr_{q \sim \mathcal{Q}_x}[q \in H_\alpha]$, which can be estimated by using the heavy-sample protocol. We require the prover to send p^* first, and then we verify the prover's claim by running the heavy-sample protocol; if the test passes, then we give p^* to the generalized Feigenbaum-Fortnow protocol as input.

Summarizing the discussion above, our whole simulation algorithm is given below.

Inputs. A string $x \in \{0,1\}^*$ of length n.

Promise. Let $\alpha_0 := \alpha_0(n)$. Then, for any $\alpha \in [\alpha_0, \mathrm{e}^3\alpha_0]$, we assume that

$$\Pr_M[M^{L_\alpha}(x) = L(x)] \geq \frac{7}{8},$$

(which is guaranteed by Corollary 27).

Protocol.

1. (Preprocess) Set an error parameter $\epsilon := 1/c_0 m(n)$ for a sufficiently large constant c_0 (represented in unary).
2. (Verifier) Pick a threshold $\alpha \in_R \mathcal{A}_{\alpha_0,\epsilon} \subseteq [\alpha_0, \mathrm{e}^3\alpha_0]$ randomly. Send α to the prover.
3. (Prover) Send $p^* \in [0,1]$ to the verifier. An honest prover sends $p^* := \Pr_{q\sim\mathcal{Q}_x}[q \in H_\alpha]$.
4. (Verifier and Prover) Run the heavy-sample protocol in order to verify that $p^* \approx \Pr_{q\sim\mathcal{Q}_x}[q \in H_\alpha]$ for the distribution $\mathcal{Q}_x$ and error parameter ϵ and parameter $c := 100$; if the test does not pass, output $\perp$ and halt.
5. (Verifier and Prover) Build a circuit C simulating $\overline{M}$ on the hardwired input x, and a circuit V_x simulates the AM verifier for α-heaviness. Run the generalized Feigenbaum-Fortnow protocol on input $(C, V_x, \delta := \frac{1}{100}, p^*)$, and output the result of the protocol.

Now we argue that our simulation protocol is correct:

Claim 32. *The simulation protocol stated above satisfies the following: for any $x \in \{0,1\}^*$,*

- *(Completeness) there exists a prover such that the verifier outputs $L(x)$ with probability at least $3/4$, and*
- *(Soundness) for any prover, the verifier outputs $L(x)$ or $\perp$ with probability at least $3/4$.*

Proof. Fix any input $x \in \{0,1\}^*$. By Lemma 30, with probability at least $1 - \frac{1}{100}$ over the choice of $\alpha \in_R \mathcal{A}_{\alpha_0,\epsilon}$, the condition (8) holds for $c := 100$ and $\mathcal{Q} := \mathcal{Q}_x$; that is,

$$\Pr_{q\sim\mathcal{Q}_x}\left[\mathcal{Q}_x(q) \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})\right] \leq 100\epsilon.$$

In what follows, we assume this event happens and analyze the probability of the completeness and soundness.

Suppose that a prover sends p^*. If the prover is honest then we have $p^* = \Pr_{q\sim\mathcal{Q}_x}[q \in H_\alpha]$. Hence the completeness of the heavy-sample protocol implies that, with probability at least $1 - O(\epsilon) \gg 1 - \frac{1}{100}$, the protocol accepts. On the other hand, if a cheating prover sends p^* such that $|p^* - \Pr_{q\sim\mathcal{Q}_x}[q \in H_\alpha]| > 16c\epsilon$, then with probability at least 0.99 the cheat can be caught by the soundness of the heavy-sample protocol.

Thus, at the point that the generalized Feigenbaum-Fortnow protocol starts, for any prover, with probability at least 0.98, we have $|p^* - \Pr_{q\sim\mathcal{Q}_x}[q \in H_\alpha]| \leq 16c\epsilon$ and moreover the condition (8) holds. Under this event, the promise of the generalized Feigenbaum-Fortnow protocol is satisfied:

- Define $a := L(x)$, $\epsilon_0 := \frac{1}{8}$, and $R := H_\alpha$. Then we have $\Pr_C[C^R = a] \geq 1 - \epsilon_0$ by the promise of our simulation protocol (Corollary 27).
- The advice p^* satisfies $|p^* - \Pr_{q\sim\mathcal{Q}_x}[q \in R]| \leq \epsilon_1$ for $\epsilon_1 := 16c\epsilon$.
- By Claim 29 and (8), we have $\Pr_{q\sim\mathcal{Q}_x}[q \notin V_{\mathrm{YES}} \cup V_{\mathrm{NO}}] \leq \Pr_{q\sim\mathcal{Q}_x}[\mathcal{Q}_x(q) \notin (1 \pm \epsilon)\alpha 2^{-|q|}] \leq \epsilon_2$ for $\epsilon_2 := 100\epsilon$.

Therefore, from the correctness of the generalized Feigenbaum-Fortnow protocol (Theorem 18), our simulation protocol satisfies the completeness and soundness with probability at least $1 - 0.02 - (\frac{1}{8} + 2m(n)\epsilon_1 + 3m(n)\epsilon_2 + 3\delta) \geq \frac{3}{4}$, where the last inequality holds for some large constant c_0. □

This completes the proof of Theorem 24.

6 GapMCSP is Low for $\mathsf{S}_2^{\mathrm{p}}$ if GapMCSP is in P/poly

In this section, we present an application of our proof techniques. We show that our $\mathsf{S}_2^{\mathrm{p}}$ protocol can be combined with the non-black-box reduction of [Hir18], under the assumption that GapMCSP $\in \mathsf{P}/\mathsf{poly}$.

Theorem 33. *Let A, R be any oracles such that $R \in \mathsf{P}^A/\mathsf{poly}$. Let $\alpha > 0$ be any constant. If $\mathrm{Gap}_\alpha\mathrm{MCSP}^A \in \mathsf{P}^R/\mathsf{poly}$, then $\mathsf{S}_2^{\mathrm{Gap}_\beta\mathrm{MCSP}^A} \subseteq \mathsf{S}_2^R$ for some constant $\beta > 0$.*

Prior to our work, it was shown by Cai, Chakaravarthy, Hemaspaandra, and Ogihara [CCHO05] that any downward self-reducible language $L \in \mathsf{P}/\mathsf{poly}$ is low for $\mathsf{S}_2^{\mathrm{p}}$. Before proving Theorem 33, we generalize their results to any language that has a selector, and explain a general proof strategy for showing lowness for $\mathsf{S}_2^{\mathrm{p}}$. Roughly speaking, a selector for L is an efficient algorithm that computes L given two oracles one of which is guaranteed to be equal to L.

Definition 34 (Selector [Hir15])**.** *A selector for a language L is a randomized polynomial-time oracle machine S such that, for any input $x \in \{0,1\}^*$ and oracles $A_0, A_1 \subseteq \{0,1\}^*$, if $L = A_0$ or $L = A_1$ then $\Pr_S\left[S^{A_0,A_1}(x) = L(x)\right] \geq \frac{2}{3}$.*

It was shown in [Hir15] that any downward self-reducible language admits a selector. Thus the following generalizes [CCHO05].

Theorem 35. *Let L be a language with a selector S and R be any oracle. If $L \in \mathsf{P}^R/\mathsf{poly}$ then $\mathsf{S}_2^L \subseteq \mathsf{S}_2^R$.*

Proof. The idea is as follows: We request two competing provers of S_2^R to send R-oracle circuits C_0, C_1 that compute L. Then, for every query q of L, one can decide whether $q \stackrel{?}{\in} L$ by running S and using C_0^R and C_1^R as oracles. Details follow.

Take any $A \in \mathsf{S}_2^L$, and let V be an S_2^L-machine that witnesses $A \in \mathsf{S}_2^L$. Take some constants c, d such that V runs in time n^c and S runs in time n^d on inputs of length n.

Now we describe an $\mathsf{S}_2 \cdot \mathsf{BP} \cdot \mathsf{P}^R$ algorithm that computes A: Given an input $x \in \{0,1\}^*$ of length n, for each $i \in \{0,1\}$, the ith competing prover sends an S_2-type certificate y_i for M. Moreover, each prover sends a polynomial-size R-oracle circuit C_i; an honest prover sends a circuit C_i such that C_i^R computes L on every input of length at most n^{cd}. Then simulate V using the two certificates (i.e., run V on input (x, y_0, y_1)), where each query q that V makes is answered with $S^{C_0^R, C_1^R}(q)$. It is easy to see the correctness of this algorithm.

Therefore, we have $A \in \mathsf{S}_2 \cdot \mathsf{BP} \cdot \mathsf{P}^R$, and by [RS98], we can derandomize the randomized computation by using the power of S_2 and obtain $L \in \mathsf{S}_2 \cdot \mathsf{BP} \cdot \mathsf{P}^R = \mathsf{S}_2^R$. □

In light of Theorem 35, at the core of Theorem 33 is the existence of a "non-black-box" selector. That is, we claim that one can efficiently compute $\mathrm{Gap}_\beta\mathrm{MCSP}^A$ given two polynomial-size circuits one of which is guaranteed to compute $\mathrm{Gap}_\alpha\mathrm{MCSP}^A$.

Lemma 36. *Let A, R be any oracles such that $R \in \mathsf{P}^A/\mathsf{poly}$. Let $\alpha, c > 0$ be any constants. Then, there exist a randomized polynomial-time algorithm S and a constant β such that, for any $n \in \mathbb{N}$ and any R-oracle circuits C_0, C_1 of size n^c such that one of them computes $\mathrm{Gap}_\alpha\mathrm{MCSP}^A$ on every input of length at most n, for any instance $x \in \{0,1\}^n$ of $\mathrm{Gap}_\beta\mathrm{MCSP}^A$, $S^R(x, C_0, C_1)$ decides x correctly with high probability.*

We call S a "non-black-box" selector because the algorithm makes use of the property that two candidate algorithms solving $\mathrm{Gap}_\alpha\mathrm{MCSP}^A$ are modeled as polynomial-size circuits instead of oracles. The non-black-box property prevents us from generalizing Theorem 33 for every R unlike Theorem 35. We leave it as an interesting open question whether Theorem 33 holds for any oracles R and A.

The non-black-box selector S uses the property that $\mathrm{GapMCSP}^A$ is reducible to any *polynomial-size* oracle avoiding $G^{\mathsf{int},A}$, as captured in the following lemma.

Lemma 37 ($\mathrm{GapMCSP}^A$ Reduces to Natural Properties)**.** *Let A, R be any oracles. Let $\alpha > 0$ be any constant. Let $D = \{D_m\}_{m\in\mathbb{N}}$ be a family of R-oracle polynomial-size circuits D_m^R on 2^m inputs that $\frac{7}{8}$-avoids $G^{\mathsf{int},A}\colon \{0,1\}^{\widetilde{O}(2^{\alpha m})} \to \{0,1\}^{2^m}$. Then, there exist a constant $\beta > 0$ and a polynomial-time computable function G that takes a size parameter $s \in \mathbb{N}$, a truth table of a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ and a string z and returns a string $G_s^f(z)$ of length $m = m(n,s) \le 2^n$ such that:*

1. *if $\mathsf{size}^A(f) \le s$, then $D(G_s^f(z)) = 0$ for every z, and*
2. *if $\mathsf{size}^R(f) \ge 2^{(1-\beta)n}s$, then $\Pr_z[D(G_s^f(z)) = 1] \ge \frac{3}{4}$.*

In the case when $A = \varnothing$, Lemma 37 follows from the work of Carmosino, Impagliazzo, Kabanets and Kolokolova [CIKK16]. Indeed, they implicitly showed that the search version of $\mathrm{Gap}_\beta\mathrm{MCSP}$ is reducible to D; using the fact that $\mathrm{MCSP} \in \mathsf{NP}$, one can reduce the decision problem $\mathrm{Gap}_\beta\mathrm{MCSP}$ to its search version (see [Hir18]). However, an efficient decision-to-search reduction for $\mathrm{Gap}_\beta\mathrm{MCSP}^A$ may not exist for a general oracle A, and thus we need a more direct proof.

Proof. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. We define G as the Nisan-Wigderson generator [NW94] instantiated with a hardness-amplified version of f as an underlying function. Specifically, let k be a parameter chosen later. We define a hardness-amplified version of f as $f^{\oplus k}(x^1, \cdots, x^k) = f(x^1) \oplus \cdots \oplus f(x^k)$ for $(x^1, \cdots, x^k) \in \{0,1\}^{nk}$.

We proceed to the definition of the Nisan-Wigderson generator NW: Let p be an arbitrary prime such that $kn \le p \le 2kn$ and $m \in \mathbb{N}$ be a parameter chosen later. For each string $q \in \{0,1\}^m$, we associate a polynomial q' of degree m over the field $\mathbb{F}_p$ defined as $q'(a) = \sum_{i=1}^m q_i a^i$ for any $a \in \mathbb{F}_p$. Take an arbitrary subset $D \subseteq \mathbb{F}_p$ of size kn, and define $S_q := \{\, (a, q'(a)) \mid a \in D \,\} \subseteq (\mathbb{F}_p)^2$. For a string $z \in \{0,1\}^{p^2}$, denote by $z_{S_q} \in \{0,1\}^{kn}$ the string obtained by concatenating the ith bit of z for every $i \in S_q$, identifying $i \in (\mathbb{F}_p)^2$ with $i \in [p^2]$. The function $G_s^f(z)$ is defined as $\mathrm{NW}^{f^{\oplus k}}(z) := (f(z_{S_q}) \,|\, q \in \{0,1\}^m) \in \{0,1\}^{2^m}$, that is, the truth table of the Boolean function that maps $q \mapsto f(z_{S_q})$. (We will choose k and m depending on the size parameter s.)

Consider any f such that $\mathsf{size}^A(f) \le s$. From the construction, it is easy to see that

$$\mathsf{size}^A(\mathrm{NW}^{f^{\oplus k}}(z)) \le \mathsf{poly}(\mathsf{size}^A(f), m, k, n) \le \mathsf{poly}(s, m, k, n)$$

for every $z \in \{0,1\}^{p^2}$. For a sufficiently large m, this can be bounded above by $2^{\alpha m}$. Thus we can take m as the smallest integer such that $\mathsf{poly}(s, m, k, n) \le 2^{\alpha m}$. Since D avoids $G^{\mathsf{int},A}$, we obtain the first condition of Lemma 37 (i.e., $D_m(\mathrm{NW}^{f^{\oplus k}}(z)) = 0$).

Next, we prove the contrapositive of the second condition. We assume $\Pr_z[D_m^R(\mathrm{NW}^{f^{\oplus k}}(z)) = 1] < \frac{3}{4}$. Since D_m^R is $\frac{7}{8}$-dense, we have $\Pr_{w\in_R\{0,1\}^m}[D_m^R(w) = 1] \ge \frac{7}{8}$. In particular, D_m^R distinguishes

$\mathrm{NW}^{f^{\oplus k}}$ from the uniform distribution with advantage $\frac{7}{8}-\frac{3}{4}=\frac{1}{8}$. By the security proof of the Nisan-Wigderson generator [NW94], we obtain an oracle circuit C such that $\Pr_{\bar{x}\in_R\{0,1\}^{nk}}[C^{D_m^R}(\bar{x}) = f^{\oplus k}(\bar{x})] \geq \frac{1}{2}+\Omega(\frac{1}{2^m})$ and the size of C is $\mathsf{poly}(2^m)$. Since the size of the circuit D_m is at most a polynomial in the input length 2^m, we can replace oracle gates of C with D_m and obtain an R-oracle circuit C_1 of size $\mathsf{poly}(2^m)$ such that C_1^R computes $f^{\oplus k}$ with probability $\frac{1}{2}+\Omega(\frac{1}{2^m})$. By Yao's XOR lemma (cf. [GNW11]), we obtain a circuit C_2 of size $\mathsf{poly}(2^m, 1/\delta)$ such that $\Pr_{x\in_R\{0,1\}^n}[C_2^R(x) = f(x)] \geq 1-\delta$, where δ is an arbitrary parameter that satisfies $(1-\delta)^k \leq \Omega(\frac{1}{2^m})$. We define $k := \Theta(m/\delta)$ so that this inequality is satisfied, and we take $\delta := 2^{-\beta n}s/2n$ for some sufficiently small constant $\beta > 0$. Finally, we correct the error of C_2 in a trivial way: Let φ be a DNF formula of size at most $\delta n 2^n$ such that $\varphi(x) = 1$ if and only if $C_2^R(x) \neq f(x)$. Then a circuit $C_3^R := C_2^R \oplus \varphi$ is a circuit of size $\mathsf{poly}(2^m, 1/\delta) + \delta n 2^n$ that computes f exactly. By the definition of m, we have $2^m = \Theta(\mathsf{poly}(s, n, 1/\delta))$, and thus $\mathsf{poly}(2^m, 1/\delta) \leq \mathsf{poly}(s, n, 1/\delta) \leq (sn/\delta)^c$ for some constant c. Therefore, $\mathsf{size}^R(f) \leq 2^{c\beta n}(2n)^{2c} + 2^{(1-\beta)n-1}s < 2^{(1-\beta)n}s$ for $\beta < 1/(c+1)$ and a sufficiently large n. □

Now we construct a non-black-box selector S for GapMCSP.

Proof of Lemma 36. The idea is exactly the same with Theorem 16. Namely, given two oracles R_0, R_1 such that one of them is guaranteed to be a dense subset of random strings, $R_0 \cap R_1$ is also a dense subset of random strings. We apply this idea to the case of GapMCSP. In our case, by a "random string" we mean a truth table that cannot be compressed into a small circuit.

Let $f\colon \{0,1\}^n \to \{0,1\}$ and $s \in \mathbb{N}$ be an instance of $\mathrm{Gap}_\beta\mathrm{MCSP}^A$, and $m = m(n,s)$ be as in Lemma 37. Let C_0, C_1 be R-oracle circuits one of which computes $\mathrm{Gap}_\alpha\mathrm{MCSP}^A$ for every instance of length 2^m ($\leq 2^n$). For each $i \in \{0,1\}$, we fix the size parameter of the inputs to C_i to $2^{\alpha n/2}$; that is, define $C'^R_i(g) := C_i^R(g, 2^{\alpha n/2})$. When the ith circuit is correct, C'^R_i accepts every $g \in \{0,1\}^{2^m}$ such that $\mathsf{size}^A(g) \leq 2^{\alpha n/2}$, and rejects every g such that $\mathsf{size}^A(g) > 2^{(1-\alpha)n+\alpha n/2} = 2^{(1-\alpha/2)n}$. In particular, for a random $g \in_R \{0,1\}^{2^m}$, we have $\Pr[C'^R_i(g) = 1] \leq o(1)$ by a simple counting argument.

Here is the algorithm S for solving $\mathrm{Gap}_\beta\mathrm{MCSP}^A$: For each $i \in \{0,1\}$, we verify that C_i has a small number of YES instances by sampling. Specifically, we pick $g \in_R \{0,1\}^{2^m}$, verify that $C'^R_i(g) = 0$, and repeat this test $O(1)$ times. If one of these tests fails, it means that C'_i is not likely to compute $\mathrm{Gap}_\alpha\mathrm{MCSP}^A$ correctly; thus we redefine $C'_i := 0$. In particular, if $\Pr_{g\in_R\{0,1\}^{2^m}}[C'^R_i(g) = 1] \geq \frac{1}{16}$, then with high probability C'_i is redefined. We then define a circuit D so that $D^R(g) := \neg(C'^R_0(g) \vee C'^R_1(g))$ for $g \in \{0,1\}^{2^m}$. By the test above, with high probability, D^R is dense, that is, $\Pr_g[D^R(g) = 1] \geq \frac{7}{8}$. Now let G be the function of Lemma 37. Pick a random z and accept if and only if $D(G_s^f(z)) = 0$.

We claim the correctness of the algorithm S described above. We verify that the hypothesis of Lemma 37 for $R = A$ is satisfied with high probability. Since $R \in \mathsf{P}^A/\mathsf{poly}$, the R-oracle polynomial-size circuit D can be simulated by an A-oracle polynomial-size circuit. Moreover, D^R avoids $\mathrm{Im}(G^{\mathsf{int},A})$ since every $g \in \{0,1\}^{2^{\alpha n/2}}$ is rejected, and D^R is $\frac{7}{8}$-dense with high probability. Applying Lemma 37, the algorithm accepts every instance (f,s) such that $\mathsf{size}^A(f) \leq s$ and rejects every instance (f,s) such that $\mathsf{size}^A(f) \geq 2^{(1-\beta)n}s$, with high probability. □

Proof of Theorem 33. Let S be the algorithm of Lemma 36. Let c be a constant such that there exists a R-oracle circuit C of size n^c such that C^R computes $\mathrm{Gap}_\alpha\mathrm{MCSP}^A$. Let $L \in \mathsf{S}_2^{\mathrm{Gap}_\beta\mathrm{MCSP}^A}$,

and V be an S_2-type verifier for L. We present an $\mathsf{S}_2 \cdot \mathsf{BP} \cdot \mathsf{P}^R$ algorithm W: For each $i \in \{0,1\}$, the ith competing prover sends a polynomial-size R-oracle circuit C_i of size n^c and a certificate for V. Using the two certificates, W simulates V, and each query q that V makes is answered by running $S^R(q, C_0, C_1)$. The correctness of W follows from Lemma 36. Therefore, we have $L \in \mathsf{S}_2 \cdot \mathsf{BP} \cdot \mathsf{P}^R = \mathsf{S}_2^R$. □

7 Power of Kolmogorov Random Strings

In this section, we show that every language in the exponential-time hierarchy is reducible to the set of Kolmogorov-random strings under PH reductions.

Theorem 38. $\mathsf{EXPH} \subseteq \mathsf{PH}^{R_{\mathrm{K}_U}}$ *for any universal Turing machine* U.

Combining our results with [CDE+14, AFG13], we obtain the following:

Corollary 39. $\mathsf{EXPH} \subseteq \bigcap_U \mathsf{PH}^{R_{\mathrm{K}_U}} \subseteq \mathsf{EXPSPACE}$.

Proof. The lower bound follows from Theorem 38. For the upper bound, by the results of [CDE+14, AFG13], we have $\bigcap_U \mathsf{P}_{\parallel}^{R_{\mathrm{K}_U}} \subseteq \mathsf{PSPACE}$. By a standard padding argument, we obtain $\bigcap_U \mathsf{EXP}_{\parallel}^{R_{\mathrm{K}_U}} \subseteq \mathsf{EXPSPACE}$, from which the upper bound follows since $\mathsf{PH}^{R_{\mathrm{K}_U}} \subseteq \mathsf{EXP}_{\parallel}^{R_{\mathrm{K}_U}}$. □

Theorem 38 immediately disproves Conjecture 9 under the assumption that the exponential-time hierarchy does not collapse to the exponential-time analogue of BPP.

Corollary 40. $\bigcap_U \mathsf{P}_{\parallel}^{R_{\mathrm{K}_U}} \not\subseteq \mathsf{BPP}$ *unless* $\mathsf{EXPH} \subseteq \mathsf{BPEXP}$.

Proof. Assume that $\bigcap_U \mathsf{P}_{\parallel}^{R_{\mathrm{K}_U}} \subseteq \mathsf{BPP}$. Then by a padding argument, we also obtain $\bigcap_U \mathsf{EXP}_{\parallel}^{R_{\mathrm{K}_U}} \subseteq \mathsf{BPEXP}$. It follows from Theorem 38 that $\mathsf{EXPH} \subseteq \mathsf{BPEXP}$. □

Now we proceed to a proof of Theorem 38. Let H denote the halting problem. We first observe that, given H as oracle, one can compute any decidable language with oracle access to H in P^H.

Lemma 41. *Let k be any positive constant. Let M be any oracle machine that, on every input x, halts eventually and makes a query of length at most $|x|^k$. Then the language decided by M is in P^H.*

Proof Sketch. The proof is essentially the same with [ABK+06b, Theorem 27], and thus we just sketch a proof. The idea is to decide M in the following two steps: First, by using a binary search and the oracle access to H, one can decide the number of all the strings in H of length at most n^k in polynomial time. Then, given the number of strings in H of length at most n^k as advice, the computation of M^H becomes now computable, and hence it reduces to H. □

We also recall that the halting problem is reducible to the set of random strings under $\mathsf{P/poly}$ reductions, which was established by exploiting the fact that the set of random strings can be distinguisher for any computable pseudorandom generator.

Theorem 42 ([ABK+06b])**.** $H \in \mathsf{P}^{R_{\mathrm{K}_U}}/\mathsf{poly}$ *for any universal Turing machine* U.

While the ingredients above are enough to obtain EXP reductions, in order to obtain PH reductions, we make use of the efficient proof system of PH given by Kiwi, Lund, Spielman, Russell, and Sundaram [KLS$^+$00]. For simplicity, we state their results in the case of the number of alternation is 2, but their results hold for every constant number of alternation. We also state their results in terms of Σ_2^{EXP} instead of Σ_2^{P}.

Theorem 43 (Kiwi, Lund, Spielman, Russell, and Sundaram [KLS$^+$00])**.** *For every language L in Σ_2^{EXP}, there exists a randomized polynomial-time verifier such that,*

1. *for every input $x \in L$, there exists an oracle A such that for any oracle B, $V^{A,B}(x)$ accepts with probability 1, and*
2. *for every input $x \notin L$, for all oracles A, there exists an oracle B, $V^{A,B}(x)$ accepts with probability at most $\frac{1}{2}$.*

We are now ready to present a proof of Theorem 38.

Proof of Theorem 38. The main idea is that, given oracle access to the set of random strings, Theorem 42 tells us that there is a succinct witness for any exponential-time computation. We abbreviate R_{K_U} as R in this proof. We only give a detailed proof for $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{PH}^R$, since it is straightforward to extend the proof.

First, we present a proof that $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{EXP}^R$. Let V be an exponential-time verifier for $L \in \Sigma_2^{\mathsf{EXP}}$, and c be a constant such that for every input x of length n, it holds that $x \in L$ if and only if there exists a certificate y of length 2^{n^c} such that $V(x, y, z)$ accepts for all z of length 2^{n^c}; V runs in time $2^{n^{O(1)}}$. We regard the computation as a game between the first player A and the second player B.

Here is our exponential-time algorithm solving L with oracle access to R: Let $s_A(n)$ and $s_B(n)$ be some polynomials specified later. Given input x of length n, the algorithm accepts if and only if there exists an oracle circuit A of size $s_A(n)$ such that $V(x, \mathtt{tt}(A^R), \mathtt{tt}(B^R))$ accepts for all oracle circuits B of size $s_B(n)$, where $\mathtt{tt}(A^R)$ denotes the truth table of the function computed by A^R; the algorithm checks this condition by an exhaustive search. Since there are at most exponentially many circuits of polynomial size, this algorithm runs in exponential time.

We claim the correctness of the algorithm. Fix any input $x \in L$ of length n. In this case, the correctness readily follows from the fact that there exists a succinct witness under the oracle R: Indeed, let y_x be the lexicographically minimum certificate for $x \in L$. Since each bit of y_x is decidable (in the sense that the language $\{\, (x, i) \mid \text{the } i\text{th bit of } y_x \text{ is } 1 \,\}$ is decidable), by Theorem 42, there exists an oracle circuit A of size $s_A(n) := \mathsf{poly}(n, \log |y_x|) = \mathsf{poly}(n, n^c)$ such that $\mathtt{tt}(A^R) = y_x$. Thus our EXP^R algorithm accepts no matter how the adversarial circuit B is chosen.

Now fix any input $x \notin L$ of length n. This case requires a more delicate argument. Here we need to claim that, for every circuit A of size s_A, there exists a circuit B that encodes a strategy that beats the strategy of A^R. The point is that, given any circuit A, the lexicographically first strategy against the strategy of A^R is computable with oracle access to H. Indeed, let $z_{x,A}$ denote the lexicographically first string such that $V(x, \mathtt{tt}(A^R), z)$ rejects. Consider the language $L' := \{\, (x, A, i) \mid \text{the } i\text{th bit of } z_{x,A} \text{ is } 1 \,\}$. Since R is reducible to H, the language L' is computable with oracle access to H. By Lemma 41, $L' \in \mathsf{P}^H$; thus by Theorem 42, we obtain $L' \in \mathsf{P}^R/\mathsf{poly}$. This means that for every circuit A there exists a circuit B_A of size $\mathsf{poly}(n, |A|, \log |z_{x,A}|) = \mathsf{poly}(n, s_A(n), n^c)$ such that $\mathtt{tt}(B_A^R) = z_{x,A}$. Thus our algorithm rejects.

In order to extend the proof above to $\Sigma_{2k}^{\mathsf{EXP}}$ for every constant k, we modify the EXP^R algorithm so that it checks whether, given input x of length n, $\exists$ a circuit C_1 of size $s_1(n)$, $\forall$ a circuit C_2 of size $s_2(n)$, $\cdots$, $\forall$ a circuit C_{2k} of size $s_{2k}(n)$ such that a verifier $V(x, \mathtt{tt}(C_1^R), \cdots, \mathtt{tt}(C_{2k}^R))$ accepts, where $s_1(n), \cdots, s_{2k}(n)$ are some appropriately chosen polynomials.

We now explain how to reduce the complexity of the EXP reduction to PH. For simplicity, we again focus on a proof of $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{PH}^R$. Note that, in the proof above, the bottleneck of the computation is the evaluation of $V(x, \mathtt{tt}(A^R), \mathtt{tt}(B^R))$, where V is an exponential-time verifier. We replace V with the randomized polynomial-time verifier of Theorem 43; then we obtain the following Σ_2^R algorithm: Existentially guess a circuit A of size at most $s_A(n)$, and universally guess a circuit B of size at most $s_B(n)$ as well as a coin flip for V, Then accept if and only if $V^{A,B}(x)$ accepts on the guessed coin flip sequence. □

A similar technique gives the exact characterization of $\mathsf{S}_2^{\mathrm{exp}}$ in terms of black-box reductions to a dense subset of Kolmogorov-random strings.

Theorem 44 (The formal version of Theorem 7)**.** *Let $s\colon \mathbb{N} \to \mathbb{N}$ be a function such that $\ell^\epsilon \leq s(\ell) \leq \ell - 2$ for some constant $\epsilon > 0$ and every large ℓ. Let $\gamma\colon \mathbb{N} \to [0, \frac{1}{2}]$ be a function such that $\gamma(\ell) \geq 1/\ell^c$ for some constant $c > 0$ and every large ℓ. Fix any universal Turing machine U. Let $G_\ell\colon \{0,1\}^{s(\ell)} \to \{0,1\}^\ell$ be a function such that $G_\ell(d) = U(d)$ for every $d \in \{0,1\}^{s(\ell)}$ such that $|U(d)| = \ell$. Then we have*

$$\bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{EXP}^{R^{\leq \mathsf{poly}}} \quad = \quad \bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{BPEXP}^{R^{\leq \mathsf{poly}}} \quad = \quad \mathsf{S}_2^{\mathrm{exp}}.$$

Proof. By Theorem 17, we have $\bigcap_R \mathsf{BPEXP}^{R^{\leq \mathsf{poly}}} \subseteq \mathsf{S}_2^{\mathrm{exp}}$, and it is obvious that $\mathsf{EXP}^{R^{\leq \mathsf{poly}}} \subseteq \mathsf{BPEXP}^{R^{\leq \mathsf{poly}}}$; thus it remains to prove $\mathsf{S}_2^{\mathrm{exp}} \subseteq \mathsf{EXP}^{R^{\leq \mathsf{poly}}}$ for every R that γ-avoids G.

First, observe that Theorem 42 can be generalized to any dense subset R of Kolmogorov-random strings. Indeed, the proof of [ABK+06b] only exploits the property that the set of Kolmogorov-random strings can be used as a distinguisher for a computable hitting set generator. Thus we have $H \in \mathsf{P}^R/\mathsf{poly}$. In particular, relative to the oracle R, there exists a succinct witness for $\mathsf{S}_2^{\mathrm{exp}}$.

Let $L \in \mathsf{S}_2^{\mathrm{exp}}$ and V be an exponential-time verifier associated with L running in time 2^{n^k} for some constant k: That is, for every input x of length n,

1. if $x \in L$ then $\exists y \in \{0,1\}^{2^{n^k}}, \forall z \in \{0,1\}^{2^{n^k}}, V(x,y,z) = 1$, and
2. if $x \notin L$ then $\exists z \in \{0,1\}^{2^{n^k}}, \forall y \in \{0,1\}^{2^{n^k}}, V(x,y,z) = 0$.

Our $\mathsf{EXP}^{R^{\leq \mathsf{poly}}}$ algorithm is as follows: Instead of doubly exponentially many certificates $y, z \in \{0,1\}^{2^{n^k}}$, we exhaustively search all possible oracle circuits Y, Z of size at most $\mathsf{poly}(n)$ that take n^k inputs, check the condition that $\exists Y, \forall Z, V(x, \mathtt{tt}(Y^R), \mathtt{tt}(Z^R)) = 1$, and accept if and only if this condition is satisfied. The correctness follows from the fact that each bit of the lexicographically first witness can be computed by a polynomial-size circuit with oracle access to R. □

We mention that the reducibility notion of Theorem 44 can be significantly improved by using some efficient proof system:

Theorem 45. *For any G and γ satisfying the same condition with Theorem 44, we have*

$$\mathsf{EXP}^{\mathsf{NP}} \subseteq \bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{S}_2^R \subseteq \mathsf{S}_2^{\exp}.$$

The proof follows from the following result and Theorem 35.

Lemma 46 ([Hir15])**.** *For any $\mathsf{EXP}^{\mathsf{NP}}$-complete language L, there exists a* selector *for L. That is, there exists a randomized polynomial-time oracle machine S such that, for any input $x \in \{0,1\}^*$ and oracles $A_0, A_1 \subseteq \{0,1\}^*$, if $L \in \{A_0, A_1\}$ then $\Pr_S\left[S^{A_0,A_1}(x) = L(x)\right] \geq \frac{2}{3}$.*

Proof of Theorem 45. Under any dense subset R of Kolmogorov-random strings, we have $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{P}^R/\mathsf{poly}$ (by Theorem 42). Thus by taking any $\mathsf{EXP}^{\mathsf{NP}}$-complete problem L, we obtain $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{S}_2^L \subseteq \mathsf{S}_2^R$ by combining Lemma 46 and Theorem 35. □

Finally, we mention that in the case of reductions to the set of random strings, the $\mathsf{S}_2^{\mathrm{p}}$ reductions of Theorem 45 can be derandomized to obtain P^{NP} reductions.

Theorem 47. $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{P}^{\mathsf{NP}^{R_{\mathrm{K}_U}}}$.

Proof. By Theorem 45, we immediately obtain $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{S}_2^{R_{\mathrm{K}_U}}$. By the relativized version of Cai's result [Cai07], we have $\mathsf{P}^{\mathsf{NP}^{R_{\mathrm{K}_U}}} \subseteq \mathsf{S}_2^{R_{\mathrm{K}_U}} \subseteq \mathsf{ZPP}^{\mathsf{NP}^{R_{\mathrm{K}_U}}}$; thus it remains to derandomize the computation of ZPP under an $\mathsf{NP}^{R_{\mathrm{K}_U}}$ oracle. One can find the lexicographically first Kolmogorov-random string by a $\mathsf{P}^{\mathsf{NP}^{R_{\mathrm{K}_U}}}$ algorithm. By Lemma 41 and Theorem 42, the circuit complexity relative to an $\mathsf{NP}^{R_{\mathrm{K}_U}}$ oracle of any Kolmogorov-random string of length n is at least $n^{\Omega(1)}$. Thus by using a Kolmogorov-random string as a hard function of the Impagliazzo-Wigderson pseudorandom generator [IW97], one can derandomize the computation of ZPP under an $\mathsf{NP}^{R_{\mathrm{K}_U}}$ oracle. □

Acknowledgement

We thank the anonymous reviewers for helpful comments.

References

[ABFL14] Eric Allender, Harry Buhrman, Luke Friedman, and Bruno Loff. Reductions to the set of random strings: The resource-bounded case. *Logical Methods in Computer Science*, 10(3), 2014.

[ABK06a] Eric Allender, Harry Buhrman, and Michal Koucký. What can be efficiently reduced to the Kolmogorov-random strings? *Ann. Pure Appl. Logic*, 138(1-3):2–19, 2006.

[ABK+06b] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from Random Strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

[ABX08] Benny Applebaum, Boaz Barak, and David Xiao. On Basing Lower-Bounds for Learning on Worst-Case Assumptions. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 211–220, 2008.

[AD17] Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. *Inf. Comput.*, 256:2–8, 2017.

[ADF+13] Eric Allender, George Davie, Luke Friedman, Samuel Hopkins, and Iddo Tzameret. Kolmogorov Complexity, Circuits, and the Strength of Formal Theories of Arithmetic. *Chicago J. Theor. Comput. Sci.*, 2013, 2013.

[Adl78] Leonard M. Adleman. Two Theorems on Random Polynomial Time. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 75–83, 1978.

[AFG13] Eric Allender, Luke Friedman, and William I. Gasarch. Limits on the computational power of random strings. *Inf. Comput.*, 222:80–92, 2013.

[AGGM06] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 701–710, 2006.

[AGGM10] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. Erratum for: on basing one-way functions on NP-hardness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 795–796, 2010.

[AH17] Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 54:1–54:14, 2017.

[AHK17] Eric Allender, Dhiraj Holden, and Valentine Kabanets. The Minimum Oracle Circuit Size Problem. *Computational Complexity*, 26(2):469–496, 2017.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of mathematics*, pages 781–793, 2004.

[All12] Eric Allender. Curiouser and Curiouser: The Link between Incompressibility and Complexity. In *How the World Computes - Turing Centenary Conference and 8th Conference on Computability in Europe, CiE 2012, Cambridge, UK, June 18-23, 2012. Proceedings*, pages 11–16, 2012.

[All17] Eric Allender. The Complexity of Complexity. In *Computability and Complexity - Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*, pages 79–94, 2017.

[Bab85] László Babai. Trading Group Theory for Randomness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 421–429, 1985.

[BB15] Andrej Bogdanov and Christina Brzuska. On Basing Size-Verifiable One-Way Functions on NP-Hardness. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 1–6, 2015.

[BFKL10] Harry Buhrman, Lance Fortnow, Michal Koucký, and Bruno Loff. Derandomizing from Random Strings. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 58–63, 2010.

[BT06] Andrej Bogdanov and Luca Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006.

[Cai07] Jin-yi Cai. S_2^P is subset of ZPP^{NP}. *J. Comput. Syst. Sci.*, 73(1):25–35, 2007.

[Can96] Ran Canetti. More on BPP and the Polynomial-Time Hierarchy. *Inf. Process. Lett.*, 57(5):237–241, 1996.

[CCHO05] Jin-yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Inf. Comput.*, 198(1):1–23, 2005.

[CDE+14] Mingzhong Cai, Rodney G. Downey, Rachel Epstein, Steffen Lempp, and Joseph S. Miller. Random strings and tt-degrees of Turing complete C.E. sets. *Logical Methods in Computer Science*, 10(3), 2014.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 10:1–10:24, 2016.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

[For89] Lance Fortnow. The Complexity of Perfect Zero-Knowledge. *Advances in Computing Research*, 5:327–343, 1989.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GNW11] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR-Lemma. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 273–301. 2011.

[GS86] Shafi Goldwasser and Michael Sipser. Private Coins versus Public Coins in Interactive Proof Systems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 59–68, 1986.

[GV08] Dan Gutfreund and Salil P. Vadhan. Limitations of Hardness vs. Randomness under Uniform Reductions. In *Proceedings of the Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX)*, pages 469–482, 2008.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[Hir15] Shuichi Hirahara. Identifying an Honest EXP^NP Oracle Among Many. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 244–263, 2015.

[Hir18] Shuichi Hirahara. Non-black-box Worst-case to Average-case Reductions within NP. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018.

[HK18] Shuichi Hirahara and Akitoshi Kawamura. On characterizations of randomized computation using plain Kolmogorov complexity. *Computability*, 7(1):45–56, 2018.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[Hol05] Thomas Holenstein. Key agreement from weak bit agreement. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 664–673, 2005.

[Hol06] Thomas Holenstein. Pseudorandom Generators from One-Way Functions: A Simple Construction for Any Hardness. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 443–461, 2006.

[HP15] John M. Hitchcock and Aduri Pavan. On the NP-Completeness of the Minimum Circuit Size Problem. In *Proceedings of the Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 236–245, 2015.

[HRV13] Iftach Haitner, Omer Reingold, and Salil P. Vadhan. Efficiency Improvements in Constructing Pseudorandom Generators from One-Way Functions. *SIAM J. Comput.*, 42(3):1405–1430, 2013.

[HS17] Shuichi Hirahara and Rahul Santhanam. On the Average-Case Complexity of MCSP and Its Variants. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 7:1–7:20, 2017.

[HW16] Shuichi Hirahara and Osamu Watanabe. Limits of Minimum Circuit Size Problem as Oracle. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 18:1–18:20, 2016.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 220–229, 1997.

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs Time: Derandomization under a Uniform Assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001.

[KC00] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.

[KLS+00] Marcos A. Kiwi, Carsten Lund, Daniel A. Spielman, Alexander Russell, and Ravi Sundaram. Alternation in interaction. *Computational Complexity*, 9(3-4):202–246, 2000.

[Ko91] Ker-I Ko. On the Complexity of Learning Minimum Time-Bounded Turing Machines. *SIAM J. Comput.*, 20(5):962–986, 1991.

[Lev84] Leonid A. Levin. Randomness Conservation Inequalities; Information and Independence in Mathematical Theories. *Information and Control*, 61(1):15–37, 1984.

[MW17] Cody D. Murray and R. Ryan Williams. On the (Non) NP-Hardness of Computing Circuit Complexity. *Theory of Computing*, 13(1):1–22, 2017.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[Ost91] Rafail Ostrovsky. One-Way Functions, Hard on Average Problems, and Statistical Zero-Knowledge Proofs. In *Proceedings of the Structure in Complexity Theory Conference*, pages 133–138, 1991.

[Rab79] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Cambridge, MA, USA, 1979.

[RR97] Alexander A. Razborov and Steven Rudich. Natural Proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

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

[Rud97] Steven Rudich. Super-bits, Demi-bits, and NP/qpoly-natural Proofs. In *Proceedings of the Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 85–93, 1997.

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity*, 16(4):331–364, 2007.

[Vad06] Salil P. Vadhan. An Unconditional Study of Computational Zero Knowledge. *SIAM J. Comput.*, 36(4):1160–1214, 2006.

[Yap83] Chee-Keng Yap. Some Consequences of Non-Uniform Conditions on Uniform Classes. *Theor. Comput. Sci.*, 26:287–300, 1983.

A Security Proofs of Hitting Set Generators Based on SZK

In this section, we show that the security of a hitting set generator can be proved by using a uniform black-box reduction, based on a worst-case hardness assumption of SZK.

Reminder of Theorem 5. Let $\epsilon > 0$ be any constant, and R be any oracle $\frac{1}{2}$-avoiding $G^{\mathsf{int}} = \{G^{\mathsf{int}}_n : \{0,1\}^{n^\epsilon} \to \{0,1\}^n\}_{n\in\mathbb{N}}$. Then, $\mathsf{SZK} \subseteq \mathsf{BPP}^R$.

Proof Sketch. Take any problem $L \in \mathsf{SZK}$. Allender and Das [AD17] showed that there exists a candidate *auxiliary-input one-way function* f_L such that the task of solving L reduces to the task of inverting f_L. (This fact was first shown by Ostrovsky [Ost91]). Here we say that a randomized algorithm A inverts an auxiliary-input one-way function f_L if, for every auxiliary-input x, $\Pr_{A,y}[A(x, f_L(x,y)) \in f_L^{-1}(x,\text{-})] \geq 1/\mathsf{poly}(|x| + |y|)$. This task can be done by a randomized polynomial-time algorithm with oracle access to R, by constructing a candidate pseudorandom

(function) generator G^{f_L} based on f_L using [HILL99, GGM86], and observing that R can distinguish G^{f_L} from the uniform distribution. (See [ABK+06b, Theorem 45], [AH17] for details.) □

Next, we show that a worst-case hardness assumption of integer factorization implies the existence of a hitting set generator with a *nonadaptive* security proof.

Reminder of Theorem 6. Let $\epsilon > 0$ be any constant, and R be any oracle $\frac{1}{2}$-avoiding $G^{\mathsf{int}} = \{G^{\mathsf{int}}_n : \{0,1\}^{n^\epsilon} \to \{0,1\}^n\}_{n\in\mathbb{N}}$. Then, factoring the product of two primes can be done in $\mathsf{ZPP}^R_{\parallel}$.

Proof. We follow the proof of [ABK+06b, Theorem 47] showing that Integer Factorization is in $\mathsf{ZPP}^{\mathrm{MCSP}}$. The idea is that, by using [HILL99, GGM86], one can invert any auxiliary-input one-way function f_N with oracle access to R (cf. [ABK+06b, Theorem 45], [AH17]). For the purpose of integer factorization, we can use some *regular* one-way function f_N, and we observe that the security reduction of [HILL99] is nonadaptive in this case.

Let $N \in \mathbb{N}$ be an input. We may assume without loss of generality that N is odd; hence, $N = pq$ for some odd prime numbers p, q. We may also assume that $p \neq q$. Let $f_N \colon (\mathbb{Z}/N\mathbb{Z})^* \to (\mathbb{Z}/N\mathbb{Z})^*$ be Rabin's (auxiliary-input) one-way function [Rab79]: $f_N(x) = x^2 \mod N$, where we regard N as an auxiliary input. As observed in [Rab79], f_N is a 4-to-1 function. (That is, for any a with $\gcd(a, N) = 1$, the number of $x \in (\mathbb{Z}/N\mathbb{Z})^*$ such that $x^2 \equiv a \pmod N$ is 4. Indeed, the condition is equivalent to saying that $x^2 \equiv a \pmod p$ and $x^2 \equiv a \pmod q$, and each congruence has exactly 2 solutions; by the Chinese remainder theorem, we obtain exactly 4 solutions.) Rabin showed that the task of factoring N is efficiently reducible to the task of inverting f_N on average. Therefore, it remains to show that one can efficiently invert f_N with parallel queries to R.

From the regular candidate one-way function f_N, a candidate pseudorandom generator $G_N = \{G_{N,n} : \{0,1\}^{n/2} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ can be constructed as in [HILL99, Theorem 5.5] so that the reduction of the security proof for G_N is nonadaptive. By using a construction of [GGM86], we extend the output length of G_N so that $G'_N = \{G'_{N,n} : \{0,1\}^n \to \{0,1\}^{n^{O(1/\epsilon)}}\}_{n\in\mathbb{N}}$ with the property that $G'_{N,n}(z) \in \mathrm{Im}(G^{\mathsf{int}}_{n^{O(1/\epsilon)}})$ for every seed z (that is, the function represented by $G'_{N,n}(z)$ as a truth table can be computed by a circuit of size $n^{O(1)}$). Thus the oracle R distinguishes G'_N from the uniform distribution, and hence we obtain a nonadaptive randomized algorithm factoring N. Finally, the randomized algorithm can be made zero-error by using the primality test of [AKS04]. □

As an immediate corollary, we obtain a new hardness result of MCSP under nonadaptive reductions.

Corollary 48. *Factoring the product of two primes can be done in* $\mathsf{ZPP}^{\mathrm{Gap}_\epsilon\mathrm{MCSP}}_{\parallel}$ *for every constant* $\epsilon > 0$.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il