Unexpected Power of Random Strings

Shuichi Hirahara
National Institute of Informatics, Tokyo, Japan
s_hirahara@nii.ac.jp

Abstract

There has been a line of work trying to characterize BPP (the class of languages that are solvable by efficient *randomized* algorithms) by efficient nonadaptive reductions to the set of Kolmogorov-*random* strings: Buhrman, Fortnow, Koucký, and Loff (CCC 2010 [10]) showed that every language in BPP is reducible to the set of random strings via a polynomial-time nonadaptive reduction (irrespective of the choice of a universal Turing machine used to define Kolmogorov-random strings). It was conjectured by Allender (CiE 2012 [1]) and others that their lower bound is tight when a reduction works for every universal Turing machine; i.e., "the only way to make use of *random* strings by a nonadaptive polynomial-time algorithm is to *derandomize* BPP."

In this paper, we *refute* this conjecture under the plausible assumption that the exponential-time hierarchy does not collapse, by showing that the exponential-time hierarchy EXPH can be solved in exponential time by nonadaptively asking the oracle whether a string is Kolmogorov-random or not. In addition, we provide an exact characterization of $\mathsf{S}_2^{\mathrm{exp}}$ in terms of exponential-time-computable nonadaptive reductions to arbitrary dense subsets of random strings.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Problems, reductions and completeness

Keywords and phrases Kolmogorov-Randomness, Nonadaptive Reduction, BPP, Symmetric Alternation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.41

Related Version Part of the results of this paper appeared in `https://eccc.weizmann.ac.il/report/2019/025/`.

Funding *Shuichi Hirahara*: ACT-I, JST and JSPS KAKENHI Grant Number 18H04090.

Acknowledgements I thank Eric Allender, Michal Koucký, and Osamu Watanabe for helpful discussions, and thank anonymous reviewers for helpful comments. I got the ideas of this work during the joint work with Osamu Watanabe.

1 Introduction

Randomness arises everywhere in the theory of computation, and there are at least three different notions of randomness: (1) BPP, one of the most important complexity classes, is the class of languages solvable by a two-sided-error randomized polynomial-time algorithm with high probability. (2) Pseudorandom generators enable us to derandomize BPP. Its one-sided-error version is called a *hitting set generator*. (3) The notion of *Kolmogorov complexity* enables us to quantify the amount of randomness in a finite string from the perspective of compressibility. In this study, we explore the relationship existing among these different notions of "randomness" – efficient randomized algorithms, pseudorandomness, and Kolmogorov-randomness. We present the interplay among these notions of randomness, while reviewing the literature in the subsequent section.

1.1 Pseudorandomness and Kolmogorov-Randomness

The Kolmogorov complexity of a string x can informally expressed as the length of the shortest program that prints x. More formally, the Kolmogorov complexity of $x \in \{0,1\}^*$ with respect to a Turing machine U is defined as

$$\mathrm{K}_U(x) := \min\{\, |d| \mid U \text{ outputs } x \text{ on input } d \,\}.$$

We choose a "universal" Turing machine U so that the Kolmogorov complexity is smallest up to an additive constant. A Turing machine U is said to be *universal* if, for every Turing machine M, there exists a constant c such that $\mathrm{K}_U(x) \leq \mathrm{K}_M(x)+c$ for every string $x \in \{0,1\}^*$. Note that there exists a universal Turing machine because one can take a Turing machine U that, given an input (M,x), simulates a Turing machine M on input x.

Kolmogorov complexity enables us to define the notion of Kolmogorov-randomness. A string x is said to be *(Kolmogorov-)random* if there exists no shorter program that prints x. More generally, given a threshold $s\colon \mathbb{N} \to \mathbb{N}$, we say that $x \in \{0,1\}^\ell$ is s-random if $\mathrm{K}_U(x) \geq s(\ell)$. Throughout this paper, we fix any reasonable threshold s (e.g., $s(\ell) := \ell/2$) and denote by R_{K_U} the set of s-random strings, i.e., $R_{\mathrm{K}_U} := \{\, x \mid \mathrm{K}_U(x) \geq s(|x|) \,\}$.

An important property of the set R_{K_U} of random strings is that R_{K_U} is an extremely powerful distinguisher for any computable hitting set generator. More specifically, there is a simple but powerful connection between pseudorandomness and Kolmogorov-randomness: Consider any "pseudorandom" string x that is generated by a computable process from a short seed. By definition, the string x is not Kolmogorov-random; hence R_{K_U} distinguishes any pseudorandom distribution from the uniform distribution.

A function $G\colon \{0,1\}^{s(\ell)} \to \{0,1\}^\ell$ $(\, s(\ell) < \ell \,)$ is called a *hitting set generator* secure against a class $\mathcal{C}$ if there exists no $\mathcal{C}$-algorithm R that avoids G; here we say that R *avoids* G if R rejects every string in the range of G, and R accepts at least a half of inputs of length ℓ. The notion of hitting set generator naturally arises when one tries to derandomize one-sided-error randomized algorithms: For instance, if there is an efficient hitting set generator of seed length $s(\ell) = O(\log \ell)$ and secure against linear-size circuits, then by exhaustively trying all the seeds of G, one can completely derandomize one-sided-error randomized algorithms, i.e., $\mathsf{RP} = \mathsf{P}$.

What can be efficiently solved by asking the oracle R_{K_U} whether a string is random or not? As mentioned earlier, it is easy to observe that R_{K_U} avoids any computable family of functions $G = \{G_\ell\colon \{0,1\}^{s(\ell)-O(\log \ell)} \to \{0,1\}^\ell\}_{\ell \in \mathbb{N}}$.[1] More generally, any "dense" subset R of random strings R_{K_U} avoids any computable hitting set generator. (Here we say that $R \subseteq \{0,1\}^*$ is *dense* if R contains at least a half of inputs for each input length.) By exploiting this property, a line of work exhibited the power of R_{K_U} as an oracle. Allender, Buhrman, Koucký, van Melkebeek, and Ronneburger [5] proved that $\mathsf{PSPACE} \subseteq \mathsf{P}^{R_{\mathrm{K}_U}}$. Allender, Buhrman, Koucký [4] showed that $\mathsf{NEXP} \subseteq \mathsf{NP}^{R_{\mathrm{K}_U}}$. In the case of nonadaptive reductions, Buhrman, Fortnow, Koucký, and Loff [10] showed that $\mathsf{BPP} \subseteq \mathsf{P}_{\parallel}^{R_{\mathrm{K}_U}}$, where $\mathsf{P}_{\parallel}^{R_{\mathrm{K}_U}}$ denotes the class of languages solvable in polynomial time with nonadaptive oracle access to R_{K_U}.[2] Here, a reduction is said to be *nonadaptive* (also known as a truth-table or parallel-query reduction) if the queries of the reduction do not depend on previous answers from the oracle.

---

[1] Specifically, for every $\ell \in \mathbb{N}$ and seed $z \in \{0,1\}^{s(\ell)-O(\log \ell)}$, the Kolmogorov complexity of $G_\ell(z)$ is less than $s(\ell)$; thus $G_\ell(z) \notin R_{\mathrm{K}_U}$. On the other hand, a simple counting argument shows that $|R_{\mathrm{K}_U} \cap \{0,1\}^\ell| \geq 2^{\ell-1}$ for every ℓ.

[2] The subscript $\parallel$ stands for parallel queries.

1.2 BPP and Kolmogorov-Randomness

Kolmogorov-randomness and BPP originated from different disciplines. The notion of Kolmogorov-randomness usually sits within the realm of *computability theory* as R_{K_U} is not computable, whereas *complexity theory* deals with efficient computations, such as BPP. In particular, it is obvious that there is no computable upper bound on the class $\mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$, since R_{K_U} itself is not computable.

Nevertheless, a surprising interdisciplinary connection between complexity classes and Kolmogorov-random strings has been uncovered in the line of work (e.g., [4, 7, 13]). When a reduction works no matter which universal Turing machine U is used in the definition of R_{K_U}, a computable upper bound can be obtained. Cai, Downey, Epstein, Lempp, and Miller [13] showed that a language reducible to R_{K_U} for every prefix-free universal Turing machine U is computable; Allender, Friedman, and Gasarch [7] showed that any computable language L that reduces to R_{K_U} for every prefix-free universal Turing machine U via a polynomial-time nonadaptive reduction is in PSPACE. Therefore:

▶ **Theorem 1** ([10, 7, 13]).

$$\mathsf{BPP} \subseteq \bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}} \subseteq \mathsf{PSPACE},$$

where the intersection is taken over all prefix-free universal Turing machines U.

For some technical reasons, the upper bounds of [7, 13] are known to hold only for "prefix-free" universal Turing machines. A similar upper bound can be obtained for plain universal Turing machines by imposing a super-constant minimum query length restriction on reductions (cf. Hirahara and Kawamura [16]). Thus, we herein focus on (plain) universal Turing machines for the sake of simplicity.

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 [3] that, intuitively, short queries to Kolmogorov-random strings would only be used as a source of pseudorandomness. Allender [1] thus conjectured that the lower bound of Theorem 1 is exactly tight, and then a fair amount of effort has been made to verify the conjecture.

▶ **Conjecture 2** ([10, 1, 6, 3, 16]). $\mathsf{BPP} = \bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$, *where the intersection is taken over all universal Turing machines.*

Beyond its curiosity, such a characterization of BPP might make it possible to study BPP by using the techniques from computability theory, and could be a completely new approach for resolving the $\mathsf{P} = \mathsf{BPP}$ conjecture (as speculated in [1, 6]). Moreover, Conjecture 2 is interesting from the viewpoint of the study of the Minimum Circuit Size Problem (MCSP [20]). In some technical sense, Kolmogorov complexity can be seen as the minimum size of a circuit with oracle access to the halting problem (cf. [5]); thus R_{K_U} can be regarded as a computability-theoretic analogue of MCSP. In the light of this, Conjecture 2 states non-NP-hardness of R_{K_U} under nonadaptive polynomial-time reductions (unless $\mathsf{NP} \subseteq \mathsf{BPP}$), and thus would make it possible to obtain some new insights about NP-hardness of MCSP, which has been the focus of many recent studies on MCSP (e.g. [22, 17, 18, 9, 8]).

The "evidence" in favor of Conjecture 2 seemed to be pilling up, by considering various restrictions on the reductions. For example, Allender, Buhrman, Koucký [4] showed that conjunctive nonadaptive reductions from computable languages to R_{K_U} can be simulated

in P; Hirahara and Kawamura [16] showed that some restricted variant of $\bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$ lies between BPP and $\Sigma_2^{\mathrm{p}} \cap \Pi_2^{\mathrm{p}} \cap \mathsf{P/poly}$. We refer interested readers to the survey of Allender [2] for detailed background on MCSP and R_{K_U}.[3]

1.3 Our Results: Unexpected Power of Random Strings

In this work, we disprove Conjecture 2 under the plausible assumption that the exponential-time hierarchy does not collapse to BPEXP, by presenting unexpected power of R_{K_U}. Not only do we disprove Conjecture 2 itself, but also argue that the intuition of [3] is not correct.

Our main new insight is to consider a tally language in $\bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$. Recall that the intuition behind Conjecture 2 is that

1. On input 1^n, any long query q to R_{K_U} is answered as "No," since a nonadaptive polynomial-time machine cannot make any query whose Kolmogorov complexity is larger than $O(\log n)$, and that
2. Any query of length $O(\log n)$ *should be* useful only as a source of "pseudorandomness" [3]. We interpret this as follows: Short queries to R_{K_U} should be replaceable with queries to any oracle R that avoids any computable hitting set generator (i.e., R is a dense subset of random strings).

In order to closely examine these intuitions, let us focus on tally languages $L \subseteq \{1^n\}$. By a standard padding argument, this is essentially equivalent to considering an exponential-time analogue of Conjecture 2. More specifically, if Conjecture 2 is true, then a padding argument shows that

$$\mathsf{BPEXP} = \bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$$

is also true.[4] In order to examine the latter intuition, we also focus on an arbitrary oracle R that avoids U. Specifically, Conjecture 2 implies that the subclass $\bigcap_R \mathsf{EXP}_{\|}^{R}$ of $\bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$ is also contained in BPEXP, i.e.,

$$\bigcap_R \mathsf{EXP}_{\|}^{R} \subseteq \mathsf{BPEXP},$$

where the intersection is taken over all dense subsets R of random strings; thus, we first study the subclass $\bigcap_R \mathsf{EXP}_{\|}^{R}$.

We completely settle what can be solved by a nonadaptive exponential-time reduction to dense subsets R of random strings.

▶ **Theorem 3** (Characterizing $\mathsf{S}_2^{\mathrm{exp}}$ Using Efficient Reductions to Dense Subsets of Random Strings)**.** *For any universal Turing machine U, it holds that*

$$\mathsf{S}_2^{\mathrm{exp}} = \bigcap_R \mathsf{EXP}_{\|}^{R},$$

where the intersection is taken over all the dense subsets R of R_{K_U}. (The condition of R can be equivalently stated as follows: R avoids a universal Turing machine U regarded as a family of functions $U = \{U_\ell \colon \{0,1\}^{s(\ell)} \to \{0,1\}^{\ell}\}_{\ell \in \mathbb{N}}$.)

[3] Some unpublished results mentioned there are included in this paper.

[4] In fact, this padding argument together with [4] already shows that Conjecture 2 is false unless NEXP $\subseteq$ BPEXP, which was completely overlooked before this work. Our results significantly weaken the hypothesis.

Here $\mathsf{S}_2^{\mathrm{exp}}$ represents an exponential-time analogue of $\mathsf{S}_2^{\mathrm{p}}$ (the second level of the symmetric hierarchy [23, 14]), and is the class of languages which admit a "competing-two-prover" system: an exponential-time verifier must compute the correct answer when one of the provers presents a correct certificate.

Theorem 3 improves the result of [10] (conjectured to be tight) in two ways: First, it is known that $\mathsf{BPEXP} \subseteq \mathsf{S}_2^{\mathrm{exp}}$, and the inclusion is likely to be proper (e.g., note that $\mathsf{NEXP} \subseteq \mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{S}_2^{\mathrm{exp}}$ [23]); thus, Theorem 3 refutes Conjecture 2 unless $\mathsf{S}_2^{\mathrm{exp}} \subseteq \mathsf{BPEXP}$. Second, unlike the proof of [10], our proof does not exploit any specific property of R_{K_U}; we only rely on the pseudorandomness property of R_{K_U} (i.e., the property that R_{K_U} is a distinguisher for any computable hitting set generator).

It is also worthy of note that our $\mathsf{S}_2^{\mathrm{exp}}$ upper bound of Theorem 3 significantly improves the previous $\mathsf{EXPSPACE}$ upper bound of Allender, Friedman, and Gasarch [7]: They proved $\bigcap_R \mathsf{EXP}_{\|}^R \subseteq \mathsf{EXPSPACE}$ as a corollary of their PSPACE upper bound of Theorem 1.

Let us take a look at the original conjecture again. In Conjecture 2, we are allowed to exploit a property of R_{K_U}, instead of just a property that an oracle avoids a computable hitting set generator. In this case, it is possible to reduce the entire exponential-time hierarchy to R_{K_U} via a nonadaptive exponential-time reduction, or even more efficiently, via a PH-Turing reduction.

▶ **Theorem 4.** *For every universal Turing machine U,*

$$\mathsf{EXPH} \subseteq \mathsf{PH}^{R_{\mathrm{K}_U}} \subseteq \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}.$$

Theorem 4 shows that the class $\bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$ is much closer to the $\mathsf{EXPSPACE}$ upper bound of [7] than previously conjectured. To be specific, it implies that Conjecture 2 is false unless $\mathsf{EXPH} \subseteq \mathsf{BPEXP}$.

1.4 Proof Ideas

We briefly mention our proof techniques for proving the lower bounds of Theorem 3 and Theorem 4. It is known that the halting problem is reducible to dense subsets R of random strings via a *nonuniform* polynomial-time reduction [5]. Therefore, by exhaustively searching polynomial-size R-oracle circuits (in exponential time), we can find a "succinct witness" for $\mathsf{S}_2^{\mathrm{exp}}$, whose correctness can be verified in exponential time. In order to extend this argument to EXPH, recall that EXPH can be regarded as a constant-round two-player game whose winner can be computed in exponential time. A winning strategy of round $k+1$ must depend on previous strategies (R_{K_U}-oracle circuits) of rounds $1, \cdots, k$. Using the fact that an R_{K_U}-oracle circuit can be evaluated with oracle access to the halting problem, it can be argued that there exists a polynomial-size R-oracle circuit which encodes a winning strategy for round $k+1$.

In order to prove the $\mathsf{S}_2^{\mathrm{exp}}$ upper bound of Theorem 3, we simulate an exponential-time reduction M by an $\mathsf{S}_2^{\mathrm{exp}}$ algorithm as follows: The ith competing prover sends a set $\bar{R}_i$ for each $i \in \{1,2\}$; the honest prover sends the set $\{0,1\}^* \setminus R_{\mathrm{K}_U}$ of nonrandom strings. While we cannot know which of $\bar{R}_1$ or $\bar{R}_2$ is correct, $\{0,1\}^* \setminus (\bar{R}_1 \cup \bar{R}_2)$ is guaranteed to be a dense subset of R_{K_U}, to which the reduction M works correctly.

The rest of this paper is organized as follows. After reviewing some background in Section 2, Theorem 3 is proved in Section 3. Theorem 4 is proved in Section 4. In Appendix A, the $\mathsf{EXP}_{\|}$ reduction of Theorem 3 is improved.

2 Preliminaries

We identify a language $L \subseteq \{0,1\}^*$ with its characteristic function $L\colon \{0,1\}^* \to \{0,1\}$. Let $[n] := \{1,\dots,n\}$ for $n \in \mathbb{N}$. We say that $R \subseteq \{0,1\}^*$ is *dense* if $|R \cap \{0,1\}^n| \geq 2^{n-1}$ for every $n \in \mathbb{N}$. For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, we denote by $\mathtt{tt}(f)$ the truth table of f, i.e., the concatenation of $f(x)$ for all $x \in \{0,1\}^n$ in the lexicographical order. We often identify a Boolean circuit C with the function computed by C.

2.1 On the Threshold of Kolmogorov-Random Strings

Recall that a string $x \in \{0,1\}^*$ is said to be $s(\text{-})$-*random* if $\mathrm{K}_U(x) \geq s(|x|)$. Throughout this paper, we fix any reasonable threshold $s(\text{-})$ of randomness so that there exists a universal constant $\epsilon > 0$ satisfying $n^\epsilon \leq s(n) \leq n-2$ for any $n \in \mathbb{N}$, and denote by R_{K_U} the set of $s(\text{-})$-random strings, i.e., $R_{\mathrm{K}_U} := \{\, x \in \{0,1\}^* \mid \mathrm{K}_U(x) \geq s(|x|) \,\}$. The upper bound of the threshold is for assuming that the set of random strings is sufficiently dense:

▶ **Fact 5.** *$|\{0,1\}^n \setminus R_{\mathrm{K}_U}| \leq 2^{n-2}$ for any $n \in \mathbb{N}$.*

Proof. The number of nonrandom strings is bounded by the number of programs of length less than $s(n)$, which is at most $\sum_{i=0}^{s(n)-1} 2^i \leq 2^{s(n)} \leq 2^{n-2}$. ◀

2.2 Symmetric Alternation

$\mathsf{S}_2^{\mathrm{p}}$ denotes the class of languages that admit a competing-two-prover system. More formally:[5]

▶ **Definition 6** ([14, 23])**.** *$\mathsf{S}_2^{\mathrm{p}}$ is the class of languages L such that there exist a polynomial $p(n) = n^{O(1)}$ and a polynomial-time algorithm V such that, for every input $x \in \{0,1\}^*$,*

1. *$\exists y \in \{0,1\}^{p(|x|)}, \forall z \in \{0,1\}^{p(|x|)},\ V(x,y,z) = L(x)$, and*
2. *$\exists z \in \{0,1\}^{p(|x|)}, \forall y \in \{0,1\}^{p(|x|)},\ V(x,y,z) = L(x)$.*

$\mathsf{S}_2^{\mathrm{exp}}$ is an exponential-time analogue of $\mathsf{S}_2^{\mathrm{p}}$: the definition of $\mathsf{S}_2^{\mathrm{exp}}$ is obtained by allowing $p(n)$ to be exponential in the definition above (i.e., $p(n) = 2^{n^{O(1)}}$). For an oracle A, the relativized version of $\mathsf{S}_2^{\mathrm{p}}$ is denoted by S_2^A; that is, S_2^A is the class of languages that admit a competing-two-prover system with a polynomial-time A-oracle verifier.

2.3 Exponential-Time Hierarchy

EXPH denotes the exponential-time analogue of the polynomial-time hierarchy (PH). That is, a language L is in EXPH if and only if there exist a constant k, an exponential bound $e(n) = 2^{n^{O(1)}}$ and a polynomial-time algorithm M such that, for every input $x \in \{0,1\}^*$,

$$x \in L \iff \exists y_1, \forall y_2, \exists y_3, \forall y_4, \cdots, \exists y_k,\ M(x, y_1, \cdots, y_k) = 1,$$

where $y_i \in \{0,1\}^{e(n)}$ for any $i \in [k]$.

[5] We follow the definition of Canetti [14], which is equivalent to the definition of [23] (cf. [11]).

3 Reductions to Dense Subsets of Random Strings

In this section, we give the characterization of $\mathsf{S}_2^{\mathrm{exp}}$ by exponential-time nonadaptive reductions to arbitrary dense subsets of random strings.

▶ **Reminder of Theorem 3.** Fix any universal Turing machine U.

- *Lower bound:* $\mathsf{S}_2^{\mathrm{exp}} \subseteq \mathsf{EXP}_{\|}^{R}$ for any dense subset R of R_{K_U}.
- *Upper bound:* $\{ L \subseteq \{0,1\}^* \mid L \in \mathsf{EXP}_{\|}^{R}$ for every dense $R \subseteq R_{\mathrm{K}_U} \} \subseteq \mathsf{S}_2^{\mathrm{exp}}$.

3.1 Lower Bound

In order to prove the lower bound on the complexity class $\bigcap_R \mathsf{EXP}_{\|}^{R}$, we use the fact that the halting problem HALT is reducible to any dense subset of random strings via a polynomial-size oracle circuit. The halting problem HALT is the problem of taking as input a description of a Turing machine M and deciding whether M halts.

▶ **Theorem 7** ([5]). $\mathsf{HALT} \in \mathsf{P}^{R}/\mathsf{poly}$ *for any dense subset R of random strings.*

Proof Sketch. It was shown in [5, Corollary 32] that HALT is reducible to R_{K_U} via a nonadaptive P/poly reduction. The proof only uses the fact that R_{K_U} is a distinguisher for some hitting set generator; thus HALT is also reducible to any dense subset R of random strings. ◀

Proof of the Lower Bound of Theorem 3. Let $L \in \mathsf{S}_2^{\mathrm{exp}}$ and V be a polynomial-time verifier associated with L that takes certificates of length $e(n) = 2^{n^{O(1)}}$ on inputs of length n. We claim that $L \in \mathsf{EXP}_{\|}^{R}$ for every dense subset R of random strings. Our $\mathsf{EXP}_{\|}^{R}$ algorithm M exhaustively searches "succinct witnesses", i.e., circuits that encode a correct certificate.

Specifically, for some polynomial $q(n)$ that is chosen later, let $\mathcal{C}_n$ be the set of all the $q(n)$-size $\log e(n)$-input oracle circuit. On input $x \in \{0,1\}^*$, the algorithm M exhaustively searches all the circuits $Y, Z \in \mathcal{C}_{|x|}$ and accepts if and only if there exists an oracle circuit $Y \in \mathcal{C}_{|x|}$ such that, for every oracle circuit $Z \in \mathcal{C}_{|x|}$, $V(x, \mathtt{tt}(Y^R), \mathtt{tt}(Z^R))$ accepts. Since the number $|\mathcal{C}_n|$ of polynomial-size oracle circuits is at most exponential in n, M halts in exponential time. Moreover, M is a nonadaptive reduction to R because the length of any query is bounded by some polynomial $q(n)$ (thus, in exponential time, one can ask every possible query beforehand).

We claim the correctness of M. By the definition of $\mathsf{S}_2^{\mathrm{exp}}$, for any input $x \in \{0,1\}^*$,

1. if $x \in L$, there exists a certificate $y \in \{0,1\}^{e(n)}$ such that $V(x,y,z) = 1$ for every $z \in \{0,1\}^{e(n)}$, and
2. if $x \notin L$, there exists a certificate $z \in \{0,1\}^{e(n)}$ such that $V(x,y,z) = 0$ for every $y \in \{0,1\}^{e(n)}$.

Consider the function η that maps (x,i) to the ith bit of the lexicographically first certificate $y \in \{0,1\}^{e(|x|)}$ such that $V(x,y,z) = 1$ for every $z \in \{0,1\}^{e(|x|)}$ (if exists), where $i \in \{0,1\}^{\log e(|x|)}$. Since η is computable, by Theorem 7, there exists a polynomial-size oracle circuit Y' such that $Y'^R(x,i) = \eta(x,i)$ for any (x,i). Therefore, for every input $x \in L$, by fixing x we obtain a polynomial-size circuit $Y_x := Y'(x,\text{-})$ such that $V(x, \mathtt{tt}(Y_x^R), z) = 1$ for every z. By choosing the bound $q(n)$ large enough, we obtain $Y_x \in \mathcal{C}$, and thus the algorithm M accepts. Similarly, one can prove that, for every input $x \notin L$, there exists a circuit $Z_x \in \mathcal{C}$ such that $V(x, y, \mathtt{tt}(Z_x^R)) = 0$ for every y. ◀

3.2 Upper Bound

Next, we prove the $\mathsf{S}_2^{\mathrm{exp}}$ upper bound on $\bigcap_R \mathsf{EXP}_{\|}^R$. Take an arbitrary language $L \in \bigcap_R \mathsf{EXP}_{\|}^R$. This means that, for every dense subset R of random strings, there exists an exponential-time oracle machine M_R such that M_R solves L given oracle access to R. That is, the reduction M_R is allowed to depend on the oracle R, which makes it hard to simulate the reduction. We first show, by using a diagonalization argument, that there exists a *single* machine that works for every oracle R.

▶ **Lemma 8.** *For any language L, the following are equivalent:*

1. *$L \in \bigcap_R \mathsf{EXP}_{\|}^R$, where the intersection is taken over all dense subsets R of R_{K_U}.*
2. *There exists an exponential-time nonadaptive oracle machine M such that, for every dense subset R of R_{K_U}, for every input $x \in \{0,1\}^*$, $M^R(x) = L(x)$.*

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 exponential-time nonadaptive oracle machine M, there exists a dense subset R of R_{K_U} such that $M^R(x) \neq L(x)$ for some $x \in \{0,1\}^*$. We claim that there exists a *single* dense subset $R \subseteq R_{\mathrm{K}_U}$ such that $L \notin \mathsf{EXP}_{\|}^R$.

To this end, let $\{M_e\}_{e\in\mathbb{N}}$ be the enumeration of all exponential-time nonadaptive oracle machines. We will construct some dense subset $R_e \subseteq R_{\mathrm{K}_U}$ and input x_e (and $\ell_e \in \mathbb{N}$) by induction on $e \in \mathbb{N}$, so that M_e fails to compute L on input x_e, given oracle access to R_{e+1}; 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 dense subset $R'_{e+1} \subseteq R_{\mathrm{K}_U}$ and some input $x_e \in \{0,1\}^*$ such that

- $M_e^{R'_{e+1}}(x_e) \neq L(x_e)$, 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 an exponential-time 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 dense subset $\hat{R}_{e+1} \subseteq R_{\mathrm{K}_U}$ such that $M_e'^{\hat{R}_{e+1}}(x_e) \neq L(x_e)$ for some $x_e \in \{0,1\}^*$; by the definition of M'_e, we obtain $M_e^{R'_{e+1}}(x_e) \neq L(x_e)$ 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 a dense subset of R_{K_U}. 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 is a dense subset of R_{K_U} by the construction above. By the choice of $(\ell_e)_{e\in\mathbb{N}}$, we have

$$M_e^R(x_e) = M_e^{R_{e+1}}(x_e) \neq L(x_e),$$

for every exponential-time nonadaptive oracle machine M_e. Thus $L \notin \mathsf{EXP}_{\|}^R$. ◀

We are ready to present the $\mathsf{S}_2^{\mathrm{exp}}$ algorithm that simulates an exponential-time nonadaptive reduction to dense subsets of random strings, which completes the proof of Theorem 3.

Proof of the Upper Bound of Theorem 3. Take any language $L \in \bigcap_R \mathsf{EXP}_{\|}^R$. By Lemma 8, there exists an exponential-time nonadaptive oracle machine M such that, for every dense subset R of R_{K_U}, for every input $x \in \{0,1\}^*$, $M^R(x) = L(x)$. Our goal is to simulate M in $\mathsf{S}_2^{\mathrm{exp}}$.

First, we can assume, without loss of generality, that the length of any query that M makes on input x is bounded by some polynomial $|x|^{O(1)}$. Indeed, since M is a nonadaptive reduction, the set of all the queries $Q(x)$ that M makes on input x can be computed in exponential time; thus, for each query $q \in Q(x)$, its Kolmogorov complexity $\mathrm{K}_U(q)$ is at most $|x| + \log |Q(x)| + O(1) \leq |x|^{O(1)}$, since q is described by x and the index of q in $Q(x)$. Therefore, for any query q whose length is larger than $|x|^{O(1)}$, the answer from the oracle R is "No" because $q \notin R_{\mathrm{K}_U}$ $(\supseteq R)$.

Now we present an $\mathsf{S}_2^{\mathrm{exp}}$ algorithm that simulate M. The idea is that two competing provers send the set of nonrandom strings. Given two possible sets of nonrandom strings $\bar{R}_1, \bar{R}_2 \subseteq \{0,1\}^*$ (one of which is guaranteed to be correct), $R := \{0,1\}^* \setminus (\bar{R}_0 \cup \bar{R}_1)$ is a subset of R_{K_U}. Moreover, since the number of nonrandom strings is small, the set R is dense enough. Details follow.

Let $p(n)$ be a polynomial that upper-bounds the length of any query that M makes on inputs of length n. Our $\mathsf{S}_2^{\mathrm{exp}}$ algorithm operates as follows: Fix any input x of length n. The ith prover ($i \in \{1,2\}$) sends, for each $\ell \leq p(n)$, a subset $\bar{R}_{i,\ell} \subseteq \{0,1\}^\ell$ of size at most $2^{\ell-2}$; an honest prover sets $\bar{R}_{i,\ell} := \{0,1\}^\ell \setminus R_{\mathrm{K}_U}$. Define $\bar{R}_i := \bigcup_{\ell \leq p(n)} \bar{R}_{i,\ell}$. Note that such subsets can be encoded as a string of exponential length. The verifier sets $R := \{0,1\}^* \setminus (\bar{R}_1 \cup \bar{R}_2)$, and accept if and only if $M^R(x)$ accepts.

We claim the correctness of the $\mathsf{S}_2^{\mathrm{exp}}$ algorithm. Assume that the i^*th prover is honest. For each length $\ell \leq p(n)$ and $i \in \{1,2\}$, we assumed that $|\bar{R}_{i,\ell}| \leq 2^{\ell-2}$; thus $|\bar{R}_{1,\ell} \cup \bar{R}_{2,\ell}| \leq 2^{\ell-1}$, from which it follows that R is dense. Moreover, since the number of nonrandom strings of length ℓ is at most $2^{\ell-2}$ (Fact 5)[6], the i^*th prover can set $\bar{R}_{i^*,\ell} := \{0,1\}^\ell \setminus R_{\mathrm{K}_U}$. Define $R' := \{\, q \in R \mid |q| \leq p(n) \,\} \cup \{\, q \in R_{\mathrm{K}_U} \mid |q| > p(n) \,\}$. Then, from the argument above, R' is a dense subset of R_{K_U}, and hence $M^{R'}(x) = L(x)$. Since M does not make any query of length $> p(n)$, it follows that $M^R(x) = M^{R'}(x) = L(x)$. ◀

In Appendix A, we will mention that the reducibility notion of Theorem 3 can be significantly improved from $\mathsf{EXP}_{\|}$ to $\mathsf{S}_2^{\mathrm{p}}$ (at the cost of a slight loss in the lower bound).

4 The Unexpected 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.

▶ **Reminder of Theorem 4.** For every universal Turing machine U,

$$\mathsf{EXPH} \subseteq \mathsf{PH}^{R_{\mathrm{K}_U}}.$$

For the purpose of refuting Allender's conjecture (Conjecture 2), it is enough to show the weaker statement that $\mathsf{EXPH} \subseteq \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$. Indeed, we observe that a simple padding argument essentially refutes Conjecture 2:

▶ **Corollary 9.** $\mathsf{BPP} \neq \bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$ *unless* $\mathsf{EXPH} \subseteq \mathsf{BPEXP}$*, where the intersection is taken over all universal Turing machines.*

Proof. Assume that $\bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}} \subseteq \mathsf{BPP}$. Then by the standard padding argument, we also obtain $\bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}} \subseteq \mathsf{BPEXP}$. (Specifically, take any language $L \in \bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$, and define a padded version $L' := \{\, x 10^{2^{p(|x|)}} \mid x \in L \,\}$ for some large enough polynomial p so that $L' \in \bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$. By the assumption, we have $L' \in \mathsf{BPP}$, which implies that $L \in \mathsf{BPEXP}$.) It follows from Theorem 4 that $\mathsf{EXPH} \subseteq \bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}} \subseteq \mathsf{BPEXP}$. ◀

[6] This is because of our choice of the threshold for Kolmogorov-randomness.

Now we proceed to a proof of Theorem 4. We first observe that any language computable with oracle access to HALT is in $\mathsf{P}^{\mathsf{HALT}}$.

▶ **Lemma 10.** *Let M be any oracle machine that, on inputs of length n, halts in finite steps and makes a query of length at most $n^{O(1)}$. Then the language decided by M^{HALT} is in $\mathsf{P}^{\mathsf{HALT}}$.*

Proof Sketch. The proof is essentially the same with [5, Theorem 27], and thus we just give a proof sketch. The idea is to decide M^{HALT} in the following two steps: First, by using a binary search and oracle access to HALT, one can decide the number of all the strings in HALT of length at most $n^{O(1)}$ in polynomial time. Then, given as advice the number of strings in HALT of length at most $n^{O(1)}$, the computation of M^{HALT} becomes now computable, and hence it reduces to HALT. ◀

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 [21]. 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 11** (Kiwi, Lund, Spielman, Russell, and Sundaram [21])**.** *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 4.

Proof of Theorem 4. The main idea is that, given oracle access to the set of random strings, Theorem 7 tells us that there is a "succinct witness" for any exponential-time computation. However, unlike the proof of Theorem 3, here we need to claim that there exists a succinct witness that encodes some winning strategy of two player games, which may depend on a R_{K_U}-oracle circuit that encodes the opponent's strategy. For simplicity, we will only give a detailed proof for $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{PH}^{R_{\mathrm{K}_U}}$, since it is straightforward to extend the proof.

First, we present a proof of $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$. Let V be a polynomial-time verifier for $L \in \Sigma_2^{\mathsf{EXP}}$, and $e(n) = 2^{n^{O(1)}}$ be an exponential bound 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 $e(n)$ such that $V(x, y, z)$ accepts for all z of length $e(n)$; Note that V runs in time $2^{n^{O(1)}}$. We regard the computation as a game between the first player y and the second player z.

Our $\mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$ algorithm operates as follows: Let $s_Y(n)$ and $s_Z(n)$ be some polynomials specified later. Given input x of length n, the algorithm accepts if and only if there exists an oracle circuit Y of size $s_Y(n)$ such that $V(x, \mathtt{tt}(Y^{R_{\mathrm{K}_U}}), \mathtt{tt}(Z^{R_{\mathrm{K}_U}}))$ accepts for all oracle circuits Z of size $s_Z(n)$, where $\mathtt{tt}(Y^{R_{\mathrm{K}_U}})$ denotes the truth table of the function computed by $Y^{R_{\mathrm{K}_U}}$; 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_{K_U}: Indeed, let y_x be the lexicographically first 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 7, there exists an oracle circuit Y of size $s_Y(n) := \mathsf{poly}(n, \log |y_x|) = \mathsf{poly}(n)$ such that $\mathtt{tt}(Y^{R_{\mathrm{K}_U}}) = y_x$. Thus $V(x, y_x, \mathtt{tt}(Z^{R_{\mathrm{K}_U}}))$ accepts no matter how the adversarial circuit Z 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 Y of size $s_Y(n)$, there exists a circuit Z that encodes a strategy that beats the strategy of $\mathtt{tt}(Y^{R_{\mathrm{K}^U}})$. The point is that, given any circuit Y, the lexicographically first winning strategy of the second player is computable with oracle access to HALT. Indeed, let $z_{x,Y}$ denote the lexicographically first string such that $V(x, \mathtt{tt}(Y^{R_{\mathrm{K}^U}}), z)$ rejects. Consider the language $L' := \{ (x, Y, i) \mid \text{the } i\text{th bit of } z_{x,Y} \text{ is } 1 \}$. Since R_{K^U} is reducible to HALT, the language L' is computable with oracle access to HALT. By Lemma 10, $L' \in \mathsf{P}^{\mathsf{HALT}}$; thus by Theorem 7, we obtain $L' \in \mathsf{P}^{R_{\mathrm{K}^U}}/\mathsf{poly}$. This means that for every circuit Y there exists a circuit Z_Y of size $s_Z(n) := \mathsf{poly}(n, |Y|, \log |z_{x,Y}|) = \mathsf{poly}(n, s_Y(n))$ such that $\mathtt{tt}(Z_Y^{R_{\mathrm{K}^U}}) = z_{x,Y}$. Thus, our algorithm rejects. This completes the proof of $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{EXP}_{\|}^{R_{\mathrm{K}^U}}$.

In order to extend the proof above to $\Sigma_{2k}^{\mathsf{EXP}}$ for every constant k, we modify the $\mathsf{EXP}^{R_{\mathrm{K}^U}}$ algorithm so that it checks, given input x of length n, whether $\exists$ 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 $V(x, \mathtt{tt}(C_1^{R_{\mathrm{K}^U}}), \cdots, \mathtt{tt}(C_{2k}^{R_{\mathrm{K}^U}}))$ 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 $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{PH}^{R_{\mathrm{K}^U}}$. Note that, in the proof above, the bottleneck of the computation is the evaluation of $V(x, \mathtt{tt}(Y^{R_{\mathrm{K}^U}}), \mathtt{tt}(Z^{R_{\mathrm{K}^U}}))$, where V runs in time $2^{|x|^{O(1)}}$. We replace V with the randomized polynomial-time verifier of Theorem 11; then we obtain the following $\Sigma_2^{R_{\mathrm{K}^U}}$ algorithm: Existentially guess a circuit Y of size at most $s_Y(n)$, and universally guess a circuit Z of size at most $s_Z(n)$ as well as a coin flip for V, Then accept if and only if $V^{Y,Z}(x)$ accepts on the guessed coin flip sequence. ◀

References

1. Eric Allender. Curiouser and Curiouser: The Link between Incompressibility and Complexity. In *Proceedings of the 8th Conference on Computability in Europe (CiE)*, pages 11–16, 2012. doi:10.1007/978-3-642-30870-3_2.
2. 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. doi:10.1007/978-3-319-50062-1_6.
3. 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. doi:10.2168/LMCS-10(3:5)2014.
4. 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. doi:10.1016/j.apal.2005.06.003.
5. Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from Random Strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. doi:10.1137/050628994.
6. 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. URL: http://cjtcs.cs.uchicago.edu/articles/2013/5/contents.html.
7. Eric Allender, Luke Friedman, and William I. Gasarch. Limits on the computational power of random strings. *Inf. Comput.*, 222:80–92, 2013. doi:10.1016/j.ic.2011.09.008.
8. Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. *TOCT*, 11(4):27:1–27:27, 2019. doi:10.1145/3349616.

9 Eric Allender, Dhiraj Holden, and Valentine Kabanets. The Minimum Oracle Circuit Size Problem. *Computational Complexity*, 26(2):469–496, 2017. doi:10.1007/s00037-016-0124-0.

10 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. doi:10.1109/CCC.2010.15.

11 Jin-yi Cai. $\mathsf{S}_2^p \subseteq \mathsf{ZPP}^{\mathsf{NP}}$. *J. Comput. Syst. Sci.*, 73(1):25–35, 2007. doi:10.1016/j.jcss.2003.07.015.

12 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. doi:10.1016/j.ic.2005.01.002.

13 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. doi:10.2168/LMCS-10(3:15)2014.

14 Ran Canetti. More on BPP and the Polynomial-Time Hierarchy. *Inf. Process. Lett.*, 57(5):237–241, 1996. doi:10.1016/0020-0190(96)00016-6.

15 Shuichi Hirahara. Identifying an Honest EXP^NP Oracle Among Many. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 244–263, 2015. doi:10.4230/LIPIcs.CCC.2015.244.

16 Shuichi Hirahara and Akitoshi Kawamura. On characterizations of randomized computation using plain Kolmogorov complexity. *Computability*, 7(1):45–56, 2018. doi:10.3233/COM-170075.

17 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. doi:10.4230/LIPIcs.CCC.2016.18.

18 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. doi:10.4230/LIPIcs.FSTTCS.2015.236.

19 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. doi:10.1145/258533.258590.

20 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 73–79, 2000. doi:10.1145/335305.335314.

21 Marcos A. Kiwi, Carsten Lund, Daniel A. Spielman, Alexander Russell, and Ravi Sundaram. Alternation in interaction. *Computational Complexity*, 9(3-4):202–246, 2000. doi:10.1007/PL00001607.

22 Cody D. Murray and R. Ryan Williams. On the (Non) NP-Hardness of Computing Circuit Complexity. *Theory of Computing*, 13(1):1–22, 2017. doi:10.4086/toc.2017.v013a004.

23 Alexander Russell and Ravi Sundaram. Symmetric Alternation Captures BPP. *Computational Complexity*, 7(2):152–162, 1998. doi:10.1007/s000370050007.

A Improving the Complexity of the Reductions

In this appendix, we mention that the reducibility notion of Theorem 3 can be significantly improved by using the efficient proof system of [15]:

▶ **Theorem 12.** *For any universal Turing machine U,*

$$\mathsf{EXP}^{\mathsf{NP}} \subseteq \bigcap_R \mathsf{S}_2^R \subseteq \mathsf{S}_2^{\exp},$$

where the intersection is taken over all the dense subsets R of R_{K_U}.

Hirahara [15] introduced the notion of a selector, and showed that $\mathsf{EXP}^{\mathsf{NP}}$ admits a selector.

▶ **Lemma 13** ([15]). *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 $\Pr_S\left[S^{A_0,A_1}(x) = L(x)\right] \geq \frac{2}{3}$.*

We show that any language L with a selector is low for $\mathsf{S}_2^{\mathrm{p}}$ if $L \in \mathsf{P/poly}$.

▶ **Theorem 14.** *Let L be a language with a selector S and R be any oracle. Then,*

$$L \in \mathsf{P}^R/\mathsf{poly} \quad \Longrightarrow \quad \mathsf{S}_2^L \subseteq \mathsf{S}_2^R.$$

This generalizes a lowness result of [12] from any downward self-reducible language L to any language L with a selector.

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

Proof of Theorem 12. Under any dense subset R of Kolmogorov-random strings, we have $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{P}^R/\mathsf{poly}$ (by Theorem 7). 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 13 and Theorem 14. ◀

Finally, we mention that in the case of reductions to the set of random strings, the $\mathsf{S}_2^{\mathrm{p}}$ reductions of Theorem 12 can be derandomized to obtain P^{NP} reductions.

▶ **Theorem 15.** $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{P}^{\mathsf{NP}^{R_{\mathrm{K}_U}}}$ *for any universal Turing machine U.*

Proof. By Theorem 12, we immediately obtain $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{S}_2^{R_{\mathrm{K}_U}}$. By the relativized version of Cai's result [11], 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 10 and Theorem 7, 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 [19], one can derandomize the computation of ZPP under an $\mathsf{NP}^{R_{\mathrm{K}_U}}$ oracle. ◀