On Randomized Reductions to the Random Strings

Michael Saks
Department of Mathematics, Rutgers, The State University of New Jersey, Piscataway, NJ, USA

Rahul Santhanam
Department of Computer Science, University of Oxford, UK

Abstract

We study the power of randomized polynomial-time non-adaptive reductions to the problem of approximating Kolmogorov complexity and its polynomial-time bounded variants.

As our first main result, we give a sharp dichotomy for randomized non-adaptive reducibility to approximating Kolmogorov complexity. We show that any computable language L that has a randomized polynomial-time non-adaptive reduction (satisfying a natural honesty condition) to $\omega(\log(n))$-approximating the Kolmogorov complexity is in $\mathsf{AM} \cap \mathsf{coAM}$. On the other hand, using results of Hirahara [28], it follows that every language in NEXP has a randomized polynomial-time non-adaptive reduction (satisfying the same honesty condition as before) to $O(\log(n))$-approximating the Kolmogorov complexity.

As our second main result, we give the first negative evidence against the NP-hardness of polynomial-time bounded Kolmogorov complexity with respect to randomized reductions. We show that for every polynomial t', there is a polynomial t such that if there is a randomized time t' non-adaptive reduction (satisfying a natural honesty condition) from SAT to $\omega(\log(n))$-approximating K^t complexity, then either $\mathsf{NE} = \mathsf{coNE}$ or E has sub-exponential size non-deterministic circuits infinitely often.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Kolmogorov complexity, randomized reductions

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.29

Funding *Michael Saks*: Supported in part by the Simons Foundation under Grant 332622.
Rahul Santhanam: Partially funded by EPSRC New Horizons Grant EP/V048201/1.

Acknowledgements We thank the anonymous reviewers for useful comments. The second author benefited from conversations with Shuichi Hirahara and Rahul Ilango.

1 Introduction

Meta-complexity studies the complexity of computational problems that are themselves about complexity, such as the Minimum Circuit Size Problem MCSP which asks if a Boolean function represented by its truth table has circuits of a given size, and the problem K^{poly} of determining the polynomial-time bounded Kolmogorov complexity of a string. One of the main open questions in meta-complexity is: Are MCSP and related problems such as K^{poly} NP-hard? This is a question with a long history; indeed, Levin is reported [15] to have delayed publication of his seminal NP-completeness results [40] because he hoped to show that MCSP was NP-complete. More than 50 years on, the question remains unresolved, despite much effort [41, 12, 29, 34, 36, 35], and MCSP is one of the few remaining natural problems in NP for which there is no clear evidence either of NP-hardness or of non-NP-hardness.

Why is it so difficult to show MCSP is NP-hard? This has been investigated in a series of works [37, 43, 33, 30, 14, 13, 44]. The theme of this line of works is that hardness of MCSP under various kinds of deterministic reducibility is related to our ability to prove longstanding complexity conjectures. The basic idea is simple: since we can generate negative instances of

SAT efficiently, an efficient honest reduction from SAT to MCSP would allow us to generate negative instances of MCSP efficiently, which would enable us to prove circuit lower bounds for EXP. There are many refinements of this idea in the aforementioned series of works, and in particular [42] unconditionally rule out hardness of MCSP under polylogarithmic-time projections, a class of reductions that is sufficient to show hardness for all known natural NP-complete problems.

Note that the difficulty identified above relies on the *determinism* of the reductions. Indeed, truth tables of hard Boolean functions are easy to generate for randomized algorithms - a uniformly random string is likely to be hard! Therefore, previous works shed no light on whether MCSP and related problems can be shown to be hard under *randomized* reductions. This question is at the heart of our work.

▶ **Question 1.** *How powerful are randomized reductions to meta-complexity problems?*

We note that several works showing hardness results for meta-complexity problems [6, 8, 11, 34, 36, 28] do employ randomness in their reductions, which is a further reason to consider Question 1. As far as we are aware, there is no evidence in the literature against solving all of NP with efficient randomized reductions to meta-complexity problems such as MCSP [1], K^{poly} or even the problem of computing Kolmogorov complexity, even for the case of many-one reductions.

In fact, the hardness results for meta-complexity problems mentioned above [6, 8, 11, 34, 36, 28] are even *robust* to close approximations of the complexity measure of interest, in the sense that the reduction continues to work even if the oracle only knows a close approximation to the complexity measure rather than an exact value. This motivates us to consider Question 1 more broadly in terms of *robustness to approximation* for randomized reductions. It is known, for instance [37, 6], that if one-way functions exist, then circuit size of a Boolean function with truth table size N is even hard to approximate to within an $N^{1-\epsilon}$ factor for any $\epsilon > 0$. One might hope that such strong inapproximability results continue to hold under the weaker assumption that $\mathsf{NP} \not\subseteq \mathsf{BPP}$. Moreover, an intriguing recent line of work [22, 27, 45] on worst-case to average-case reductions for meta-complexity problems seem to *require* an inapproximability assumption to infer a conclusion on average-case hardness. So, for example, if our goal is to show that NP has worst-case to average-case reductions using these results, we need to show NP-hardness of *approximating* circuit size or K^{poly} complexity.

1.1 Our Results

Our two main results address Question 1 by showing that efficient randomized non-adaptive reductions to the meta-complexity problems of additively approximating the Kolmogorov complexity and additively approximating the polynomial-time bounded Kolmogorov complexity are surprisingly weak, even when the approximation term is super-logarithmic[2]. We first state the results informally, and then describe them in more detail.

[1] As pointed out by an anonymous reviewer, the results of [43] *unconditionally* rule out NP-hardness of MCSP under a very special form of randomized reduction: *projections* computable in *sublinear* time. The techniques of [43] do not seem immediately applicable to reductions that are not projections, or where the bits of the output take at least linear time to compute.

[2] Note that a smaller approximation term makes a negative result stronger.

▶ **Theorem 1** (Informal). *If there is a "nice" randomized non-adaptive poly-time reduction from L to approximating the Kolmogorov complexity within some super-logarithmic additive term, then $L \in \mathsf{AM} \cap \mathsf{coAM}$.*

▶ **Theorem 2** (Informal). *Under plausible complexity assumptions, if there is a "nice" randomized non-adaptive fixed poly-time reduction from L to approximating the K^{poly} complexity within some super-logarithmic additive term, then $L \in \mathsf{AM} \cap \mathsf{coAM}$.*

Here "nice" has a fairly standard meaning: the reduction is required to be polynomially honest, meaning that it doesn't make queries which are too small. In the statement of Theorem 2, "fixed poly-time" means that the reduction runs within a time bound that is independent of the polynomial time bound used in the definition of K^{poly}. As direct corollaries of Theorem 1 and Theorem 2, we get that under plausible complexity assumptions, neither additively approximating the Kolmogorov complexity to a super-logarithmic term nor additively approximating the polynomial-time bounded Kolmogorov complexity to a super-logarithmic term is NP-hard under efficient randomized reductions.

All our results in this paper have to do with non-adaptive reductions - analysing adaptive reductions is an interesting open question. Indeed, it is not even known whether hardness of MCSP under general *deterministic* adaptive reductions leads to any new complexity lower bounds, or has any other surprising consequences[3].

1.1.1 Randomized Reductions to Approximating Kolmogorov Complexity

As a first step, we consider Question 1 when the meta-complexity problem is to approximate Kolmogorov complexity. Several known reductions to meta-complexity problems such as MCSP [6, 8, 11, 34, 36], especially those that exploit ideas from pseudorandomness, continue to work when the meta-complexity oracle is substituted with a Kolmogorov complexity oracle. Moreover, the power of reductions to Kolmogorov complexity has been studied extensively in its own right [7, 5, 21, 32, 10, 9, 4, 28]. Question 4.8 of Allender's survey [2] explicitly asks about the power of reductions to approximating Kolmogorov complexity.

When considering reductions to a Kolmogorov complexity[4] oracle, there is an ambiguity: Kolmogorov complexity K_U is defined with respect to some universal Turing machine U. In several works [5, 21, 10, 9, 4], the following approach has been taken to resolve the ambiguity. A language L is said to be reducible to Kolmogorov complexity if L is reducible to K_U for *every* universal machine U. In this paper, we take a different approach, as proposed by Allender [2]: we consider reductions to *additively approximating* Kolmogorov complexity rather than reductions to exact Kolmogorov complexity. In practice, reductions rarely take advantage of the exact Kolmogorov complexity of a queried string, so this doesn't lose us too much in terms of modelling known reductions. Moreover, by varying the approximation parameter, we can ensure robustness with respect to the universal machine, and even with respect to the precise notion of Kolmogorov complexity. For instance, reducibility to $\omega(1)$-additively approximating Kolmogorov complexity is robust to the universal machine, using

[3] Some partial results in this direction appear in [44]. Analogous questions about the power of adaptivity in the setting of worst-case to average-case reductions [23, 20] have remained open despite significant effort.

[4] Most works in the literature consider the Kolmogorov random strings as oracle, or else the *overgraph* for Kolmogorov complexity [3]. Without loss of generality, we instead consider the *functional* oracle that when asked a query q, returns the Kolmogorov complexity of q. Note that the Kolmogorov random strings and the overgraph are both reducible to the functional oracle why simple deterministic non-adaptive reductions.

the fact that for any two universal machines U and U', K_U and $K_{U'}$ are at most a constant apart. Reducibility to $O(\log(n))$-additively approximating Kolmogorov complexity is robust to whether we consider the standard version of Kolmogorov complexity or the prefix free version[5], using the fact that for any string, the Kolmogorov complexity and prefix-free Kolmogorov complexity are at most a logarithmic term apart.

Intuitively, a deterministic non-adaptive reduction cannot make use of the Kolmogorov randomness oracle in an effective way, because queries generated by the reduction on compressible inputs have low Kolmogorov complexity. However, if the reduction is allowed to be *randomized*, this limitation disappears, because the reduction can take advantage of randomness to produce queries of high complexity. Hirahara [28] recently showed that every language in NEXP (non-deterministic exponential time) is randomized polynomial-time non-adaptively reducible to the Kolmogorov random strings. This implies a strong positive answer to Question 1 when the meta-complexity problem is to approximate Kolmogorov complexity, and the approximation gap is *small*.

We show, somewhat counter-intuitively, that randomized polynomial-time non-adaptive reductions to *approximating* the Kolmogorov complexity are inherently limited when the approximation gap is $\omega(\log(n))$. Indeed, languages with such reductions (satisfying a natural honesty condition) are in $\mathsf{AM} \cap \mathsf{coAM}$. As far as we are aware, the previous best complexity upper bound known for languages that are randomized polynomial-time non-adaptively reducible to $\omega(\log(n))$-approximating the Kolmogorov complexity was EXPSPACE [10].

Indeed, we get a sharp *complexity dichotomy* based on the approximation gap, by combining our results with those of [28]:

▶ **Theorem 3.** *If a language L is computable and there is a randomized polynomial-time non-adaptive reduction F from L to $\omega(\log(n))$-additively approximating Kolmogorov complexity such that F is polynomially honest and has inverse polynomial advantage, then $L \in \mathsf{AM} \cap \mathsf{coAM}$. On the other hand, every $L \in \mathsf{NEXP}$ has a randomized polynomial-time non-adaptive reduction making q queries to $O(\log(n))$-additively approximating Kolmogorov complexity that is polynomially honest and has constant advantage.*

Theorem 3 is a more precise version of Theorem 1.

A polynomially honest reduction is one where any query on input x is of size $|x|^{\Omega(1)}$. We do have a more general result that is operative even when queries are only guaranteed to have super-constant size, but we adopt the formulation of Theorem 3 here because it is more easily stated. The honesty condition is a very natural one that is satisfied in most reductions of which we are aware. It appears for technical reasons, which we describe in Section 1.2. The *advantage* of a reduction is ϵ if it is correct with probability at least $1/2 + \epsilon$; thus the condition on advantage in the statement of Theorem 3 is the mildest reasonable one.

Our result yields an interesting phenomenon in terms of the tradeoff between *adaptivity* of the reduction and the *robustness to approximation* of the reduction. As mentioned before, it is shown in [28] that approximating Kolmogorov complexity when the approximation term is sufficiently small, i.e., logarithmically bounded, is hard for NEXP under randomized non-adaptive reductions. When the randomized reduction is allowed to be adaptive, it follows from work of [6] that approximating Kolmogorov complexity, even for large *multiplicative* approximation gap $|q|^{1-\epsilon}$ for any $\epsilon > 0$, is hard for PSPACE. Theorem 3 shows that we

[5] The prefix free version of Kolmogorov complexity considers only prefix-free Turing machines, i.e., Turing machines M such that if M halts on x, M does not halt on xy for any non-empty y. This is more natural in certain contexts.

cannot get a hardness result that combines the best of both worlds: if we require the reduction to be non-adaptive, and in addition the approximation gap is large, then the power of the reduction decreases significantly.

Theorem 3 has an application to *non-adaptive black-box* constructions of hitting set generators from worst-case hardness assumptions. Recall that a hitting set generator with seed length $s(n) < n$ is an efficiently computable function from $s(n)$ bits to n bits such that the range of the generator hits every dense set that is computable by polynomial-size circuits. Motivated by connections to uniform hardness-randomness tradeoffs, [26] showed that if there is a randomized non-adaptive reduction from a language L to avoiding the range of an exponential-time computable hitting set generator with seed length $< n/4$, then $L \in \mathsf{BPP}^{\mathsf{NP}}$. Recently, [31] improved this by showing that if there is a randomized non-adaptive reduction from a language L to avoiding the range of an exponential-time computable hitting set generator with seed length $(1-\epsilon)n$ for some $\epsilon > 0$, then $L \in \mathsf{AM} \cap \mathsf{coAM}$. Since a Kolmogorov complexity oracle can be used to avoid the range of any computable hitting set generator (by using the fact that outputs of the hitting set generator have non-trivial Kolmogorov complexity), we get the following immediate corollary:

▶ **Corollary 4.** *If a language L is computable and there is a randomized polynomial-time non-adaptive reduction F from L to avoiding the range of a computable hitting set generator with seed length $n - \omega(\log(n))$ such that F is polynomially honest, then $L \in \mathsf{AM} \cap \mathsf{coAM}$.*

Corollary 4 is not directly comparable to the main result of [31] because of the honesty condition, but modulo this condition, it shows an *optimal* limitation on reductions to avoiding the range of hitting set generators in terms of the seed length. We note that [28] shows that for every language $L \in \mathsf{NEXP}$, L randomized polynomial-time non-adaptively reduces to avoiding the range of a $\mathsf{EXP}^{\mathsf{NP}}$-computable hitting set generator with seed length $n - O(\log(n))$. Under standard derandomization assumptions [38], $\mathsf{AM} \cap \mathsf{coAM} = \mathsf{NP} \cap \mathsf{coNP}$, which is strictly contained in NEXP.

As discussed in [31], close connections between hitting-set generators and average-case hardness mean that Corollary 4 also relates to a long line of work ruling out non-adaptive black-box worst-case to average-case reductions for NP [23, 20, 1].

1.1.2 Randomized Reductions to Approximating Polynomial-Time Bounded Kolmogorov Complexity

The main application of Theorem 3 is to the question of NP-hardness of *meta-complexity* problems. As an immediate consequence of Theorem 3, even computing an approximation of Kolmogorov complexity with small gap is not NP-hard under (honest) randomized non-adaptive reductions, unless there is a surprising complexity collapse.

▶ **Corollary 5.** *Suppose there is a randomized polynomial-time non-adaptive reduction from SAT to $\omega(\log(n))$-additively approximating Kolmogorov complexity that is polynomially honest. Then the Polynomial Hierarchy collapses.*

Corollary 5 follows from Theorem 3 by using the fact that if SAT is in $\mathsf{AM} \cap \mathsf{coAM}$, then the Polynomial Hierarchy collapses [46].

As far as we are aware, Corollary 5 gives the first negative implication of the assumption that approximating Kolmogorov complexity is NP-hard under randomized non-adaptive reductions. Several recent works on hardness of meta-complexity problems [34, 36, 35] are only able to handle small approximation gap - Corollary 5 provides one possible explanation of this (in cases when the reduction is not refined enough to distinguish between Kolmogorov complexity and other complexity measures).

Corollary 5 does not immediately imply that NP-hardness of K^{poly} under randomized reductions is unlikely, as there is no known efficient non-adaptive reduction from K^{poly} to Kolmogorov complexity. By working quite a bit harder, we are able to show the following result.

▶ **Theorem 6.** *Suppose that there is a polynomially bounded function $t' : \mathbb{N} \to \mathbb{N}$ such that for all large enough $t = \text{poly}(t')$, there is a randomized time t' non-adaptive reduction that is polynomially honest, has fixed query length and inverse polynomial advantage, from SAT to $\omega(\log(m))$-additively approximating K^t complexity. Then either* E *has non-deterministic circuits of size $2^{o(n)}$ infinitely often, or* NE = coNE.

Theorem 6 is a more precise version of Theorem 2.

This appears to be the first negative evidence against NP-completeness of a meta-complexity problem in NP with respect to *randomized* non-adaptive reductions in the unrelativized setting[6]. Note that previous results on consequences of hardness with respect to deterministic reductions of MCSP do not suggest that such reductions are *unlikely*, but rather that they are *hard to show*. In contrast, the consequent of Theorem 6 would be considered unlikely by many complexity theorists.

A subtlety in the statement of Theorem 6 is that the negative evidence depends on the running time of the reduction being smaller than the time bound for Kolmogorov complexity. Typically, reductions from SAT to NP-complete L run in quasi-linear time. Theorem 6 suggests that the picture is very different for K^{poly} - if it is indeed NP-complete under randomized polynomial-time reductions, then the reductions are likely to need a lot of time.

1.2 Proof Techniques

We sketch the ideas behind the proofs of our main results: Theorem 3 and Theorem 6. We first introduce some notation, as per Section 2. Given a function $\beta : \mathbb{N} \to \mathbb{N}$, we say that two functional oracles $O : \Sigma^* \to \mathbb{N}$ and $O' : \Sigma^* \to \mathbb{N}$ are β-close if $|O(q) - O'(q)| \geq \beta(|q|)$ for every string q. In the following, we use β-closeness with $\beta(n) = \omega(\log(n))$.

Suppose M is an oracle machine implementing the reduction hypothesized in Theorem 3. At a high level, we'd like to construct an explicit computable oracle K' such that :

1. K' is β-close to K
2. There is an Arthur-Merlin protocol that given a string q and integer a to $K'(q)$, allows Merlin to prove to Arthur that a is "close" to $K'(q)$.

Suppose we have such an oracle K'. The hypothesis that M is a non-adaptive reduction to β-approximating Kolmogorov complexity with inverse polynomial advantage implies that $M^{K'}$ computes L with advantage $\epsilon = 1/n^{O(1)}$. We then hope to use the second condition above to show that it is possible to give an AM protocol that allows Merlin to give a proof of the output value of $M^{K'}(x)$ by having Merlin provide answers to all of the queries and then prove them to Arthur.

The proof roughly follows this strategy, but there are several obstacles that require modification. We don't know how to construct the desired oracle K'. Instead we consider a generalized notion of oracle called a *context-sensitive* oracle. This is an oracle O whose

[6] A result of Ko [39] states there is no relativizing NP-hardness proof for K^{poly}, but this seems to say little about the unrelativized case. Also, a result of Hirahara and Watanabe [30] shows that 1-query randomized reductions to MCSP that are "oracle independent" only exist for languages in AM ∩ coAM. We do not need the "oracle indepedent" restriction and our results hold for any number of queries, as long as they are non-adaptive.

answer to a query q can depend not just on q but also on the "context" within which q is generated, i.e., the oracle machine M making the query and the input x on which M makes the query. In what follows we use oracle to mean a possibly context-sensitive oracle and refer to an ordinary oracle as *context-insensitive*.

We will define a context-sensitive oracle which we call J that will be used in place of K' in the above outline. Here is a rough description of J. Fix the machine M and input x. Suppose that the machine M makes k queries to K, where the choice of queries is a function of the auxiliary random string ρ. For $i \in \{1, \ldots, k\}$, let $q_i(\rho)$ denote the ith query when the random string is ρ. Let $\pi(q)$ denote the probability (over ρ and random $i \in \{1, \ldots, k\}$) that $q_i(\rho) = q$. The context-sensitive oracle $J_x^M(q)$ is the ceiling of $\log 1/\pi(q)$. It is not hard to show that, for fixed M and x, $J(q)$ is β-close to $K(q)$ for all but a small fraction of q.

We want to show that (a) M^J computes the same language as M^K, and (b) the language computed by M^J is in $\mathsf{AM} \cap \mathsf{coAM}$. We can't show either of these, but we are able to state and prove small modifications to these assertions that suffice to give the desired conclusion.

We would like that M^J computes the same language as M^K. By hypothesis $L = M^{K'}$ for every *context-insensitive* oracle K' that is a β-approximation to K with advantage ϵ, but J is context-sensitive. Nevertheless, we are able to show that for all sufficiently long inputs x, the probability (with respect to the randomness of M) that $M^J(x) \neq M^K(x)$ is at most $\epsilon/2$, and since M^K computes L with advantage ϵ, we conclude that M^J computes L on all sufficiently long inputs with advantage at least $\epsilon/2$. To do this we fix a "sufficiently long" input x. We describe how to construct a context insensitive oracle K' (depending on x) that satisfies (i) if M^J and $M^{K'}$ are run on x then with probability at least $1 - \epsilon/2$ over the auxiliary random string ρ, J and K give the same answers on all of the queries $q_1(\rho), \ldots, q_k(\rho)$ and therefore the probability that M^J and $M^{K'}$ give different output on x is at most $\epsilon/2$, and (ii) K' is β-close to K, which implies that $M^{K'}$ computes L with advantage ϵ. Therefore $M^J(x)$ gives the correct output with advantage $\epsilon/2$.

We conclude that M^K and M^J compute the same language except for finitely many strings. We can modify M^J to fix those strings.

For the second part of the proof we aim to show that one can simulate the computation of M^J in $\mathsf{AM} \cap \mathsf{coAM}$. The J oracle can be viewed as solving an approximate counting problem: For a given query q, how many pairs (ρ, i) have $q_i(\rho) = q$, and such counting problems can be approximated in $\mathsf{AM} \cap \mathsf{coAM}$ [25, 20].

However, there is a major difficulty in translating this intuition into a simulation of M^J in $\mathsf{AM} \cap \mathsf{coAM}$. We are not guaranteed that the Merlin-Arthur protocol provides *consistent* answers that depend only on x and M. In order to get around this diffculty, our main idea is to use the following perturbation argument.

We define a family of oracles $J[\gamma]$ for $\gamma \in [0, 1)$. $J[\gamma]$ is a shifted version of J; it is the ceiling of the logarithm of $\log(1/(1+\gamma)\pi(x))$. We show that given M, there is a γ (in fact a random γ works with high probability) such that $M^{J[\gamma]}$ can be computed in $\mathsf{AM} \cap \mathsf{coAM}$.

In order to use this we show a stronger version of the first step: for all sufficiently long inputs x, for all $\gamma \in [0, 1)$, the probability that $M^J[\gamma]$ disagrees with M^K is at most $\epsilon/2$.

The Arthur-Merlin protocol we use bears significant similarities to the ideas of [23, 20, 31], however we need to work a bit harder to accommodate the perturbation argument.

Why does the honesty condition comes into our results? Our argument works by contradiction and involves using the fact that the first x on which the simulation fails has low Kolmogorov complexity. But this complexity bound is in terms of $|x|$, while the approximation condition on the Kolmogorov complexity oracle is in terms of the query length $|q|$. Thus it is important that there is some way of connecting $|x|$ and $|q|$ when q is a query

asked on x, and this is what the honesty condition guarantees. In fact our more general Theorem 14 is applicable even when the reduction is not polynomially honest, but it does require that query length is at least super-constant.

We now proceed to sketch the ideas behind Theorem 6. A natural idea is to try to *derandomize* the reduction using a standard derandomization hypothesis, and then argue that efficient deterministic non-adaptive reductions are unlikely. However, this doesn't work for the following reason: in order to derandomize the reduction using a standard derandomization hypothesis, we need to fool the test that checks, given fixed x and random string r, that the input x is consistent with the output $f(x, r)$ of the reduction. This test involves running machines both for the langage from which we are reducing and the language to which we are reducing, and a naive implementation takes at least non-deterministic time t (to simulate K^t). Hence the derandomized reduction will run in time $\text{poly}(t)$, which is greater than the time bound for Kolmogorov complexity, and we do not know how to argue against this possibility.

Instead, we use Theorem 3. The idea is to show that, under a standard derandomization hypothesis, oracle access to K^t can be replaced by oracle access to K, without affecting the correctness of the reduction. We prove a new lemma stating that for with high probability over samples from a distribution D samplable in time t', the K^t complexity of the sample q is at most $O(\log(|q|))$ apart from the K complexity. This lemma uses the derandomization hypothesis that E requires exponential-size non-deterministic circuits, and the time bound t is polynomially bounded in t' once the parameters of the derandomization hypothesis are fixed. The derandomization hypothesis is required to get a hitting set generator whose seed length has optimal dependence on the error parameter [19]. Our lemma implicitly improves a result of Antunes and Fortnow [16], which requires a stronger derandomization hypothesis.

However, we are unable to implement this idea for an arbitrary language L that reduces to K^t. The idea *does* work for unary languages however, and we get that every unary language in NP is in $\mathsf{AM} \cap \mathsf{coAM}$, applying Theorem 3. The derandomization hypothesis can now be applied one more time to get a simulation in $\mathsf{NP} \cap \mathsf{coNP}$. By a standard upward translation argument, we get that $\mathsf{NE} = \mathsf{coNE}$. Thus, under the assumption on reducibility, either the derandomization hypothesis fails or $\mathsf{NE} = \mathsf{coNE}$, both of which are unlikely or, at the very least, surprising consequences.

2 Preliminaries

2.1 Basic Complexity Notions

We refer to the book by Arora and Barak [18] for definitions of standard complexity classes.

A time-constructible function $T : \mathbb{N} \to \mathbb{N}$ is a function such that there is a Turing machine transducer M, which for each n, on input 1^n halts with output $T(n)$ within $O(T(n))$ steps.

We say that L is a tally language if it is contained in Σ^* for some single-letter alphabet Σ.

2.2 Oracle Turing Machines

We consider randomized oracle Turing machines (OTMs). Our OTMs have a read-only input tape, a write-only output tape, one or more work tapes, a random tape, and an oracle tape. The alphabet for all tapes is $\{0, 1\}$. If the TM never makes oracle queries we say it is *oracle-free*.

We refer to an ordered pair (M, x) where M is a Turing machine and x is the input as a *context*.

We assume without loss of generality that a polynomial-time bounded randomized Turing Machine M comes equipped with a poly-time computable function $r^M : \{0,1\}^* \longrightarrow \mathbb{N}$ where r_n^M is a polynomial upper bound on the number of random bits generated by the machine on input x. On input x the TM starts by generating an auxiliary random string ρ of length r_x^M which is written on the random tape, and the program operates deterministicially on the pair x, ρ.

Normally, an oracle is a function O from a query set Q to an answer set A. We also need a more general notion of a *context-sensitive* oracle in which the response depends on the query q and the context (machine M and input x that is calling the oracle). We write $O_x^M(q)$ for the output of oracle O to query q.

Unless otherwise specified all oracles in this paper have query set $Q = \{0,1\}^*$ and answer set $A = \mathbb{N}$.

An OTM M *instantiated by oracle* O, denoted M^O is the program that uses oracle O to answer any queries. If O is computable, then M^O is *computably instantiated*, and can be identified with an oracle-free Turing machine, even in the case that O is a context-sensitive oracle.

The randomized OTMs we consider are *decision machines* which means that they always halt and output either 1 (identified with ACCEPT) or 0 (identified with REJECT). The language $L(M^O)$ recognized by an (possibly randomized) instantiated decision OTM M^O is the set of x for which the probability that $M(x)$ accepts is greater than $1/2$.

Let $\epsilon : \mathbb{N} \longrightarrow [0,1]$. We say that L *is accepted by* M^O *with advantage* ϵ if every accepted string x is accepted with probability at least $(1 + \epsilon(|x|))/2$ and every rejected string is accepted with probability at most $(1 - \epsilon(|x|))/2$. Note that computing with advantage $\epsilon = 1$ means that $M^O(x)$ always outputs $L(x)$.

An algorithm that accepts L with advantage ϵ can be converted to an algorithm with advantage $(1-\epsilon')$ by repeating the algorithm $O(\log(1/\epsilon') + (1/\epsilon))^2$ times and taking majority vote.

We restrict attention to OTMs that are *nonadaptive*. A nonadaptive OTM comes equipped with:

- A computable function $k^M : \{0,1\}^* \longrightarrow \mathbb{N}$. For $x \in \{0,1\}^*$, k_x^M is the number of oracle queries that M makes on input x. For convenience (and with no significant loss of generality) we assume that k_n^M is a power of 2.
- A computable function q^M that takes as input a string x and an integer $i \in \{1, \ldots, k_{|x|}^M\}$, and (if the OTM is randomized) a random string ρ of length $r_{|x|}^M$, and outputs an element of $\{0,1\}^*$. $q_x^M(i,\rho)$ is the ith query asked by M on input x when the random string is ρ.
- A computable function OUT^M that takes as input x and the sequence of query answers $(a_1, \ldots a_{k_x^M})$ and (if the OTM is randomized) the random string ρ and outputs either 0 or 1. $\text{OUT}_x^M(a_1, \ldots, a_{k_x^M}, \rho)$ is the output of the algorithm on input x, random string ρ when the query answers are $r_1, \ldots, r_{k_x^M}$.

In the above definition the nonadaptive OTM M is deterministic if r_x^M is identically 0. Thus the function q_x^M depends only on the query index i.

Q_x^M denotes the set of all $q \in \{0,1\}^*$ such that on input x, M asks query q with positive probability. Q_x^M is a finite set of size at most $2^{r_x^M} k_x^M$ since there are $2^{r_x^M}$ choices for ρ and each results in k_x^M queries.

We say that M is α-*honest*, for $\alpha : \mathbb{N} \longrightarrow \mathbb{N}$, provided that on any input x, all oracle queries have length at least $\alpha(|x|)$. If $\alpha(n) = n^\tau$ for some constant $\tau > 0$ we say that M is *polynomially honest*. Conventionally, a polynomially honest oracle algorithm is referred to simply as *h*onest, but it will be useful for us to distinguish different kinds of honesty.

If M is α-honest for some α that tends to ∞ we say that M is *weakly honest*. We say that M is *fixed query length* if the length of each query of M on input x depends only on $|x|$.

2.3 Approximate Oracles and Robust Oracle Algorithms

Recall that in this paper we consider oracles that take as input a string and output an integer. We now introduce two related concepts for an oracle (possibly context-sensitive) O:

- Approximation of O by another oracle O' that is "suitably close".
- An oracle algorithm M instantiated by O that is robust with respect to replacing O by any O' that is close enough to O.

The closeness of two oracles is measured by a nonnegative valued *closeness function* β with two arguments, a string q and natural number n. The value of β at these arguments is written $\beta_n(q)$. We say that O and P are *β-close with respect to M* provided that for all inputs x to M, and queries $q \in Q_x^M$, $|O_x^M(q) - P_x^M(q)| \leq \beta_{|x|}(q)$.

Let M be a randomized oracle algorithm, O be an oracle and L be the language accepted by M^O. Let β be a closeness function and let $\epsilon : \mathbb{N} \longrightarrow [0,1]$. We say that M is *(β, ϵ)-robust* with respect to oracle O provided that for every oracle O' that is β-close to O, $M^{O'}$ accepts L with advantage ϵ. We say that M is *β-robust* if it is $(\beta, 1)$-robust, which means that for every oracle O' that is β-close to O, $M^{O'}(x)$ always outputs the correct answer $L(x)$.

2.4 Descriptions and Kolmogorov complexity

Throughout we assume a fixed pairing function $< \cdot >$ and a standard encoding of Turing machines M by binary strings, with $\tilde{M}$ denoting the encoding of M.

We fix a universal Turing machine U that for any Turing machine M and input x takes as input $< \tilde{M}, x >$ and outputs $M(x)$. The Kolmogorov complexity $K(q)$ of q (with respect to U), denoted $K(q)$, is the minimum length of y such that $U(y) = q$. We define the Kolmogorov complexity of a machine M to be $K(\tilde{M})$. We have:

▶ **Proposition 7.** *For every computable function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ there is a constant C_f such that if $q = f(z)$ then $K(q) \leq K(z) + C_f$.*

In particular, for f being the identity function ID we have

▶ **Proposition 8.** *For all strings q, $K(q) \leq |q| + C_{ID}$.*

▶ **Proposition 9.** $\sum_{q \in \{0,1\}^*} \frac{2^{-K(q)}}{K(q)^2} \leq 2$.

Proof. Letting S_k be the set of strings of Kolmogorov complexity exactly k, we can rewrite the above sum as:

$$\sum_{k \geq 1} \sum_{q \in S_k} \frac{2^{-k}}{k^2}$$

Since $|S_k| \leq 2^k$, this is at most $\sum_{k \geq 1} 1/k^2 < 2$. ◀

The *canonical order* on $\{0,1\}^*$ is the order in which x precedes y if $|x| < |y|$ or $|x| = |y|$ and x is lexicographically less than y.

▶ **Proposition 10.** *Let L be a computable language that contains infinitely many strings.*

1. *Let $\Lambda : \mathbb{N} \longrightarrow \mathbb{N}$ be any computable unbounded function. Let x_j be the first string (according to the canonical order) satisfying $|x_j| \geq \Lambda(j)$ and $x \in L$. There is an integer C (depending on L and Λ) such that for all integers j, $K(x_j) \leq \log(j) + C$.*

2. *Let $\kappa : \mathbb{N} \longrightarrow \mathbb{Z}$ be any computable nondecreasing function that tends to ∞. Let L be a computable language that contains infinitely many strings. Then L contains a string x for which $K(x) < \kappa(|x|)$.*

Proof. Let M be a Turing Machine that computes L and N be a Turing machine that computes Λ. We can construct a machine M' that on input j outputs x_j. M first evaluates $\Lambda(j)$ and then enumerates strings x of length at least $\Lambda(j)$ in the canonical order and stops when it finds the first string in L. M' is a program whose length is a constant (depending on $|N|$ and $|M|$) and so the pair M', j has a description of length $\log(j) + C$.

For the second part, given κ, let Λ be the function where $\Lambda(j)$ is the least integer m such that $2\log(j) < \kappa(m)$. The function Λ is computable. Applying the first part, there is a constant C (depending on L and Λ) such that for all j, x_j is a string of length at least $\Lambda(j)$ such that $K(x_j) \leq \log(j) + C$. Choose $j = 2^C$. Then x_j has length at least $\Lambda(j)$, and $K(x_j) \leq \log(j) + C \leq 2\log(j) < \kappa(\Lambda(j)) \leq \kappa(|x_j|)$, as required. ◀

We also need the followng proposition giving symmetry of information for Kolmogorov complexity.

▶ **Proposition 11** (Symmetry of information). *Let x, y be strings. Then $K(x; y) = K(x) + K(y|x) + O(\log(K(x, y)))$.*

2.5 Resource-Bounded Kolmogorov Complexity

We study various notions of resource-bounded Kolmogorov complexity, and associated decision problems.

In the R_K problem, the instance is a string x, and the question is whether $K(x) \geq |x|/2$. Given a function $g : \mathbb{N} \to \mathbb{N}$, the R_K problem with gap g is the promise problem whose NO instances are strings x with $K(x) \geq |x|/2$ and whose YES instances are strings x with $K(x) < |x|/2 - g(|x|)$. Given a function $\alpha : \mathbb{N} \to \mathbb{R}$ such that $\alpha(n) \geq 1$ for all n, the R_K problem with multiplicative gap α is the promise problem whose NO instances are strings x with $K(x) \geq |x|/2$ and whose YES instances are strings x such that $K(x) < |x|/(2\alpha(|x|))$.

Given a time bound $t : \mathbb{N} \to \mathbb{N}$, the K^t complexity of a string is defined as follows: $K(x) = min\{|p| : U(p)$ halts and outputs x within t steps$\}$.

In the MCSP problem, the instance is a string x together with a parameter s, and the question is whether x is the truth table of a Boolean function with circuit complexity at most s.

MCSP is easily seen to be in NP, but it is unknown whether MCSP is NP-hard. A brute-force search strategy of trying all circuits C of size at most s and checking if any of them computes the function with truth table x can easily be implemented to run in time $\text{poly}(|x|)s^{O(s)}$.

3 Simulating M^K: the deterministic case

In this section we consider deterministic oracle machines M such that $M^{K'}$ computes L for any context-sensitive oracle K' that is suitably close to K and show (under suitably assumptions) that L is especially simple: either in P or P/poly. Allender, Buhrman, Friedman and Loff [4] use related ideas to show that any computable language that is non-adaptively reducible to K^t for some fixed time bound t is in P/poly. Their task is somewhat simpler because the reduction is to time-bounded Kolmogorov complexity for some fixed time bound; we, on the other hand, need to exploit the robustness of our presumed reduction with respect to approximation.

▶ **Theorem 12.** *Let $\beta = \beta_n(q)$ be a closeness function that is computable in time polynomial in n and $|q|$. Let L be a decidable language and M be a polynomial time deterministic oracle Turing Machine such that L is computed by $M^{K'}$ for any K' that is β-close to K. Let C and r be constants so that the running time of M on input x of length at most n is at most Cn^r.*

1. *Suppose that $\beta_n(q) = (2r+1)\log(n) + \omega_n(1)$. Then $L \in \mathsf{P}$.*
2. *If M is polynomially honest and $\beta_n(q) = \omega(\log(|q|))$ then $L \in \mathsf{P}$.*
3. *If $\beta_n(q) = \delta K(q)$ for some constant $\delta > 0$ then $L \in \mathsf{P/poly}$.*

Proof. Let L and M be as hypothesized.

For each string x, recall that k_x is the number of queries asked by M on input x. Let $Q_x = \{q_x(1), \ldots, q_x(k_x)\}$ be the set of queries asked on input x. Each of the queries $q_x(i)$ can be computed given i,x and M and so $K(q) \leq \log(k_x) + 2K(x) + O(1)$ for any $q \in Q_x$. Since $k_x \leq C|x|^r$ we have that for all x and all $q \in Q_x$:

$$K(q) \leq K(x) + 2r\log(|x|) + O(1). \tag{1}$$

(The factor 2 multiplying r is more than is needed, but is used to keep the expressions simple.)

The proofs of the first and third parts of the theorem have the following structure. In each part we make a specific definition for an oracle U, define B to be the set of strings x such that $M^U(x) \neq L(x)$. If B is finite, we can modify M to the machine $\hat{M}$ which on input x checks whether $x \in B$ and if so outputs $L(x)$ and otherwise runs M^U. The machine $\hat{M}^U$ also computes L. In the first part U is a just the oracle that always outputs 0, and so the resulting computation is in P. In the third part, U is a more complicated oracle, but the operation of U when M^U is run on input of length n can be implemented efficiently given an advice string A_n of polynomial length, and so the resulting computation is in $P/poly$.

So it will suffice to prove that B is finite. We will make use of the following:

▶ **Lemma 13.** *Let z be a string such that for all queries $q \in \{q_z(1), \ldots, q_z(k_z)\}$ we have $|U_z(q) - K(q)| \leq \beta_{|z|}(q)$. Then on input z, M^U outputs $L(z)$. In particular, $z \notin B$.*

Proof. Given such a z define the context-sensitive oracle U' as follows: $U'_y(q)$ is equal to $U_y(q)$ if $y = z$ and $q \in \{q_z(1), \ldots, q_z(k_z)\}$ and is equal to $K(y)$ otherwise. It follows immediately from the hypothesis on z that the oracle U' is β-close to K and therefore by the hypothesis of the theorem $M^{U'}$ computes L correctly on all inputs. Also $M^{U'}$ and M^U behave identically on input z, so $M^U(z)$ outputs $L(z)$. ◀

To prove that B is finite we assume for contradiction that B is infinite. Let F be the subset of $x \in B$ such that x is lexicographically minimum among all strings $y \in B$ of length x. Then F is also infinite. We then prove:

(*) For any sufficiently large $x \in F$, $|K(q) - U_x(q)| \leq \beta_{|x|}(q)$ for every query $q \in Q_x$.

We then select $z \in F$ so that $|z|$ is sufficiently large for this to happen. But then Lemma 13 contradicts that $z \in B$.

We now carry out this strategy for the first and third part of the Theorem.

For the first part of the Theorem, as mentioned, we define U so that $U_x(q) = 0$ for any input x and query q. Note that M^U is a polynomial time oracle-free computation.

Defining B as above, it suffices to show that B is finite. Suppose for contradiction that B is infinite, and define F as above. As noted above, it suffices to show (*) above. Note that for $x \in F$, $K(x) = \log|x| + O(1)$ since x can be described by the machine M, the oracle-free

machine N that computes L (since L is computable) and the number $|x|$. Using (1), for any $q \in Q_x$ we have $K(q) \leq (2r+1)\log(|x|)+O(1)$. By the hypothesis on β, and for x sufficiently large we get that for all $q \in Q_z$, $K(q) \leq \beta_{|x|}(q)$ and therefore $|U(q) - K(q)| \leq \beta_{|x|}(q)$.

The second part of the theorem follows from the first part. Assume that M is polynomially honest. Then there is a constant $\gamma > 0$ so that on input of length n, all queries have size at least n^γ. By hypothesis, $\beta_n(q) = \omega(\log(|q|)) = \omega(\log(n))$, and thus the hypothesis of the first part is satisfied.

For the third part, given δ and r as hypothesized let $D = (2r+2)/\delta$. Define the (context sensitive) oracle U as follows: on input x and oracle query q, $U_x(q)$ outputs $\min(K(q), D\log(n)($. Again we need to prove that the set B is finite. Assuming this is true, we can modify the machine M^V to the machine $\hat{M}^V$ which on input x first checks whether $x \in B$, and if so outputs $L(x)$ and otherwise runs $M^W[x]$. Now $\hat{M}^V$ can be implemented in P/poly since for any input x of size n, the action of V on queries asked during the computation $M^V(x)$ can be specified by an advice string A_n of polynomial size. Indeed, let S_n be the set of strings of length at most cn^r that satisfy $K(q) < D\log(n)$ and let A_n be the set of ordered pairs $(q, K(q))$ for $q \in S_n$. (Note that during the computation of $M^V(x)$ given a query q, $V_x(q) = D\log(n)$ unless $q \in S_n$, in which case the second part of the ordered pair specifies the query answer.)

For use below, we observe that $K(A_x) \leq D\log(|x|)+O(1)$ since $A_{|x|}$ can be reconstructed by knowing the number $T \leq 2n^D$ of programs of length less than $D\log(|x|)$ that output a string in $S_{|x|}$. (Given T, $S_{|x|}$ can be determined by interleaving the execution of all programs of length less than $D\log(n)$ until T of them halt with an output of length at most cn^r. S_n is then the set of such output strings, and for each string $q \in S_n$, $K(q)$ is the length of the shortest program that output q.)

It remains to prove that B is finite, and again we suppose for contradiction that B is infinite, and define the infinite set F as above. Again we will show that (*) is satisfied. For $x \in F$, $K(x) = \log|x| + K(A_{|x|}) + O(1)$ since since x can be described by the machine M, the oracle-free machine N that computes L (since L is computable) the advice string $A_{|x|}$ (which allows simulation of the oracle V during the computation of $M(x)$) and the number $|x|$. From (1), for any $q \in Q_x$ we have $K(q) \leq (2r+1)\log(|x|) + K(A_x) + O(1) \leq (2r+1+D)\log(|x|) + O(1)$ which is at most $D(1+\delta)\log(|x|)$ provided that $|x|$ is sufficiently large. For any such x, if $K(q) < D\log(|x|)$, then $U_x(q) = K(q)$ so that $|U_x(q) - K(q)| = 0$, or $K(q) \geq D\log(|x|)$, in which case $U_x(q) = D\log(|x|)$ and $K(q) \in [D\log(|x|), D(1+\delta)\log(|x|)]$ and so $|U_x(q) - K(q)| \leq \delta D\log(|x|) \leq \delta K(q)$, establishing (*) as required. ◀

4 Simulating robust K-oracle computation in $AM \cap coAM$

Throughout this section we make the following assumptions.

H1 M is a randomized nonadaptive decision OTM

H2 M^K recognizes the computable language L.

H3 $\beta = \beta_n(q)$ is a computable closeness function.

H4 $\epsilon : \mathbb{N} \longrightarrow [0,1]$.

H5 M is (β, ϵ)-robust with respect to oracle K.

H6 $\alpha : \mathbb{N} \longrightarrow \mathbb{N}$ is an unbounded non-decreasing function.

Also, let k_n^M denote the maximum of k_x^M over x of length n.

Our overall goal is to prove the following theorem:

▶ **Theorem 14.** *Suppose $M, \alpha, \beta, \epsilon$ satisfy hypotheses [H1-H6]. Suppose also that $\beta_n(q) = \log(k_n^M/\epsilon(n)) + 2\log K(q) + \alpha(n) + 5$. Then both L and $\bar{L}$ have constant round Arthur-Merlin protocols for membership whose running time is polynomial in the running time of M and $1/\epsilon$. In particular if M runs in polynomial time then $L \in AM \cap coAM$.*

The rather cumbersome condition on β arises from the proof. It is more natural if the closeness function depends only on the query q and not on the length of the input x to the calling algorithm.

We prove Theorem 14 in two parts. First, in this section, we construct a parameterized family $\{J[\gamma] : \gamma \in [0,1)\}$ of computable context-sensitive oracles J based on simple combinatorial properties of the machine M and show that there exists an integer ℓ such that under [H1]-[H6], $M[\ell]^{J[\gamma]}$ computes L with advantage $\epsilon/2$ for every $\gamma \in [0,1)$.

Then, in the next section, we show that if L satisfies the conclusion of the first part, then it has an efficient AM protocol and coAM protocol.

4.1 Some additional preliminaries

Let (M, x) be a context. As usual, for a set or quantity Z that depends on the context, we denote this dependence by Z_x^M, but we often suppress this dependence and write simply Z. For example we write r for r_x^M and k for k_x^M.

Let $\Gamma = \Gamma_x^M = \{(\rho, i) : (\rho, i) \in \{0,1\}^r \times \{1, \ldots, k\}\}$. Under the assumption stated earlier that k is a power of 2 we have that $\log|\Gamma| = r + \log k$ is an integer, and we can identify Γ with the set $\{0,1\}^{r+\log k}$.

It will be convenient to introduce notation for some elementary functions of positive real numbers.

For a nonnegative integer s:

- ϕ is the function given by $\phi(s) = \lceil \log(\frac{\Gamma}{s}) \rceil = \log(|\Gamma|) - \lfloor \log s \rfloor$.
- μ is the function given by: $\mu(s)$ is the least nonnegative number γ such that $s(1+\gamma)$ is a power of 2. Note that $\mu(s) \in [0,1)$.

It is easy to check:

▶ **Proposition 15.** *For $\gamma \in [0,1)$ and $s > 0$, $\phi((1+\gamma)s) = \phi(s)$ if $\gamma < \mu(s)$ and $\phi((1+\gamma)s) = \phi(s) - 1$ if $\gamma \geq \mu(s)$.*

The following technical fact will be needed later.

▶ **Proposition 16.** *Let $s, t \in \mathbb{N}$ and $\tau \in [0,1)$ with $t \in [s/(1+\tau), s(1+\tau)]$. Let $\gamma \in [0,1)$ be such that $\gamma \notin [\mu(s) - 2\tau, \mu(s) + 2\tau] \cup [1 - 2\tau, 1) \cup [0, 2\tau)$. Then $\phi((1+\gamma)s)) = \phi((1+\gamma)t)$.*

Proof. We prove the contrapositive. Suppose $\phi((1+\gamma)s) \neq \phi((1+\gamma)t)$. Then $\lfloor \log((1+\gamma)s) \rfloor \neq \lfloor \log((1+\gamma)t) \rfloor$. This means that there is an integral power of 2 between them, which means that the interval $[(1+\gamma)s/(1+\tau), (1+\gamma)(1+\tau)s]$ contains an integral power of 2, say 2^w. Note that $s \leq 2^{w+1} < 8s$. This implies that $(1+\mu(s))s = 2^{w+1}$, or $(1+\mu(s))s = 2^w$, or $(1+\mu(s))s = 2^{w-1}$.

Case 1. Suppose $(1+\mu(s))s = 2^{w+1}$. This means $s > 2^w$, and since $[(1+\gamma)s/(1+\tau), (1+\gamma)(1+\tau)s]$ contains 2^w, we have that $\tau > \gamma$. Hence $\gamma \in [0, 2\tau)$.

Case 2.Suppose $(1+\mu(s))s = 2^w$. Then $(1+\mu(s))s$ belongs to the interval $[(1+\gamma)s/(1+\tau), (1+\gamma)(1+\tau)s]$ so $1+\mu(s)$ belongs to the interval $[(1+\gamma)/(1+\tau), (1+\gamma)(1+\tau)] \subseteq [1+\gamma-2\tau, 1+\gamma+2\tau]$ which implies $\gamma \in [\mu(s) - 2\tau, \mu(s) + 2\tau]$.

Case 3. Suppose $(1+\mu(s))s = 2^{w-1}$. Then $2 \leq 2(1+\mu(s)) = 2^w/s \leq (1+\gamma)(1+\tau) \leq 1+\gamma+2\tau$ which implies $\gamma \geq 1 - 2\tau$. ◀

4.2 Replacing K by the oracle $J[\gamma]$

As described earlier the context (M, x) determines a function $q_x^M : \Gamma \longrightarrow \{0,1\}^*$ where $q_x^M(\rho, i)$ is the ith query asked by M on input x when the random string is ρ.

For a query q, we define $S(q) = S_x^M(q)$ to be the set $\{(\rho, i) \in \Gamma : q_x^M(\rho, i) = q\}$.

Let $s(q) = |S(q)|$ and let $\pi(q) = \pi_x^M(q) = \frac{s_x^M(q)}{|\Gamma|}$. This is the probability, with respect to (ρ, i) chosen uniformly from Γ that $q(\rho, i) = q$. Note that for the function ϕ defined earlier we have $\phi(s(q)) = \lceil \log \frac{1}{\pi(q)} \rceil$.

For each $\gamma \in [0, 1)$, define the function $J[\gamma]$ on queries by:

$$J[\gamma](q) = J[\gamma]_x^M(q) = \phi((1+\gamma)s(q)).$$

If $\gamma = 0$ we often write $J(q)$ for $J[\gamma](q)$.

From the definition we observe that for all $\gamma \in [0, 1)$:

$$J[\gamma](q) \in [\log(\frac{1}{\pi(q)}) - 1, \log(\frac{1}{\pi(q)}) + 1]. \tag{2}$$

We view $J[\gamma]$ as a context-sensitive oracle and we will see below that $J[\gamma]_x^M$ can be used in place of K in a suitably robust oracle algorithm. The following Lemma relates $J[\gamma]$ to K:

▶ **Lemma 17.** *Let M be a randomized oracle algorithm and x a string. Recall that Q_x^M is the set of queries q such that M on input x asks the query q with positive probability.*

1. *There are constants C_0, C_1 such that for any string $q \in Q_x^M$, $K(q) - J[\gamma]_x^M(q) \leq C_0 K(x) + C_1$.*
2. *For $u > 0$ let $P_x^M(u)$ be the set of strings in Q_x^M such that $J[\gamma]_x^M(q) - K(q) \geq u + 2\log K(q)$. Then $\pi_x^M(P_x^M(u)) \leq 2^{3-u}$.*

Proof. Fix M and x. We omit the superscript M and subscript x from $J[\gamma]$, π, P and Q.

For the first part, order the elements of Q lexicographically by the triple $(1/\pi(q), |q|, q)$. For a query q, let $p(q)$ be the number of $q' \in Q$ that precede q in this order. Given the Turing machine M and input x we can construct a Turing machine of size $O(K(x)) + O(1)$ that takes as input an integer $h \in \{0, \ldots, |Q| - 1\}$ and outputs q such that $p(q) = h$.

Therefore $K(q) \leq O(K(x)) + \log p(q) + O(1)$. Since there are at most $1/\pi(q)$ members of Q that precede q in the order, we have $p(q) \leq (1/\pi(q))$ and therefore $\log p(q) \leq \log(1/\pi(q)) \leq J[\gamma](q) + 1$. Therefore $K(q) - J[\gamma](q) \leq O(K(x)) + O(1)$, as required.

For the second part, (2) implies $\pi(q) \leq 2^{1-J[\gamma](q)}$. For $q \in P(u)$ we therefore have $\pi(q) \leq 2^{1-K(q)-u}/K(q)^2$. Summing over all $q \in P(u)$ this is at most $2^{1-u} \sum_{q \in P(u)} 2^{-K(q)}/K(q)^2$ which is at most 2^{2-u} by Proposition 9. ◀

4.3 Replacing a K oracle by a $J[\gamma]$ oracle

In this section we prove:

▶ **Theorem 18.** *Assume hypotheses [H1]-[H6] and that $\beta_{|x|}(q) = \log(k_x^M/\epsilon(|x|)) + 2\log K(q) + \alpha(|x|) + 5$. Then there is an integer ℓ such that for all $\gamma \in [0, 1]$, $M[\ell]^{J[\gamma]}(x)$ computes L with advantage $\epsilon/2$, where $M[\ell]$ is the OTM that outputs $L(x)$ on any input x of length less than ℓ and otherwise executes M on x. Letting $N = M[\ell]$, there is an OTM N such that $N^{J[\gamma]}(x)$ computes L with advantage $\epsilon/2$.*

Proof. Fix M. Say that x is bad for γ if $M^{J[\gamma]}(x)$ outputs $L(x)$ with probability less than $\frac{1}{2}(1+\epsilon(|x|)/2)$ and let $B[\gamma]$ be the set of strings that are bad for γ. Let $B = \cup_{\gamma\in[0,1)}B[\gamma]$.

If B is finite, let ℓ be the length of the largest string in B. Then $M[\ell]$ satisfies the desired conclusion. So we assume that B is infinite and derive a contradiction.

Note that $x \in B$ holds if and only if one of the infinitely many conditions $x \in B[\gamma]$ holds. The following Proposition reduces this to a finite set of conditions.

▶ **Proposition 19.** *Let $x \in \{0,1\}^*$ and let $\Gamma_x = \{\mu(s(q)) : q \in Q_x^M\}$ (where Q_x^M is the (finite) set of queries that are asked with nonzero probability by M on input x and μ is the function defined in the definition of $J[\gamma]$). Then $x \in B$ if and only if $x \in B[\gamma]$ for some $\gamma \in \Gamma_x$.*

Proof. For each $\gamma \in [0,1)$ let $\gamma^- = \max\{\mu(s(q)) : q \in Q_x^M, \mu(s(q)) \le \gamma\}$. By the definition of $J[\gamma]$, we have $J[\gamma](q) = J[\gamma^-](q)$ for all $q \in Q$. Therefore $M^{J[\gamma]}(x)$ behaves identically to $M^{J[\gamma-](x)}$ and so $x \in B$ if and only if there exists $\gamma \in \Gamma_x$ such that $x \in B[\gamma]$. ◀

Let us define γ_x to be the least $\gamma \in \Gamma_x$ for which $x \in B[\gamma_x]$.

The set Γ_x is computable from x, and for each $\gamma \in \Gamma_x$ we can computably determine the probability that $M^{J[\gamma]}(x)$ agrees with $L(x)$. Therefore there is an (instantiated) program of size $|M| + O(1)$ that on input x, tests whether $x \in B$.

Let $\beta^*(n)$ be the minimum over all q and all $n' \ge n$ of $\beta_{n'}(q)$. To obtain the desired contradiction we will prove:

▶ **Lemma 20.** *For any string x there is an oracle $K[x]$ satisfying:*

- *$K[x]$ is β-close to K and therefore the probability that $M^{K[x]}$ on input x differs from M^K on input x is at most $(1-\epsilon(|x|))/2$.*
- *If $K(x) < (\beta^*(|x|) - C_1)/C_0$ (where C_0, C_1 are given in Lemma 17) then the probability that $M^{K[x]}$ on input x differs from M^J on input x is at most $\epsilon(|x|)/4$.*

The lemma implies any string x satisfying $K(x) < (\beta^*(|x|) - C_1)/C_0$ is not in B, since the probability that M^J on input x disagrees with M^K on input x can be upper bounded by $(1-\epsilon(|x|)/2 + \epsilon(|x|)/4 \le (1-\epsilon(|x|)/2)/2$. But by applying the second part of Proposition 10 to the language B and $\kappa(n) = (\beta^*(n) - C_1)/C_0$ there is a string $x \in B$ such that $K(x) < (\beta(|x|) - C_1)/C_0$, which yields the desired contradiction. Note that κ is an unbounded non-decreasing function as required for the application of Proposition 10 by the definition of β^* and the fact that α is an unbounded non-decreasing function.

So it remains to prove Lemma 20. Given x, we define the oracle $K[x]$ as follows. Recall that Q_x^M is the set of queries q that are made by M with non-zero probability when the input is $x(w)$. Let Q_{CLOSE} be the set of $q \in Q_x^M$ for which $|J[\gamma]_x^M(q) - K(q)| \le \beta_{|x|}(q)$, let $Q_>$ be the set of $q \in Q_x^M$ for which $J[\gamma_x]_x^M(q) > K(q) + \beta_{|x|}(q)$ and let $Q_<$ be the set of $q \in Q_x^M$ for which $J[\gamma_x](q) < K(q) - \beta_{|x|}(q)$ Define the oracle $K[x]$ as follows: $K[x](q) = J[\gamma_x](q)$ if $q \in Q_{\text{CLOSE}}$ and $K[x](q) = K(q)$ otherwise.

By definition $K[x]$ is β-close to K and therefore by hypothesis $M^{K[x]}$ computes L with advantage ϵ. In particular $M^{K[x]}$ on input x differs from $L(x)$ with probability at most $(1-\epsilon(|x|))/2$. This completes the first part of the lemma.

For the second part of the lemma, assume that x is such that $K(x) < (\beta(|x|) - C_1)/C_0$ (where C_0, C_1 are given in Lemma 17).

If all of the queries asked by M on input x are in Q_{CLOSE} then all query answers during the computation $M^{J[\gamma]}(x)$ are identical to those during the computation $M^{K[x]}(x)$, and therefore the output is the same. .

So it now suffices to prove an upper bound of $\epsilon(x)/4$ on the probability that at least one of the selected queries on input x belongs to $Q_< \cup Q_>$.

From the hypothesis, we have $K(x) \leq (\beta(|x|) - C_1)/C_0$. We now show that this implies that $Q_< = \emptyset$. By the first part of Lemma 17, for all queries $q \in Q_x^M$ we have $K(q) - J_x^M(q) \leq C_0 K(x) + C_1$ which is at most $\beta(|x|)$ and so $q \notin Q_<$.

Hence we only need to upper bound the probability that, on input x, at least one query belongs to $Q_>$. For brevity let $k = k_x^M$. For $i \in \{1, \ldots, k\}$ let p_i be the probability that the ith query is in $Q_>$. Let $\sigma = \sum_i p_i$, it suffices to show that $\sigma \leq \epsilon(|x|)/4$. Note that σ/k is the chance that a random query i (with respect to the random bits and the choice of $i \in \{1, \ldots, k\}$) is in $Q_>$. By the second part of Lemma 17, with $u = \beta(|x|) - 2\log(K(q))$, $p \leq 2^{2+2\log(K(q))-\beta(|x|)}$. Using the hypothesis on $\beta(|x|)$ this is at most $\epsilon(|x|)/4k$ and therefore $\sigma \leq \epsilon(|x|)/4$, as required. ◀

5 The Arthur-Merlin protocol

The starting point for this section is the conclusion of Theorem 18:

> **Hypothesis AM(N).** N is an oracle machine such such that for all $\gamma \in [0, 1)$, $N^{J[\gamma]}(x)$ computes L with advantage $\epsilon/2$.

We will show that this implies that there is an Arthur-Merlin protocol for L whose running time is in the running time of N.

We make use of two known AM protocols. The input to both protocols includes:

- A function $f : \{0,1\}^w \longrightarrow \{0,1\}^*$ given by a circuit C_f. The function f induces a probability distribution π_f on $\{0,1\}^*$, with $\pi_f(q)$ equal to $|f^{-1}(q)|/2^2$, which is the probability that a uniformly chosen element of $\{0,1\}^w$ maps to q.
- $\zeta, \delta > 0$ are input parameters for the protocol.

▶ **Lemma 21** (Entropy Estimation Protocol, [24]). *There is a constant round AM-protocol that takes as input a circuit C_f as above, an integer h and an error parameter δ, runs in time $|C_f| poly(\log(1/\delta))$, and outputs* ACCEPT *or* REJECT *with the following conditions*

1. *If $|h - H(\pi_f)| \leq 1$ then Merlin has a strategy that causes Arthur to reject with probability at most δ.*
2. *If $|h - H(\pi_f)| \geq 2$ then the probabilty that the verification is accepted is at most δ.*

▶ **Lemma 22** (Lower Bound protocol, [25]). *There is a constant round protocol with input C_f, input sequences $q_1, \ldots, q_m \in (\{0,1\}^*)^m$ and $(s_1, \ldots, s_m) \in \mathbb{N}^m$, and error parameters (ζ, δ) both in $(0, 1)$ that runs in time $C_f poly(m, 1/\zeta, \log(1/\delta))$, and outputs either* ACCEPT *or* REJECT *according to the following:*

1. *If $s_i \geq |f^{-1}(q_i)|$ for every i, then Merlin has a strategy such that the protocol rejects with probability at most δ.*
2. *If $s_i \leq |f^{-1}(q_i)|/(1 + \zeta)$ for some i then for any strategy of Merlin the probabilty that the verification is accepted is at most δ.*

We fix some notation for this section:

- x denotes the string whose membership in L is to be determined.
- $r = r_x^N$ is the length of the random string generated by N on input x.
- $k = k_x^N$ is the number of queries asked by N on input x. We assume (with no significant loss of generality) that $\log k$ is an integer.
- For $\rho \in \{0,1\}^r$, $q_1(\rho), \ldots, q_k(\rho)$ denotes the queries asked by N on input x.
- f denotes the function on $\{0,1\}^{r+\log k}$ that maps (ρ, i) to $q_i(\rho)$.
- Let $\pi = \pi_f$ be the probability distribution on $\{0,1\}^*$ induced by f, which is given by $\pi_f = |f^{-1}(q)|/2^{r+\log k}$.

- $Q = Q_x^N$ is the image of the function f, which is the set of queries that are asked with positive probability.
- For $\rho \in \{0,1\}^r$, $a_1[\gamma](\rho), \ldots, a_k[\gamma](\rho)$ is the sequence of answers from the oracle $J[\gamma]$ to the queries $q_1(\rho), \ldots, q_k(\rho)$.
- When the random string is ρ, the output of the algorithm is $\text{OUT}(a_1[\gamma](\rho), \ldots, a_k[\gamma](\rho); \rho)$. We denote this by $z[\gamma](\rho)$.
- $\epsilon = \epsilon_{|x|}^N$.

We note the following technical fact:

▶ **Proposition 23.** *Let π be a probability distribution over $\{0,1\}^w$.*

1. *If q is selected according to π, $\mathrm{E}[\log(1/\pi(q))] = H(\pi)$ where $H(\pi)$ is the binary entropy of π.*
2. *Suppose $q_1, \ldots, q_t$ are selected independently according to π. Then*

$$\mathrm{Prob}[\frac{1}{t}\sum_{i=1}^{t} \log(1/\pi(q_i)) - H(\pi)| > \zeta] \leq \frac{w^2}{t\zeta^2}.$$

Proof. The first part follows directly from the definition of $H(\pi)$. For the second part, let $Z_i = \log(1/\pi(q_i))$ and let $Z = \frac{1}{t}\sum Z_i$. Note that $\mathrm{E}[Z] = \frac{1}{t}\sum \mathrm{E}[Z_i] = H(\pi)$ by the first part. Also, note that since Z is an average of t i.i.d. random variables, it's variance is $\frac{1}{t}\mathrm{Var}(Z_i) \leq \frac{w^2}{t}$ since a random variable taking values in $[0,w]$ has variance at most w^2. By Chebyshev's inequality, for any random variable X, the probability that $|Z - \mathbb{E}[Z]| > \zeta$ is at most $Var(Z)/\zeta^2 \leq \frac{w^2}{t\zeta^2}$. ◀

In describing the Arthur-Merlin protocol we refer to the actions of "Honest Merlin" which is the ideal behavior for Merlin (which Merlin may not follow).

The protocol makes use of several parameters, whose values are determined by the analysis. For reference we list these parameters here with their values:

- t is the *sample complexity*, and it set to $t = 32000\frac{\max(r,1)^2k^2}{\epsilon(|x|)^2}$.
- τ is the *safety parameter*, which is set to $\tau = \frac{\epsilon(|x|)}{16(4k+2)}$
- ν is the *lower bound parameter*, which is set to $\nu = \frac{\epsilon(|x|)^2}{20000k^2}$.
- D is the *precision parameter* and is a positive integer. We choose $D = \lceil \frac{16(4k+2)}{\epsilon(|x|)} \rceil$.

It is straightforward to verify that these parameter values satisfy the following conditions:

▶ **Proposition 24.**

$$\begin{aligned}
\tau &\leq \frac{\epsilon(|x|)}{16(4k+2)} \\
\nu &\leq \tau\epsilon(|x|)/128k \\
t &\geq \frac{256k}{\epsilon(|x|)}\log\frac{16}{\epsilon(|x|)} \\
t &\geq 16\frac{(r+\log k)^2}{\epsilon(|x|)}. \\
t &\geq \frac{320k}{\epsilon(|x|)\tau}. \\
D &\geq \frac{1}{\tau}. \\
D &\geq \frac{32(k+1)}{\epsilon(|x|)}
\end{aligned}$$

The inequalities stated in this Proposition are needed in the analysis of the protocol. The values of the parameters were chosen to ensure these conditions.

We are now ready to state the protocol:

Step 1. Merlin provides an integer h. Arthur and Merlin perform the Entropy estimation protocol of Lemma 21 for the probability distribution π and input h with error parameter $\delta = \epsilon(|x|)/16$. (If Merlin is honest, then Merlin sets h so that $|h - H(\pi)| \leq 1$, and adopts the strategy from the first part of Lemma 21 that makes the rejection probability at most $\epsilon(|x|)/16$.)

Step 2. Random strings $\rho_1, \ldots, \rho_t$, each of length r, are generated using public coins. Let χ be the $t \times k$ matrix with $\chi_{i,j} = f(\rho_i, j)$, the jth query asked when the random string is ρ_i.

Step 3. Merlin provides a t by k positive integer matrix B (Honest Merlin sets B to be equal to the matrix B^* whose i, j entry is $|f^{-1}(\chi_{i,j})|$.)

Notational Remark. It is convenient here to define some matrices and vectors that are determined from B. For $\gamma \in [0, 1)$ the $t \times k$ matrix $A[\gamma]$ is given $A[\gamma]_{i,j} = \phi((1+\gamma)B_{i,j})$, and the vector $z[\gamma] \in \{0,1\}^t$ is given by $z[\gamma]_i = \textsc{out}(A[\gamma]_{i,1}, \ldots, A[\gamma]_{i,k})$. The matrix $A^*[\gamma]$ and vector $z^*[\gamma]$ are obtained analogously with B replaced by B^*. Note that row i of $A^*[\gamma]$ is exactly the sequence of query answers provided by oracle $J[\gamma]$ when the random string is ρ_i and $z^*[\gamma]_i$ is the output of $N^{J[\gamma]}(x)$ for random string is ρ_i.

Step 4. For each row index i, Arthur randomly chooses $\hat{j}(i) \in \{1, \ldots, k\}$. Define

$$\sigma(B) = \frac{1}{t} \sum_i (r - \log(B_{i,\hat{j}(i)})).$$

Arthur computes $\sigma(B)$ approximately to determine an integer H' that belongs to the interval $[\sigma(B) - 1, \sigma(B) + 1]$. Arthur rejects this step if $|H' - h| > 3$ and accepts this step otherwise.

Step 5. Arthur-Merlin perform the lower bound protocol of Lemma 22 for the above-mentioned function f that maps (ρ, i) to $q_i(\rho)$, with error parameters $(\zeta, \delta) = (\nu, \epsilon(|x|)/16)$. The input $(q_1, \ldots, q_m), (s_1, \ldots, s_m)$ to the protocol is the sequence of tk entries of the matrix χ and the corresponding sequence of entries of the matrix B. (If Merlin is Honest then since $B = B^*$, $s_i = |f^{-1}(q_i)|$ for every i and Merlin follows the strategy from the first part of Lemma 22 so that the probability of rejection is at most $\epsilon(|x|)/16$.)

Step 6. Arthur selects a uniformly random $\hat{\lambda} \in \{1, \ldots, D\}$ and a uniformly random $\hat{i} \in \{1, \ldots, t\}$. $\hat{z}$ is defined to be $z[\lambda/D]_{\hat{i}}$. For future reference we also define $\hat{z}^*$ to be $\hat{z}$ if B is replaced by B^*.

Step 7. If any of the steps 1,4 or 5 result in REJECT, then the protocol outputs FAILURE. Otherwise the output is $\hat{z}$.

This protocol outputs either FAILURE or 0 or 1. The main theorem of this subsection is:

▶ **Theorem 25.** *Under the hypotheses AM(N) given earlier, the above protocol satisfies:*

1. *If Merlin behaves honestly, then the protocol outputs $L(x)$ with probability at least $\frac{1}{2} + \epsilon(|x|)/16$.*
2. *For any behavior of Merlin, the probability that the protocol outputs FAILURE or outputs $L(x)$ is at least $\frac{1}{2} + \epsilon(|x|)/16$.*

▶ **Corollary 26.** *Both L and $\bar{L}$ can be recognized by AM protocols that run in time polynomial in the $1/\epsilon$ and the running time of N.*

Proof. The protocol obtained by changing a FAILURE output to 0 (resp. 1) is an AM protocol (resp. coAM protocol) for L. ◀

Proof of Theorem 25. We first note the following:

▶ **Proposition 27.** *The value $\hat{z}^*$ determined in Step 6 in the case that $B = B^*$ agrees with $L(x)$ with probability at least $\frac{1}{2} + \frac{\epsilon(|x|)}{4}$.*

Proof. The value $\hat{z}^*$ determined in Step 6 is equal to the output of $N^{J[\lambda/D]}$ on input x when the random string is $\rho_{\hat{i}}$. Since $\rho_{\hat{i}}$ is a uniformly random string, the hypothesis on N implies that this is equal to $L(x)$ with probability at least $\frac{1}{2} + \frac{\epsilon(|x|)}{4}$. ◀

We now prove the first part of the Theorem. For $s \in \{1, 4, 5\}$ define the event R_s to be the event that the test in Step s outputs REJECT.

Assume Merlin behaves honestly, and therefore $B = B^*$. By the above proposition, the probability that the output of the protocol is $L(x)$ is at least:

$$\frac{1}{2} + \frac{\epsilon(|x|)}{4} - \text{Prob}[R_1 \vee R_4 \vee R_5] \geq \frac{1}{2} + \frac{\epsilon(|x|)}{4} - \text{Prob}[R_1] - \text{Prob}[R_4] - \text{Prob}[R_5].$$

As noted in the description of Steps 1 and 5, $\text{Prob}[R_1]$ and $\text{Prob}[R_5]$ are each at most $\epsilon(|x|)/16$. So it suffices to show that $\text{Prob}[R_4] \leq \epsilon(|x|)/16$, which we now do.

Step 4 is rejected only if $|H' - h| > 3$. We have:

$$|H' - h| \leq |H' - \sigma(B^*)| + |\sigma(B^*) - H(\pi)| + |H(\pi) - h|.$$

$|H' - \sigma(B^*)| \leq 1$ by the choice of H' and the fact that $B = B^*$, and $|H(\pi) - h| \leq 1$ since Merlin is assumed to be honest. Therefore $\text{Prob}[R_4] \leq \text{Prob}[|\sigma(B^*) - H(\pi)| > 1$. So it suffices to prove:

▶ **Proposition 28.** $\text{Prob}[|\sigma(B^*) - H(\pi)| > 1] \leq \epsilon(|x|)/16$.

Proof. By the definition of B^*, $B^*_{i,\hat{j}} = 2^{r+\log k}\pi(\chi_{i,\hat{j}(i)})$ and so $r + \log k - \log(B^*_{i,\hat{j}(i)}) = \log(1/\pi(\chi_{i,\hat{j}(i)}))$ and so $\sigma(B^*) = \frac{1}{t}\sum_{i=1}^{t} \log(1/\pi(\chi_{i,\hat{j}(i)}))$. Since $\chi_{1,\hat{j}(1)}, \ldots, \chi_{t,\hat{j}(t)}$ is a sequence of independent samples from the distribution π, Proposition 23 implies

$$\text{Prob}[\frac{1}{t}\sum_i \log(1/\pi(\chi_{i,\hat{j}(i)})) - H(\pi)| > 1] < \frac{(r + \log k)^2}{t} \leq \epsilon(|x|)/16,$$

where the last inequality used Proposition 24. ◀

This completes the proof of the first part of the theorem.

We now turn to the proof of the second part. We consider an arbitrary strategy for Merlin. We break up into cases according to properties of the matrix B.

We divide into three cases.

Case 1. There is at at least one entry (i, j) with $B_{i,j} < B^*_{i,j}/(1 + \nu)$. By the property of the Lower bound protocol performed in Step 5, this will reject with probability at least $1 - \epsilon(|x|)/16 \geq \frac{1}{2}) + \epsilon(|x|)/4$.

Case 2. $B_{i,j} \geq B^*_{i,j}/(1+\nu)$ for all entries (i,j) and there are more than $\epsilon(|x|)t/16$ entries with $B_{i,j} > (1+\tau)B^*_{i,j}$. We claim that Step 4 will reject with probability at least $1-\epsilon(|x|)/8 \geq 1/2+\epsilon/4$. For this we give a lower bound on the probability $H' - H(\pi)$ exceeds 3. We have

$$\begin{aligned} H' - H(\pi) &= \sigma(B) - \sigma(B^*) - (\Sigma(B) - H') - (H(\pi) - \sigma(B^*)) \\ &\geq \sigma(B) - \sigma(B^*) - |\sigma(B) - H'| - |H(\pi) - \sigma(B^*)|. \end{aligned}$$

By the choice of H', $|\sigma(B) - H'| \leq 1$. Therefore:

$$\begin{aligned} \text{Prob}[H' - H(\pi) > 3] &\geq \text{Prob}[(\sigma(B) - \sigma(B^*) \geq 5) \wedge (|\sigma(B^*) - H(\pi)| < 1)] \\ &\geq 1 - \text{Prob}[\sigma(B) < \sigma(B^*) + 5] - \text{Prob}[|\sigma(B^*) - H(\pi)| \geq 1]. \end{aligned}$$

The desired lower bound will follow by showing that each of the two probabilities on the right is at most $\epsilon(|x|)/16$. Proposition 28 shows this for the second probability. So we now bound the first probability.

For $i \in \{1,\ldots,t\}$ define:

- $V_i = \{j \in \{1,\ldots,k\} : \log B_{i,j} \geq (1+\tau)B^*_{i,j}$. (Note that $\sum_i V_i \geq \epsilon(|x|)t/16$ by the case assumption.)
- Y_i is the 0-1 indicator of the event that $\hat{j}(i) \in V_i$, and $Y = \sum_i Y_i$.

We now make two claims.

Claim 1. $\sigma(B) - \sigma(B^*) \geq \tau Y - \nu t$.

Claim 2. $\text{Prob}[Y < \frac{1}{2}\frac{\epsilon(|x|)t}{16k}] \leq \epsilon(|x|)/16$.

If $Y \geq \frac{1}{2}\frac{\epsilon(|x|)t}{16k}$, then by the conditions on τ and ν given by Proposition 24 we have that $\tau Y - \nu t \geq 5$ and so

$$\text{Prob}[\sigma(B) - \sigma(B^*) < 5] \leq \text{Prob}[Y < \frac{1}{2}\frac{\epsilon(|x|)t}{16k}] \leq \epsilon(|x|)/16.$$

To prove the first claim, note that for i such that $Y_i = 1$, we have

$$\log(B_{i,\hat{j}(i)}) \geq \log((1+\tau)B^*_{i,\hat{j}(i)} \geq \log(B^*_{i,\hat{j}(i)}) + \tau$$

(since $\log(1+\tau) \geq \tau$ for $\tau \in [0,1]$). For i such that $Y_i = 0$, we have $B_{i,\hat{j}(i)} \geq B_{i,\hat{j}(i)}/(1+\nu)$ which implies $\log B_{i,\hat{j}(i)} \geq \log B_{i,\hat{j}(i)} - 2\nu$ since $\log(1+\nu) \leq 2\nu$). Therefore $\sigma(B) - \sigma(B^*) \geq \tau Y - \nu t$.

For the second claim, note that that $\text{E}[Y] = \sum_i \text{E}[Y_i] = \sum_i \frac{|V_i|}{k} \geq \frac{\epsilon(|x|)t}{16k}$, by the case assumption. The "multiplicative Chernoff bound" implies that if Y is a sum of independent 0-1 random variables, then $\text{Prob}[Y \leq \frac{1}{2}\text{E}[Y]] \leq e^{-\text{E}[Y]/8}$. Applying this here we obtain $\text{Prob}[Y \leq \frac{\epsilon(|x|)t}{32k} \leq e^{-\epsilon(|x|)t}256k \leq \epsilon(|x|)/16$ by Proposition 24.

Case 3. $B_{i,j} \geq B^*_{i,j}/(1+\nu)$ for all entries (i,j) and there are at most $\epsilon(|x|)t/16$ entries with $B_{i,j} > (1+\tau)B^*_{i,j}$.

By Proposition 27, $\hat{z}^*$ is equal to $L(x)$ with probability at least $\frac{1}{2} + \epsilon(|x|)/4$. We show that under the case assumption the probability that $\hat{z} \neq \hat{z}^*$ is at most $\epsilon(|x|)/8$ which will show that $\hat{z}$ is equal to $L(x)$ with probability at least $\frac{1}{2} + \epsilon(|x|)/8$.

We say that a row index $i \in \{1,\ldots,t\}$ is *unsafe* if there is at least one j such that $B_{i,j} > (1+\tau)B^*_{i,j}$ and is *safe* otherwise. We identify the following "bad" events:

B1 The selected row index $\hat{i}$ is unsafe.

B2 The selected $\hat{\lambda}$ satisfies $\hat{\lambda}/D \in [0,2\tau) \cup [1-2\tau,1) \cup \bigcup_{j=1}^{k}[\mu(B^*_{\hat{i},j}) - 2\tau, \mu(B^*_{\hat{i},j}) + 2\tau]$.

We claim that if neither bad event happens then $\hat{z} = \hat{z}^*$. Since [B1] doesn't happen, $\hat{i}$ is safe. Together with the case assumption and the fact that $\nu \leq \tau$ this implies that for each $j \in [k]$, $B_{\hat{i},j} \in [B^*_{\hat{i},j}/(1+\tau), B^*_{\hat{i},j}(1+\tau)]$. Since [B2] doesn't happen, then for all $j \in \{1,\ldots,k\}$, the hypotheses of Proposition 16 hold for $(s,t) = (B_{\hat{i},j}, B^*_{\hat{i},j})$ from which we conclude that for all $j \in \{1,\ldots,k\}$

$$A[\lambda/D]_{\hat{i},j} = \phi((1+\lambda/D)B_{\hat{i},j})) = \phi((1+\lambda/D)B^*_{\hat{i},j})) = A^*[\lambda/D]_{\hat{i},j},$$

which implies that $\hat{z}[\lambda/D] = \hat{z}^*[\lambda/D]$.

So now it will be enough to show that Prob$[B1]$ and Prob$[B2]$ are both at most $\epsilon(|x|)/16$. Prob$[B1]$ is equal to the fraction of unsafe rows, which is at most $\epsilon(|x|)/16$ by the case assumption.

Since λ is chosen uniformly from $\{1,\ldots D\}$, Prob$[B2]$ is the fraction of integers in $\{1,\ldots,D\}$ that belong to the set $[0,2D\tau) \cup [D(1-2\tau), D) \cup \bigcup_{j=1}^{k} [D(\mu(B^*_{\hat{i},j}) - 2\tau), D(\mu(B^*_{\hat{i},j}) + 2\tau)]$

Now we use the fact that the number of integers in an interval $[a,b)$ is less than $b-a+1$ and an interval $[a,b]$ is at most $b-a+1$. Therefore the number of integers in $[0,2D\tau) \cup [D(1-2\tau), D)$ is less than $4\tau D+2$ and the number of integers in each set of the big union is at most $4\tau D+1$. Summing up over all intervals the number of integers in the union is at most $(4k+4)\tau D+k+1$. Since $\tau D \leq 1/4$ by Proposition 24, this is at most $2k+2$. So the probability that λ is in this set is at most $(2k+2)/D$ which is at most $\epsilon(|x|)/16$ by Proposition 24. ◀

6 A Dichotomy

We first make Theorem 14 more concrete by identifying some interesting parameter settings. One natural setting is where the number of queries is polynomial and the advantage is inverse polynomial.

▶ **Theorem 29.** *Suppose L is computable and there is a randomized polynomial-time non-adaptive reduction F from L to $\omega(\log(|q|))$-additively approximating Kolmogorov complexity such that F is polynomially honest and has inverse polynomial advantage. Then $L \in \mathsf{AM} \cap \mathsf{coAM}$.*

Proof. We apply Theorem 14 with $k^M_n = \mathrm{poly}(n)$, $\epsilon(n) = 1/\mathrm{poly}(n)$, and $\alpha(n) = O(\log(n))$. Hence we have $\beta_n(q) = O(\log(n))$. Since F is polynomially honest, we have that $\beta_n(q) = O(\log(q))$ for every $q \in Q^M_x$ for x of length n. Hence by the conclusion of Theorem 14, we have $L \in \mathsf{AM} \cap \mathsf{coAM}$. ◀

Another natural setting is where the number of queries and the advantage are both constant. In this case, we can get a conclusion from a smaller upper bound on the approximation gap.

▶ **Theorem 30.** *Suppose L is computable and there is a computable function $\eta = \omega(1)$ such that there is a randomized polynomial-time non-adaptive reduction F from L to $(2\log(K(q)) + \eta(|q|))$-additively approximating Kolmogorov complexity such that F makes $O(1)$ queries, is weakly honest and has advantage $\Omega(1)$. Then $L \in \mathsf{AM} \cap \mathsf{coAM}$.*

Proof. Let $\lambda : \mathbb{N} \to \mathbb{N}$ be an unbounded function such that F is λ-honest. When applying Theorem 14, we choose $k^M_n = O(1)$ and $\epsilon(n) = \Omega(1)$. We also choose α appropriately so that $\beta_n(q) \geq 2\log(K(q)) + \eta(|q|)$ - this is possible since F is weakly honest, by ensuring $\alpha(\lambda^{-1}) = o(\eta)$. Once more, applying Theorem 14 gives us the desired condlusion. ◀

Theorem 29 supplies the first part of our dichotomy result. We require the following result of Hirahara [28] for the second part of our dichotomy. This is Theorem 6.3 in [28], where it is stated without mentioning the honesty property of the reduction. The proof of Theorem 6.3 in [28] does yield a polynomially honest reduction.

▶ **Theorem 31** ([28]). *For any constant $c \in \mathbb{N}$, there exists an $\mathsf{EXP}^{\mathsf{NP}}$ computable function $G = \{G_n\}$, where G_n maps $n - c\log(n)$ bits to n bits, such that for any $L \in \mathsf{NEXP}$, there is a randomized polynomial-time polynomially honest non-adaptive reduction from L to distinguishing G from uniform.*

▶ **Corollary 32.** *For every constant $c \in \mathbb{N}$ and any $L \in \mathsf{NEXP}$, there is a randomized polynomial-time non-adaptive reduction F from L to $c\log(|q|)$-additively approximating Kolmogorov complexity such that F is polynomially honest and has constant advantage.*

Proof. Let $c \in \mathbb{N}$, and let K' be any oracle that $c\log(|q|)$-additively approximates Kolmogorov complexity. Wlog we assume $c \geq 3$ - note that establishing the conclusion for a constant c also establishes it for any constant $c' \leq c$. Applying Theorem 31, we have that there is a computable function $G = \{G_n\}$, where G_n maps $n - 3c\log(n)$ bits to n bits such that for any $L \in \mathsf{NEXP}$, there is a randomized polynomial-time polynomially honest non-adaptive reduction from L to D, where D is any language distinguishing G from the uniform distribution. We define $D(q) = 1$ if $K'(q) < |q| - 1.5c\log(|q|)$. Note that any output of the generator G can be described with $\log(n) + n - 3c\log(n) + O(1)$ bits, and hence has Kolmogorov complexity $< n - 2.5c\log(n)$ for large enough n. This implies that for every n-bit output y of the generator G_n, $K'(y) < n - 1.5c\log(n)$. On the other hand, with probability at least $1 - 1/n^{c/2}$ over $y \sim U_n$. we have that $K(y) \geq n - 0.5c\log(n)$, and hence $K'(y) \geq n - 1.5c\log(n)$. Thus K' does distinguish the output of G_n from uniform. Since D' can be computed with a single deterministic query to K' of the same length as the input, we have that for any $L \in \mathsf{NEXP}$, there is a randomized polynomial-time polynomially honest non-adaptive reduction from L to K'. Since this holds for any K' that $c\log(|q|)$-additively approximates Kolmogorov complexity, the result follows. ◀

Theorem 29 together with Corollary 32 establishes the dichotomy result in Theorem 3.

7 On Randomized Reductions to K^t

In this section, we show limitations on randomized nonadaptive reductions to K^t for polynomially bounded t. We first need to define the notion of a *hitting set generator*.

▶ **Definition 33.** *Given integers n and $\ell < n$ and rational error parameter $\epsilon \in [0,1]$ with $\epsilon \geq 1/2^n$, a hitting set generator $H_{n,\epsilon}$ for size n with seed length ℓ and error ϵ is a function from $\ell(n)$ bits to n bits, such that for any circuit C_n of size n, if C_n accepts at least an ϵ fraction of inputs of length n, we have that $\Pr_{z\in\{0,1\}^\ell}[C_n(H_{n,\epsilon}(z)) = 1] > 0$. Given a function $\ell : \mathbb{N} \times [0,1] \to \mathbb{N}$, we say that a sequence $H_{n,\epsilon}$ of functions is a polynomial-time computable hitting set generator with seed length ℓ if for each integer n and rational $\epsilon \in [0,1]$ such that $\epsilon \geq 1/2^n$, $H_{n,\epsilon}$ is a hitting set generator for size n with error ϵ and seed length $\ell(n,\epsilon)$, and moreover $H_{n,\epsilon}$ can be computed in time $\mathrm{poly}(n, \log(1/\epsilon))$ when given n and $\lceil\log(1/\epsilon)\rceil$ in unary.*

A key component of our proof is a construction of hitting set generators with optimal dependence on ϵ in the seed length [19].

▶ **Theorem 34** ([19]). *If* E *requires non-deterministic circuits of size* $2^{\Omega(n)}$*, then there is a polynomial-time computable hitting set generator* H *with seed length* $O(\log(n)) + \log(1/\epsilon)$.

It is crucial for our results that the constant factor in front of the $\log(1/\epsilon)$ term is 1; if we could afford an arbitrary constant factor there, we would only need an assumption on hardness for deterministic circuits. We note that the result of [19] is stated for ϵ with a fixed dependence on n, but it is easy to see that their proof works also if $\log(1/\epsilon)$ is given as an independent parameter to the algorithm computing the hitting set generator.

We next use Theorem 34 to argue that under a standard derandomization assumptions, the Kolmogorov complexity of a typical sample from a polynomial-time samplable distributions does not differ by much from the time-bounded Kolmogorov complexity. We require that the time bound when measuring time-bounded Kolmogorov complexity is large enough compared to the time required for sampling.

▶ **Lemma 35.** *Let* $t' : \mathbb{N} \to \mathbb{N}$ *be a polynomially bounded function, and* $\{D_m\}$ *be a sequence of distributions samplable in time* $t'(m)$*, where for each* $m \in \mathbb{N}$*,* D_m *is supported on* m*-bit strings. Suppose that* E *requires non-deterministic circuits of size* $2^{\Omega(n)}$*. Then there is* $t : \mathbb{N} \to \mathbb{N}$ *such that* $t = \text{poly}(t')$ *and* $\Pr_{x \sim D_m}[K^t(x) - K(x) = \omega(\log(m))] = 1/m^{\omega(1)}$.

Proof. Let $\{D_m\}$ be a sequence of distributions, where D_m is supported on m-bit strings, sampled by an algorithm A running in time $t'(m)$. Using the standard simulation of deterministic time by circuit size, A can be simulated by a sequence of circuits $\{C'_m\}$, where C'_m is of size at most $t'(m)^2$ for large enough m. Also, let $f = \omega(\log(m))$ be an arbitrary function.

Let z be a fixed string of length at most $n = 2t'(m)^2$ that is assigned probability ϵ by D_m. We argue that the polynomial-time bounded Kolmogorov complexity of z is at most $\log(1/\epsilon) + O(\log(n))$, under the assumption that E required non-deterministic circuits of size $2^{\Omega(n)}$. Indeed, under the given derandomization assumption, there is hitting set generator $H = \{H_{n,\epsilon}\}$ with seed length $O(\log(n)) + \log(1/\epsilon))$ computable in polynomial time as per Definition 33. Let B be an algorithm that computes H in polynomial time.

We show that there must be at least one output r of $H_{n,\epsilon}$ such that C'_m outputs z on input r. Indeed, consider a circuit C_n that on input r' of length at most n, runs $C'_m(r')$ and accepts iff the answer is z. Since z has probability ϵ according to D_m and C'_m samples from D_m, we have that C_n accepts with probability at least ϵ on a uniformly random input. Since $H_{n,\epsilon}$ is a hitting set generator, we have that at least one output r of the hitting set generator is such that $C'_m(r) = z$.

Let y be the seed of length $O(\log(n)) + \log(1/\epsilon)$ for which $H_{n,\epsilon}(y) = r$. We describe z using the code of B, the string y and descriptions of n and $\lceil \log(1/\epsilon) \rceil$ in binary. This composite description is of size at most $O(\log(n)) + \log(1/\epsilon)$, and z can be computed in time $t = \text{poly}(n, \log(1/\epsilon))$ from this description by running the algorithm B with parameters n and $\log(1/\epsilon)$ and input y to produce a string r, and then running A on r to produce z. Hence $K^t(z) = O(\log(n)) + \log(1/\epsilon)$. Let C be an explicit constant such that $K^t(z) \leq C \log(m) + \log(1/\epsilon)$ for large enough n.

In the next part of our argument, we show that the cumulative probability according to D_m of strings with additive gap at least $f(m)$ between Kolmogorov complexity and t-time bounded Kolmogorov complexity is $1/m^{\omega(1)}$, establishing our result.

Partition all binary strings of length m into $2m$ groups as follows. The group $S_i, i \in [2m-1]$ contains all strings of length m sampled with probability at most $1/2^{i-1}$ and greater than $1/2^i$ from D_m. The group S_m contains all strings of length m sampled with probability at most $1/2^{2m-1}$. We will obtain our upper bound on the likelihood of a large gap between K and K^t complexity by using a union bound on the groups S_i.

Consider any group $S_i, i \in [2m-1]$. Let $g(m) = f(m) - C\log(m)$. Since $f(m) = \omega(\log(m))$, so is $g(m)$. Call a string x of length m i-bad if $x \in S_i$ and $K(x) \leq i - g(m)$. By the upper bound on Komogorov complexity, we have that there are at most $2^{i-g(m)} = 2^i/m^{\omega(1)}$ i-bad strings. Since each i-bad string is in S_i, it has probability at most $1/2^{i-1}$ of being sampled according to D_m, and therefore the cumulative probability of i-bad strings according to D_m is at most $1/2^{i-1} \cdot 2^i/m^{\omega(1)} = 1/m^{\omega(1)}$.

By the argument earlier in the proof, each string in S_i has K^t complexity at most $C\log(m)+\log(1/2^i) = C\log(m)+i$. Hence any string $x \in S_i$ such that $K^t(x)-K(x) \geq f(m)$ is i-bad, and the cumulative probability of all such strings according to D_m is $1/m^{\omega(1)}$. By a union bound over $i \in [2m-1]$, we have that the cumulative probability of all strings with gap at least $f(m)$ between Kolmogorov complexity and t-time bounded Kolmogorov complexity and that occur with probability greater than $1/2^{2m}$ in D_m is $1/m^{\omega(1)}$. Since there are only 2^m m-bit strings, the cumulative probability according to D_m of strings from S_{2m} is at most $2^{-m} = 1/m^{\omega(1)}$. Hence, by another union bound, $\Pr_{x\sim D_m}[K^t(x) - K(x) = \omega(\log(m))] = 1/m^{\omega(1)}$, as desired. ◀

We note that a weaker version of Lemma 35, where the hypothesis is that E requires Σ_2^p-oracle circuits of truly exponential size, is implicit in work of Antunes and Fortnow [16].

We require a couple of other well-known results. The first is a proposition shown using a straightforward padding argument.

▶ **Proposition 36.** *If every tally language in* $\mathsf{NTIME}(n)$ *is in* coNP*, then* $\mathsf{NE} = \mathsf{coNE}$*.*

The second is a standard derandomization result.

▶ **Theorem 37** ([38])**.** *If there is an* $\epsilon > 0$ *such that* E *does not have non-deterministic circuits of size* $2^{\epsilon n}$*, then* $\mathsf{AM} = \mathsf{NP}$*.*

Now we have the tools to state and prove the main result of this section, which also appears as Theorem 6 in the Introduction.

▶ **Theorem 38.** *Suppose that there is a polynomially bounded function* $t' : \mathbb{N} \to \mathbb{N}$ *such that for all large enough* $t = \mathrm{poly}(t')$*, there is a randomized time* t' *non-adaptive reduction that is polynomially honest, has fixed query length and inverse polynomial advantage, from SAT to* $\omega(\log(m))$*-additively approximating* K^t *complexity. Then either* E *has non-deterministic circuits of size* $2^{o(n)}$ *infinitely often, or* $\mathsf{NE} = \mathsf{coNE}$*.*

Proof. Suppose there is a randomized time t' non-adaptive reduction R from SAT to $O(\log(n))$-additively approximating K^t complexity with the properties specified in the statement of the Theorem, where $t = \mathrm{poly}(t')$ and the precise polynomial dependence is to be specified later. We use Lemma 35 to show that under the assumption that E does not have non-deterministic circuits of size $2^{\epsilon n}$ for some $\epsilon > 0$, R also reduces any tally language in $\mathsf{NTIME}(n)$ to $\omega(\log(m))$-additively approximating K complexity, and then apply Theorem 29.

Indeed, let $L \in \mathsf{NTIME}(n)$ be a tally language. By the Cook-Levin theorem, there is a m-reduction from L to SAT running in time $n\,\mathrm{polylog}(n)$. This reduction is polynomially honest and has fixed query length. By composing this reduction with the reduction R in the assumption, we get that there is a randomized time t'' non-adaptive reduction R' from L to $\omega(\log(m))$-additively approximating K^t complexity, where the reduction is polynomially honest and has fixed query length, and where $t'' = t(n\,\mathrm{polylog}(n))$ is polynomially bounded.

Now we apply Lemma 35 to the sequence of distributions D_m sampled by running the randomized reduction R' on input 1^m and outputting a random query [7]. Applying Lemma 35 and a union bound, we have that with probability $1 - 1/m^{\Omega(1)}$ over the randomness r of R', all queries z asked on randomness r have $|K(z) - K^t(z)| = O(\log(m))$, and since the reduction has inverse polynomial advantage, replacing an oracle that $\omega(\log(m))$-additively approximates K^t complexity with one that $\omega(\log(m))$-additively approximates K complexity will preserve the answer of the reduction for each string 1^m. Hence we have that the hypothesis of Theorem 29 is satisfied, and we get that $L \in \mathsf{AM} \cap \mathsf{coAM}$.

Using the assumption that E does not have non-deterministic circuits of size $2^{\epsilon n}$ once more and applying Theorem 37, we have that $L \in \mathsf{NP} \cap \mathsf{coNP}$. Since this is the case for every $L \in \mathsf{NTIME}(n)$, we can use Proposition 36 and conclude that $\mathsf{NE} = \mathsf{coNE}$, as desired. ◀

8 Future Directions

There are two obvious directions to pursue. The first is to extend our negative results to other meta-complexity problems such as MCSP. As mentioned before, [43] unconditionally rule out NP-hardness of MCSP under randomized sublinear-time projections, but nothing seems to be known about less specialized forms of reduction.

The second is to obtain evidence against NP-hardness of meta-complexity problems with respect to randomized adaptive reductions. An anonymous reviewer suggests that it might be possible to use ideas from [17] to obtain interesting consequences from NP-hardness of approximating Kolmogorov reductions with respect to randomized reductions with bounded adaptivity.

References

1 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on np-hardness. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 701–710, 2006.

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.

3 Eric Allender. The new complexity landscape around circuit minimization. In *Proceedings of 14th International Conference on Language and Automata Theory and Applications*, volume 12038 of *Lecture Notes in Computer Science*, pages 3–16. Springer, 2020.

4 Eric Allender, Harry Buhrman, Luke Friedman, and Bruno Loff. Reductions to the set of random strings: The resource-bounded case. *Log. Methods Comput. Sci.*, 10(3), 2014.

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

6 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

7 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

8 Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In *Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 25–32, 2014.

[7] Technically speaking, it might be unclear if Lemma 35 applies here, as the sampled strings do not in general have length m, but the proof of Lemma 35 continues to work as long as the sampled strings have length $m^{\Omega(1)}$

9 Eric Allender, George Davie, Luke Friedman, Samuel Hopkins, and Iddo Tzameret. Kolmogorov complexity, circuits, and the strength of formal theories of arithmetic. *Chic. J. Theor. Comput. Sci.*, 2013, 2013.

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

11 Eric Allender, Joshua A. Grochow, and Cristopher Moore. Graph isomorphism and circuit size. *CoRR*, abs/1511.08189, 2015. `arXiv:1511.08189`.

12 Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing disjunctive normal form formulas and AC0 circuits given a truth table. *SIAM J. Comput.*, 38(1):63–84, 2008.

13 Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. In *International Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 54:1–54:14, 2017.

14 Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. In *International Symposium on Theoretical Aspects of Computer Science* (STACS), pages 21–33, 2015.

15 Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *J. Comput. Syst. Sci.*, 77(1):14–40, 2011.

16 Luis Filipe Coelho Antunes and Lance Fortnow. Worst-case running times for average-case algorithms. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 298–303. IEEE Computer Society, 2009.

17 Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Proceedings of 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 211–220, 2008.

18 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 1st edition, 2009.

19 Sergei Artemenko, Russell Impagliazzo, Valentine Kabanets, and Ronen Shaltiel. Pseudorandomness when the odds are against you. In *Proceedings of the 31st Conference on Computational Complexity*, volume 50 of *LIPIcs*, pages 9:1–9:35. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

20 Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006.

21 Harry Buhrman, Lance Fortnow, Michal Koucký, and Bruno Loff. Derandomizing from random strings. In *Proceedings of the 25th Annual Conference on Computational Complexity*, CCC '10, 2010.

22 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Agnostic learning from tolerant natural proofs. In *Approximation, Randomization, and Combinatorial Optimization*, pages 35:1–35:19, 2017.

23 Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

24 Oded Goldreich and Salil P. Vadhan. On the complexity of computational problems regarding distributions. In Oded Goldreich, editor, *Studies in Complexity and Cryptography*, volume 6650, pages 390–405. Springer, 2011.

25 Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 59–68. ACM, 1986.

26 Dan Gutfreund and Salil P. Vadhan. Limitations of hardness vs. randomness under uniform reductions. In *Proceedings of 12th International Workshop on Randomness and Computation*, volume 5171 of *Lecture Notes in Computer Science*, pages 469–482. Springer, 2008.

27 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 247–258. IEEE Computer Society, 2018.

28 Shuichi Hirahara. Unexpected hardness results for kolmogorov complexity under uniform reductions. In *Proceedings of 52nd Annual Symposium on Theory of Computing*, pages 1038–1051, 2020.

29 Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. Np-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *33rd Computational Complexity Conference, CCC 2018*, pages 5:1–5:31, 2018.

30 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Conference on Computational Complexity* (CCC), pages 18:1–18:20, 2016.

31 Shuichi Hirahara and Osamu Watanabe. On nonadaptive security reductions of hitting set generators. In *Proceedings of 24th International Conference on Randomization and Computation*, volume 176 of *LIPIcs*, pages 15:1–15:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

32 John M. Hitchcock. Limitations of efficient reducibility to the kolmogorov random strings. *Comput.*, 1(1):39–43, 2012.

33 John M. Hitchcock and Aduri Pavan. On the NP-completeness of the minimum circuit size problem. In *Conference on Foundation of Software Technology and Theoretical Computer Science* (FSTTCS), pages 236–245, 2015.

34 Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and ac^0[p]. In *Proceedings of 11th Innovations in Theoretical Computer Science Conference*, volume 151 of *LIPIcs*, pages 34:1–34:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

35 Rahul Ilango. Constant depth formula and partial function versions of MCSP are hard. In *Proceedings of 61st IEEE Annual Symposium on Foundations of Computer Science*, pages 424–433, 2020.

36 Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. Np-hardness of circuit minimization for multi-output functions. In *Proceedings of 35th Computational Complexity Conference*, volume 169 of *LIPIcs*, pages 22:1–22:36. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

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

38 Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

39 Ker-I Ko. On the complexity of learning minimum time-bounded turing machines. *SIAM Journal on Computing*, 20(5):962–986, 1991.

40 Leonid Levin. Universal search problems. *Problems of Information Transmission*, 9(3):115–116, 1973.

41 William J. Masek. Some NP-complete set covering problems. Unpublished Manuscript, 1979.

42 Cody Murray and Ryan Williams. On the (non) NP-hardness of computing circuit complexity. In *Conference on Computational Complexity* (CCC), pages 365–380, 2015.

43 Cody Murray and Ryan Williams. On the (non) np-hardness of computing circuit complexity. *Theory Comput.*, 13(1):1–22, 2017.

44 Michael Saks and Rahul Santhanam. Circuit lower bounds from np-hardness of MCSP under turing reductions. In Shubhangi Saraf, editor, *Proceedings of 35th Computational Complexity Conference*, volume 169 of *LIPIcs*, pages 26:1–26:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

45 Rahul Santhanam. Pseudorandomness and the minimum circuit size problem. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 68:1–68:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

46 Stathis Zachos. Probabilistic quantifiers and games. *J. Comput. Syst. Sci.*, 36(3):433–451, 1988. doi:10.1016/0022-0000(88)90037-2.

A Limitations of Randomized m-Reductions to Approximating Kolmogorov Complexity

In this section, we show that any language that has a low-error randomized polynomial-time m-reduction to a gap version of the Kolmogorov random strings is solvable in Statistical Zero Knowledge.

▶ **Theorem 39.** *Let L be any decidable language such that there is a randomized poly-time m-reduction with error $1/n^{\omega(1)}$ from L to R_K with gap g for some time-constructible $g = \omega(\log(n))$. Then $L \in \mathsf{SZK}$.*

Proof. Let L be a language as in the statement of the theorem. Let f be a randomized poly-time m-reduction with error $1/n^{\omega(1)}$ to R_K with gap g for some $g = \omega(\log(n))$.

We can assume without loss of generality that any query made by f on input x of length n is of length $p(n)$ for some polynomial p. Indeed, let $p(n)$ be an upper bound on the running time of f. We modify the reduction by "padding" any query made by f to length $m = 2p(n)$ as follows. Let c be the constant in the $O(\cdot)$ in Proposition 11. If the query q is of length k, we sample a string $y = r0^{(m-k)/2-2c\log(n)}$ where $r \in \{0,1\}^{(m-k)/2+2c\log(n)}$ is chosen uniformly at random, and output the query qy. Observe that by Proposition 11, with probability $1 - 1/n^{\omega(1)}$, if q is a YES instance of R_K, then qy is a YES instance of R_K, and if q is a NO instance of R_K, then qy is a NO instance of R_K.

For each $x \in \{0,1\}^*$, $f(x)$ is a random variable supported on $\{0,1\}^{2p(|x|)}$. We claim that the following holds. If $x \notin L$, the random variable $f(x)$ has entropy at least $p(|x|) - g(2p(|x|))/2 + 1$, and if $x \in L$, the random variable $f(x)$ has entropy at most $p(|x|) - g(2p(|x|))/2 - 1$.

We first show that the claim implies that $L \in \mathsf{SZK}$, and then establish the claim. To see that the claim implies $L \in \mathsf{SZK}$, we use the fact that Entropy Difference is in SZK. Namely, there is a statistical zero-knowledge protocol that, given a circuit C sampling a distribution over m bits and a parameter s, accepts when the entropy of C is at least $s + 1$ and rejects when the entropy of C is at most $s - 1$. Note that a circuit C sampling $f(x)$ can easily be computed from x. Since SZK is closed under poly-time m-reductions, we have that $L \in \mathsf{SZK}$, by reducing to Entropy Difference with parameter $m/2 - g(m)/2$.

Next we establish the claim. Assume for the sake of contradiction that the claim fails for infinitely many inputs. Let $\{n_i\}$ be an infinite sequence of numbers such that the claim fails for at least one input of length n_i for each i, and let x_i be the first input of length n_i for which the claim fails. Note that $K(x_i) = O(\log(n_i))$. The reason is that we can describe x_i by a program which has n_i in binary encoded into it, and then searches for the first string z of length n_i such that the claim fails for z, i.e., either $z \in L$ and the entropy of $f(z)$ is greater than $p(|z|) - g(2p(|z|)/2 - 1$, or $z \notin L$ and the entropy of $f(z)$ is smaller than $p(|z|) - g(2p(|z|))/2 + 1$.

Let i be large enough and x_i be the first string of length n_i for which the claim fails. Let $m = 2p(n_i)$. Either $x_i \in L$ and the entropy of $f(x_i)$ is greater than $m/2 - g(m)/2 - 1$, or $x_i \notin L$ and the entropy of $f(x_i)$ is smaller than $m/2 - g(m)/2 + 1$. We show that either case leads to a contradiction.

In the first case, the random variable $f(x_i)$ has entropy greater than $m/2 - g(m)/2 - 1$. Then with probability at least $1/m$ over $y \in \{0,1\}^m$ sampled from $f(x_i)$, $K(y) > m/2 - g(m)$. Indeed, if not, the entropy of $f(x_i)$ is at most $1/m * m + (1 - 1/m) * (m/2 - g(m)) \leq m/2 - g(m) + 1 < m/2 - g(m)/2 - 1$, which contradicts the lower bound on the entropy of $f(x_i)$. But if $K(y) > m/2 - g(m)$ with probability at least $1/m$, the reduction cannot be correct for $x_i \in L$, as correctness of the reduction would imply $K(y) \leq m/2 - g(m)$ with probability $1 - 1/n^{\omega(1)}$ (as we could answer NO for all queries y with $K(y) > m/2 - g(m)$).

In the second case, the random variable $f(x_i)$ has entropy at most $m/2 - g(m)/2 + 1$. We show that in this case, with probability $1/n^{\Omega(1)}$, y sampled from $f(x_i)$ has $K(y) < m/2$, which again contradicts the correctness of the reduction.

We show that with probability $1/n^{\Omega(1)}$ over y sampled from $f(x_i)$, $K(y|x_i) < m/2 - \omega(\log(m))$, from which the previous line follows using Proposition 11 and the fact that $K(x_i) = O(\log(n_i)) = O(\log(m))$. Indeed let k be the entropy of $f(x_i)$ and let $y_1 \dots y_{2^{k+\log(m)}}$ be the first $2^{k+\log(m)}$ strings in the support of $f(x_i)$ in order of decreasing probability, and let Y be the set of these strings. Conditioned on x_i, each such string can be described with a program of size $k + \log(m) + O(\log(m))$, which is $m/2 - \omega(\log(m))$, using the facts that $k \leq m/2 - g(m)/2 + 1$ and $g(m) = \Omega(\log(m))$. We claim that with probability at least $1/m$, y sampled from $f(x_i)$ belongs to Y. Indeed, if not, we have that with probability greater than $1 - 1/m$, y sampled from $f(x_i)$ has probability at most $2^{-(k+\log(m))}$, and hence the contribution to the entropy of $f(x_i)$ from such strings is at least $(1 - 1/m)(k + \log(m)) > k$, which is impossible.

Thus we have that with probability $1/m$, y sampled from $f(x_i)$ belongs to Y, and that each string in Y has Kolmogorov complexity smaller than $m/2$. But this contradicts the correctness of the reduction, as x_i is a NO instance, and hence with probability $1 - 1/n^{\omega(1)}$, y sampled from $f(x_i)$ should have $K(y) \geq m/2$ (as otherwise we could answer YES for all queries y with $K(y) < m/2 - g(m)$). ◀