Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 198 (2015)

Limits of Minimum Circuit Size Problem as Oracle

Shuichi Hirahara*
The University of Tokyo

Osamu Watanabe†
Tokyo Institute of Technology

Abstract

The Minimum Circuit Size Problem (MCSP) is known to be hard for statistical zero knowledge via a BPP-Turing reduction (Allender and Das, 2014), whereas establishing NP-hardness of MCSP via a polynomial-time many-one reduction is difficult (Murray and Williams, 2015) in the sense that it implies ZPP $\neq$ EXP, which is a major open problem in computational complexity.

In this paper, we provide strong evidence that current techniques cannot establish NP-hardness of MCSP, even under polynomial-time Turing reductions or randomized reductions: Specifically, we introduce the notion of *oracle-independent reduction* to MCSP, which captures all the currently known reductions. We say that a reduction to MCSP is oracle-independent if the reduction can be generalized to a reduction to MCSP^A for any oracle A, where MCSP^A denotes an oracle version of MCSP. We prove that no language outside P is reducible to MCSP via an oracle-independent polynomial-time Turing reduction. We also show that the class of languages reducible to MCSP via an oracle-independent randomized reduction that makes at most one query is contained in $\mathsf{AM} \cap \mathsf{coAM}$. Thus, NP-hardness of MCSP cannot be established via such oracle-independent reductions unless the polynomial hierarchy collapses.

We also extend the previous results to the case of more general reductions: We prove that establishing NP-hardness of MCSP via a polynomial-time *nonadaptive* reduction implies ZPP $\neq$ EXP, and that establishing NP-hardness of *approximating circuit complexity* via a polynomial-time *Turing* reduction also implies ZPP $\neq$ EXP. Along the way, we prove that approximating Levin's Kolmogorov complexity is *provably* not EXP-hard under polynomial-time Turing reductions, which is of independent interest.

1 Introduction

The Minimum Circuit Size Problem (MCSP) asks, given a truth-table $T \in \{0,1\}^{2^n}$ and a size-parameter s, whether there exists a circuit on n variables of size at most s whose truth-table is T. Although it is easy to see that MCSP is in NP, MCSP is not known to be NP-hard.

MCSP is closely related to circuit complexity by its definition, and hence it is one of the central problems in computational complexity. There are a number of formal connections from the complexity of MCSP to important open problems of computational complexity: for example, if $\mathsf{MCSP} \in \mathsf{P}$ then $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{P/poly}$ [14]; if $\mathsf{MCSP} \in \mathsf{coNP}$ then MA can be derandomized ($\mathsf{MA} = \mathsf{NP}$) [1]. Therefore, it is important to determine the structural complexity of MCSP.

While there is substantial evidence that MCSP is not tractable in the sense that $\mathsf{MCSP} \notin \mathsf{BPP}$, it remains open whether MCSP is the hardest problem in NP, that is, NP-hard or not. In this paper, we will discuss why it is so difficult to establish NP-hardness of MCSP. We note that, when

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

ISSN 1433-8092

discussing relative hardness of a problem, there are several types of reductions. Our main focus will be general and powerful reductions such as polynomial-time Turing reductions and randomized reductions. That is, what problems (*e.g.,* SAT) can we solve by using MCSP as an oracle?

1.1 Background

In the seminal paper by Kabanets and Cai [14], on one hand, they exhibited evidence that MCSP is intractable; namely, they proved that factoring Blum integers can be solved faster than any known algorithms, assuming that $\mathsf{MCSP} \in \mathsf{P}$. On the other hand, they also proved that establishing NP-hardness of MCSP is difficult: if MCSP is NP-hard under a certain type of restricted polynomial-time reductions, then some circuit lower bounds hold (and, in particular, $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$); thus, establishing NP-hardness of MCSP (under the restricted reductions) is at least as difficult as proving $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$. To summarize, MCSP is "harder" than factoring Blum integers, whereas establishing NP-hardness is difficult.

These two sides have been significantly pushed forward. On the positive side on hardness of MCSP, Allender, Buhrman, Koucký, van Melkebeek and Ronneburger [1] proved cryptographic problems, such as the discrete logarithm problem and integer factoring, can be solved in $\mathsf{BPP}^{\mathsf{MCSP}}$ (*i.e.,* these problems reduce to MCSP under BPP-Turing reductions). Allender and Das [2] strengthened these results by showing that every language in statistical zero knowledge is in $\mathsf{BPP}^{\mathsf{MCSP}}$.

The negative side on hardness of MCSP was considerably strengthened by Murray and Williams [17]. They showed that, if MCSP is NP-hard under polynomial-time many-one reductions, then $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{P/poly}$ (and, in particular, $\mathsf{EXP} \neq \mathsf{ZPP}$), which is one of the central open problems in computational complexity. Thus, it is difficult to establish NP-hardness of MCSP under (general) polynomial-time many-one reductions. Moreover, they showed that, under local reductions (*i.e.,* that cannot look at a whole input), MCSP is *provably* not hard even for PARITY. Allender, Holden, and Kabanets [4] showed similar results for an oracle version of MCSP. For example, they showed that PSPACE is provably not reducible to $\mathsf{MCSP}^{\mathsf{QBF}}$ via a log space reduction; here, for an oracle A, MCSP^A denotes a problem of asking the smallest size of a circuit with A-oracle gates.

Thus, the current status of our understanding of MCSP is as follows: under the restricted reductions (*e.g.,* local reductions), MCSP is not "hard" at all, which suggests that such restricted reductions are insufficient to discuss the relative hardness of MCSP; under polynomial-time many-one reductions, it is difficult to establish NP-hardness of MCSP; nevertheless, BPP-Turing reductions to MCSP are powerful enough to solve every problem in statistical zero knowledge.

Therefore, it is very interesting to investigate whether one can push the positive side and establish NP-hardness of MCSP, or else the negative side can be pushed: More specifically, can we prove NP-hardness of MCSP under general reductions, such as BPP reductions? Can we extend the results of Murray and Williams [17] (as well as [4]) to more general reductions?

1.2 Oracle-independent Reductions

In this paper, we push the negative side further, and show that current techniques cannot be easily extended to show NP-hardness of MCSP. Specifically, we observe that current techniques do not rely on any inherent property of MCSP and instead rely on common properties that MCSP^A shares for an arbitrary oracle A. We thus introduce the notion of *oracle-independent reductions* to MCSP and then give upper bounds on classes of languages that reduces to MCSP via such reductions. We say that a reduction to MCSP is *oracle-independent* if the reduction can be generalized to MCSP^A

for an arbitrary oracle A. In other words, the reduction exploits only properties common to MCSP^A for any oracle A (instead of unrelativizing properties of MCSP).

All the known efficient reductions to MCSP are oracle-independent. The main ingredient used by almost all the reductions [1, 2] is the construction from a one-way function to a pseudorandom generator by Håstad, Impagliazzo, Levin, and Luby [12]: Specifically, since the output of a pseudorandom (function) generator is efficiently computable, the output regarded as a truth-table has significantly low circuit complexity, compared to that of a truth-table chosen from a uniform distribution. Thus, MCSP constitutes a statistical test that distinguishes a pseudorandom distribution from a uniform distribution, which enables us to break a one-way function on average, thanks to [12]. This argument exploits only the fact that MCSP constitutes a statistical test. It is easy to see that an oracle version MCSP^A can also constitute a statistical test, and hence such reductions are oracle-independent.

Recently, new types of reductions to MCSP that do not rely on breaking a one-way function have been developed by Allender, Grochow, and Moore [3]. Based on new ideas, they showed that a certain graph isomorphism problem is reducible to MCSP via a randomized reduction with zero-sided error. We will see that their reductions are also oracle-independent.

A high-level reason why these reductions are oracle-independent is as follows: We are prone to rely on the fact that a randomly chosen truth-table requires high circuit complexity, because it is in general difficult to obtain a circuit lower bound on an explicit function. The fact that many truth-tables require high circuit complexity remains unchanged for any oracle version MCSP^A, and hence a reduction that only exploits this fact (as a circuit lower bound) is inevitably oracle-independent.

We provide strong evidence that NP-hardness of MCSP cannot be shown via such oracle-independent reductions. For deterministic reductions, we prove that nothing interesting is reducible to MCSP via an oracle-independent reduction:

Theorem 1. *No language outside* P *can reduce to* MCSP *under polynomial-time Turing oracle-independent reductions. In other words, if a language L polynomial-time-Turing-reduces to* MCSP^A *for any oracle A, then $L \in \mathsf{P}$; it can be also simply stated as*

$$\bigcap_A \mathsf{P}^{\mathsf{MCSP}^A} = \mathsf{P}.$$

In contrast to previous work [14, 17, 4] which shows that NP-hardness of MCSP implies surprising consequences (*e.g.,* $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$), we emphasize that this theorem gives us an inherent limitation of a deterministic oracle-independent reduction. One implication is that NP-hardness of MCSP cannot be shown via a deterministic oracle-independent reduction unless $\mathsf{P} = \mathsf{NP}$.

We note that this precisely captures the limit of what we can deterministically reduce to MCSP. Indeed, currently no (nontrivial) deterministic reduction to MCSP is known at all. The theorem suggests one reason behind this fact: in order to construct a deterministic reduction to MCSP, we need to use a property of MCSP that cannot be generalized to MCSP^A for all A, which appears very difficult due to our few knowledge about nonrelativizing circuit lower bounds.

It should be also noted[1] that Theorem 1 implies that there exists an oracle A such that $\mathsf{MCSP} \not\leq^p_T \mathsf{MCSP}^A$ (unless $\mathsf{MCSP} \in \mathsf{P}$). At first glance (mainly due to its notation), it might be counterintuitive that an oracle version MCSP^A becomes "easier" than MCSP. The point is that the oracle A in the notation MCSP^A refers to the fact that a circuit that is minimized has oracle access to A, but this does not necessarily increase the computational difficulty of minimizing such an A-oracle circuit.

[1] This observation was given by one of the referees of CCC 2016 in the review report.

Indeed, we exploit this fact to prove Theorem 1. Roughly speaking, for any oracle-independent reduction to MCSP, we adversarially choose an oracle A so that any query that the reduction makes has circuit complexity of $O(\log n)$. Specifically, let $T_1, \ldots, T_{n^{O(1)}}$ be the truth-tables queried by the reduction (on some computation path); we encode these truth-tables into A so that the truth-table of $A(i, \text{-})$ is equal to T_i for any i. For this oracle, the reduction cannot query any truth-table that has high circuit complexity (relative to oracle A) because the size of the circuit that outputs $A(i, x)$ on input x is $O(\log n)$ for any i. We then simulate the reduction by exhaustively[2] search small circuits of size up to $O(\log n)$.

We also prove that even randomized oracle-independent reduction is not sufficient to establish NP-hardness of MCSP:

Theorem 2. *If a language L is reducible to* MCSP *via an oracle-independent randomized reduction with negligible error that makes at most one query, then $L \in \mathsf{AM} \cap \mathsf{coAM}$. In other words,*

$$\bigcap_A \mathsf{BPP}^{\mathsf{MCSP}^A[1]} \subseteq \mathsf{AM} \cap \mathsf{coAM}.$$

Here, $\mathsf{BPP}^{B[1]}$ denotes the class of languages reducible to an oracle B via a randomized reduction with negligible error that makes at most one query.

In particular, $\bigcap_A \mathsf{BPP}^{\mathsf{MCSP}^A[1]}$ does not contain NP unless $\mathsf{NP} \subseteq \mathsf{coAM}$ (and in particular the polynomial hierarchy collapses [7]). Therefore, it is impossible to establish NP-hardness of MCSP via such reductions (unless the polynomial hierarchy collapses).

Oracle-independent Reductions vs. Relativization

We note that an oracle-independent reduction is different from simple relativization. In a relativization setting, Ko [15] showed the existence of a relativized world where MCSP is an NP-intermediate problem: MCSP is neither in coNP nor is NP-complete under polynomial-time Turing reductions. Specifically, he constructed an oracle A such that NP^A is not contained in $\mathsf{P}^{\mathsf{MCSP}^A, A}$, thereby showing a relativized world where MCSP cannot be NP-hard under polynomial-time Turing reductions. This shows the computational limit of MCSP in a relativized world.

In contrast, we discuss the computational limit of MCSP in a *real world* when MCSP is used by oracle-independent reductions. Technically, by exploiting the fact that NP-machines have an oracle access, Ko [15] constructed an oracle A so that some NP^A-computation would go beyond the class $\mathsf{P}^{\mathsf{MCSP}^A, A}$. On the other hand, we construct an oracle A so that $\mathsf{P}^{\mathsf{MCSP}^A}$-computation cannot be strong; in fact, it is essentially the same as P.

1.3 Reductions to MCSP Imply Separations of Complexity Classes

We also extend the results of Murray and Williams [17] to the case of polynomial-time nonadaptive reductions and polynomial-time Turing reductions. In the former case, we prove that the same (in fact, slightly stronger) consequence can be obtained:

[2] When the "size" of a circuit refers to the number of its wires, we cannot enumerate all such circuits in polynomial time since there are $O(\log n)^{O(\log n)} = n^{O(\log \log n)}$ possible circuits of size less than $O(\log n)$, which gives only a weak upper bound. We will thus regard the "size" of a circuit as its description length, and also require that we can encode a truth-table into an oracle efficiently.

Theorem 3. *It holds that* $\mathsf{P}_{||}^{\mathsf{MCSP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$ *(unconditionally). As a consequence, if* MCSP *is* NP*-hard via a polynomial-time nonadaptive reduction, then* $\mathsf{P}_{||}^{\mathsf{NP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$.

Here, $\mathsf{P}_{||}^{\mathsf{MCSP}}$ denotes the class of languages reducible to MCSP via a polynomial-time nonadaptive reduction.

Our proof is based on the firm links between circuit complexity and resource-bounded Kolmogorov complexity, which were established by a line of work [1, 5]. In fact, the proof is so simple that we can include a proof sketch here: Allender, Koucký, Ronneburger and Roy [5] showed that Levin's Kolmogorov complexity [16] (denoted by Kt) is polynomially related to circuit complexity if and only if $\mathsf{EXP} \subseteq \mathsf{P/poly}$; thus, assuming that $\mathsf{EXP} \subseteq \mathsf{P/poly}$, circuit complexity is essentially equal to Kt-complexity. Moreover, it is well-known that $\mathsf{EXP} \neq \mathsf{P}_{||}^{\mathrm{Kt}}$ (since a polynomial-time algorithm cannot output any strings of high Kt-complexity). Thus, assuming that $\mathsf{EXP} \subseteq \mathsf{P/poly}$, we also have $\mathsf{EXP} \neq \mathsf{P}_{||}^{\mathsf{MCSP}}$. This implies that $\mathsf{EXP} \neq \mathsf{P}_{||}^{\mathsf{MCSP}} \cap \mathsf{P/poly}$ (as otherwise we may assume $\mathsf{EXP} \subseteq \mathsf{P/poly}$). Therefore, at the core of the proof of the unconditional separation in Theorem 3 is $\mathsf{EXP} \neq \mathsf{P}_{||}^{\mathrm{Kt}}$.

Now we would like to extend the argument above into the case of polynomial-time Turing reductions. Unfortunately, we could not prove $\mathsf{EXP} \neq \mathsf{P}^{\mathrm{Kt}}$ (and this is an open problem since [1]). Nevertheless, we prove that a promise problem of approximating Kt within additive error $\omega(\log n)$ is not EXP-hard under polynomial-time Turing reductions, which is of independent interest:

Theorem 4. *For any nondecreasing function* $g(n) = \omega(\log n)$*, let* $\mathrm{Gap}_g\mathrm{Kt}$ *denote a promise problem that asks for approximating* $\mathrm{Kt}(x)$ *within additive error* $g(|x|)$ *on input* x*. Then,* $\mathsf{EXP} \neq \mathsf{P}^{\mathrm{Gap}_g\mathrm{Kt}}$.

We note that, for a fixed exponential time $t(n) \geq 2^{n^2}$, Buhrman and Mayordomo [8] proved that K^t is not EXP-hard under polynomial-time Turing reductions. Here, K^t denotes resource-bounded Kolmogorov complexity such that a universal Turing machine that outputs x is required to run in time $t(|x|)$.

Now we can translate the property of Kt-complexity into that of MCSP, under the assumption that $\mathsf{EXP} \subseteq \mathsf{P/poly}$. As a consequence, we obtain:

Theorem 5. *Let* $\mathrm{Gap}^k\mathrm{MCSP}$ *be a promise problem that asks for approximating the logarithm of circuit complexity within a factor of* k*. Then, there exists a constant* $k \geq 1$ *such that* $\mathsf{EXP} \neq \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}} \cap \mathsf{P/poly}$*. In particular, if a language* L *is reducible to* $\mathrm{Gap}^k\mathrm{MCSP}$ *via a polynomial-time Turing reduction for all* $k \geq 1$*, then* $\mathsf{P}^L \cap \mathsf{P/poly} \neq \mathsf{EXP}$.

In particular, establishing NP-hardness of $\mathrm{Gap}^k\mathrm{MCSP}$ via a polynomial-time Turing reduction requires separating $\mathsf{P}^{\mathsf{NP}} \cap \mathsf{P/poly}$ from EXP.

Interestingly, as observed in [5], the BPP-reductions of [1, 2] are extremely robust in terms of approximation. Specifically:

Theorem 6 (Analogous to [5, Theorem 19])**.** *For all* $k \geq 1$*, every language in statistical zero knowledge is reducible to* $\mathrm{Gap}^k\mathrm{MCSP}$ *via a* BPP*-Turing reduction.*

These two results exhibit a striking contrast between BPP-reductions and polynomial-time Turing reductions: BPP-reductions enable us to base hardness of approximating circuit complexity on hardness of statistical zero knowledge, whereas derandomizing the BPP-reduction requires a separation of complexity classes.

Organization

The rest of the paper is organized as follows. In Section 2, we introduce some notation and the definition of circuit complexity. In Section 3, we observe that the known reductions to MCSP are oracle-independent. We prove Theorems 1 and 2 in Sections 4 and 5, respectively. In Section 6, we extend the results of Murray and Williams [17] into the case of more general reductions.

2 Preliminaries

Since we need to specify an exact definition of circuit complexity in order to discuss some subtle details, we specify how to encode two strings into one string:

Definition. *For two strings $x, y \in \{0,1\}^*$, define the* pairing function *as* $\langle x, y\rangle := 1^{|x|}0xy$.

We often write (x, y) instead of $(\langle x, y\rangle)$. We also abbreviate $\langle x, \langle y, z\rangle\rangle$ as $\langle x, y, z\rangle$. Note that $|\langle x, y\rangle| = 2|x| + |y| + 1$.

An oracle A is a subset of strings (*i.e.*, $A \subseteq \{0,1\}^*$). We identify a subset A of strings with its characteristic function $A\colon \{0,1\}^* \to \{0,1\}$. When we use diagonalization arguments, it is convenient to have the notion of *finite oracle*:

Definition. 1. *We say that A_0 is a* finite oracle *if $A_0\colon \{0,1\}^* \to \{0,1,\perp\}$ and $A_0(x) = \perp$ for all but finitely many strings $x \in \{0,1\}^*$, where $\perp$ means "undefined."*

2. *For an oracle $A \subseteq \{0,1\}^*$ and a finite oracle A_0, we say that A is* consistent *with A_0 if $A(x) = A_0(x)$ for any $x \in \{0,1\}^*$ such that $A_0(x) \neq \perp$.*

3. *Similarly, for $l \in \mathbb{N}$, we say that A and A_0 are* consistent up to length l *if it holds that $A(x) = 1$ if and only if $A_0(x) = 1$ for all strings $x \in \{0,1\}^*$ of length at most l.*

For a nonnegative integer $n \in \mathbb{N}$, we write $[n] := \{1, \cdots, n\}$. For a string $x \in \{0,1\}^n$ and $i \in [n]$, we denote by x_i the ith bit of x. We also denote by $\underline{i}_n$ an integer i padded to length n. More specifically:

Definition. *For $n \in \mathbb{N}$ and $i \in [2^n]$, let $\underline{i}_n$ denote the ith string of $\{0,1\}^n$ in the lexicographic order.*

For a set R, we write $r \in_R R$ to indicate that r is a random sample from the uniform distribution on R. For a distribution $\mathcal{D}$, we write $r \sim \mathcal{D}$ to indicate that r is a random sample from $\mathcal{D}$.

2.1 Definition of Circuit Size

Throughout this paper, we regard the description length of a circuit as its size. Thus, it is convenient to define the size of a circuit in terms of Kolmogorov complexity.

Definition. *Let U be a Turing machine. The* Kolmogorov complexity $\mathrm{K}_U(x)$ *of a string $x \in \{0,1\}^*$ with respect to U is defined as* $\mathrm{K}_U(x) := \min\{\, |d| \mid U(d) = x \,\}$.

While we follow this standard definition, we use Kolmogorov complexity in a somewhat non-standard way for discussing circuit complexity. We assume that a string x for which we consider its Kolmogorov complexity is a truth-table of a Boolean function. Thus, $|x|$ is 2^n for some $n \in \mathbb{N}$. We use a circuit interpreter for U instead of a universal Turing machine. In particular, for technical reasons, throughout this paper we will use a specific interpreter I that is defined below.

We first fix our standard (oracle) circuit interpreter. We assume any standard way to encode circuits by binary strings. Note that a circuit may be an oracle circuit that can use oracle gates outputting $A(z)$ for a given input z to the gate when a circuit is used with oracle A. Let I_0 denote a circuit interpreter for this encoding: that is, for any oracle A and a given description d of an oracle circuit C, the interpreter $I_0^A(d)$ yields the truth-table of C^A. (Thus, $|I_0^A(d)| = 2^n$ for some n and $I_0^A(d) = C^A(\underline{1}_n) \cdots C^A(\underline{2^n}_n)$.)

We will use the following facts that the standard circuit interpreter I_0^A should have:

1. $I_0^A(d)$ is computable in time polynomial in $|d|$ and $|I_0^A(d)|$, given oracle access to A.
2. For all but finitely many truth-tables $T \in \{0,1\}^*$ (where $|T|$ is a power of 2), there exists a circuit description of size less than $|T|^2$: that is, $\mathrm{K}_{I_0}(T) < |T|^2$.
3. Any oracle circuit C whose description length is at most m cannot query to an oracle any string of length greater than m. Thus, the output of C^A only depends on the membership in A of strings of length at most m.

We modify the standard circuit interpreter I_0 so that we can describe some type of circuits succinctly. For any $n \in \mathbb{N}$ and $d \in \{0,1\}^*$, let $C_{n,d}^A(x)$ be an oracle circuit that computes $A(x,d)$ (*i.e.*, $A(\langle x,d\rangle)$) for a given input $x \in \{0,1\}^n$, by using a single oracle gate with input $\langle x,d\rangle$.

Definition. *Define an interpreter I^A as follows:*

$$
\begin{aligned}
I^A(0d) &:= I_0^A(d), \\
I^A(1^n, d) &:= I_0^A(C_{n,d}^A) = A(\underline{1}_n, d)A(\underline{2}_n, d) \cdots A(\underline{2^n}_n, d),
\end{aligned}
$$

for any $n \geq 1$ and $d \in \{0,1\}^*$. For the other strings d (e.g., $d = 1101$), leave $I^A(d)$ undefined.

For $A = \varnothing$, we write I instead of $I^\varnothing$.

Remark. 1. Recall that $\langle 1^n, d\rangle = 1^n 0 1^n d$; hence I^A is well-defined. Also, the definition of I^A ensures that the description length of a circuit $C_{n,d}^A$ is at most $|\langle 1^n, d\rangle| = 2n + |d| + 1$, which is exactly equal to the length of a query $\langle \underline{i}_n, d\rangle$ to oracle A.

2. For $A = \varnothing$, we have $\mathrm{K}_I(x) = \mathrm{K}_{I_0}(x) + 1$ for any $x \in \{0,1\}^* \setminus \{0\}^*$; hence, there is essentially no difference between our circuit complexity measure $\mathrm{K}_I(x)$ and a standard description length $\mathrm{K}_{I_0}(x)$. In particular, the results of Section 6 hold under any standard circuit complexity (*e.g.,* that counts the number of gates or wires).

3. For a general oracle A, since we assumed that the circuit $C_{n,d}^A$ can be described succinctly, we cannot guarantee that minimizing our complexity measure K_{I^A} is computationally equivalent to minimizing standard circuit complexity. However, all of the previous work (*e.g.,* [15, 4]) that we are aware of holds under our encoding scheme.

We define the minimum oracle circuit size problem MCSP^A by using I^A as a circuit interpreter:

Definition. *The* minimum oracle circuit size problem MCSP^A *relative to an oracle $A \subseteq \{0,1\}^*$ takes a truth-table $T \in \{0,1\}^*$ and a size-parameter $s \in \mathbb{N}$, and decides if $\mathrm{K}_{I^A}(T) \leq s$.*

3 Why Are the Known Reductions Oracle-independent?

In this section, we argue that the known reductions to MCSP are oracle-independent. We observe that the existing reductions only exploit (as a circuit lower bound) the fact that many truth-tables require high (unrelativized) circuit complexity. Indeed, in the case of the reductions [1, 2] that rely on breaking a one-way function, the following holds:

Theorem 7 (Allender and Das [2]; see also [5, 1])**.** *Let $\epsilon \in (0,1)$ be a constant and let B be an oracle of polynomial density such that $\mathrm{K}_I(x) \geq |x|^\epsilon$ for any $x \in B$ (i.e., B is a statistical test that accepts "random" strings) . Then, every language in statistical zero knowledge is reducible to B via a* BPP*-reduction.*

Here, we say that an oracle B is of polynomial density if there exists a polynomial p such that $\Pr_{x \in_R \{0,1\}^n}[x \in B] \geq 1/p(n)$ for any $n \in \mathbb{N}$.

It is easy to see that such an oracle B can be computed, given oracle access to MCSP: indeed, define $B := \{ x \in \{0,1\}^* \mid \mathrm{K}_I(x) \geq |x|^{1/2} \}$; it is obvious that $B \in \mathsf{P}^{\mathsf{MCSP}}$; moreover, since there are at most $2^{\sqrt{n}+1}$ strings that have circuit complexity at most $\sqrt{n}$ for any $n \in \mathbb{N}$, almost all strings of length n are in B. Therefore, every language in statistical zero knowledge is reducible to MCSP via a BPP-reduction.

This argument is still valid in the case of an oracle version MCSP^A: indeed, we may define an oracle B_A as $\{ x \in \{0,1\}^* \mid \mathrm{K}_{I^A}(x) \geq |x|^{1/2} \}$ ($\in \mathsf{P}^{\mathsf{MCSP}^A}$); since $\mathrm{K}_I(x) \geq \mathrm{K}_{I^A}(x)$ for any $x \in \{0,1\}^*$, the hypothesis of the theorem remains satisfied.

Next, we show that an oracle-independent one-query reduction to MCSP allows us to convert a randomized algorithm with two-sided error into a randomized algorithm with zero-sided error. Moreover, the error probability is negligible.

Theorem 8 (Kabanets and Cai [14])**.** $\mathsf{BPP} \subseteq \bigcap_A \mathsf{ZPP}^{\mathsf{MCSP}^A[1]}$.

Proof Sketch. Pick a truth-table T uniformly at random. By making a query to MCSP^A, check if $\mathrm{K}_{I^A}(T) = n^{\Omega(1)}$. (Note that this also implies that $\mathrm{K}_I(T) = n^{\Omega(1)}$.) Now, if we successfully found a truth-table T that requires high circuit complexity, then we can use the pseudorandom generator by Impagliazzo and Wigderson [13] to derandomize a BPP computation. See [14] for the details. □

Finally, we observe that the new reductions by Allender, Grochow, and Moore [3] are oracle-independent. In fact, their reductions are not known to work under a usual definition of circuit size; instead, they presented reductions to *a* minimum circuit size problem, where "circuit size" here refers to KT-complexity. Let us recall KT-complexity briefly:

Definition (KT-complexity [1])**.** *Fix a universal (oracle) Turing machine U. For an oracle A, the KT^A-complexity of a string x is defined as*

$$\mathrm{KT}^A(x) := \min\{\, |d| + t \mid\ U^{A,d}(i) = x_i \textit{ in } t \textit{ steps for all } i \in [|x|+1] \ \}.$$

Here, $x_{|x|+1}$ is defined as $\perp$ (a stop symbol).

It is known that KT^A-complexity is polynomially related to circuit complexity relative to A; hence, we may regard KT^A as a version of circuit complexity. In order to capture KT-complexity by our notation, we define a circuit interpreter I_0^A as follows: On input $1^t 0 d$, run the universal

Turing machine $U^{A,d}(i)$ for each $i \geq 1$ one by one in time at most t. Let n be the minimum i such that $U^{A,d}(i)$ outputs $\perp$. Output the concatenation of $U^{A,d}(1), \cdots, U^{A,d}(n-1)$. This definition ensures that $\mathrm{K}_{I_0^A}(x) = \mathrm{KT}^A(x) + 1$, and that $\mathrm{K}_{I^A}(x) \leq \mathrm{K}_{I_0^A}(x) + 1 = \mathrm{KT}^A(x) + 2$.

For this particular interpreter I^A, we prove:

Theorem 9 (Allender, Grochow, and Moore [3])**.** *For any oracle A, the rigid graph isomorphism problem is reducible to MCSP^A via a one-query BPP-reduction.*

Proof Sketch. We only observe why their reduction still works for MCSP^A, where A denotes an arbitrary oracle A. See [3] for the details.

Given two graphs (G_0, G_1), they constructed a string x' whose length is a power of 2 and a threshold θ that satisfy the following: If the graphs are isomorphic, then $\mathrm{KT}(x') \ll \theta$ with probability 1. If the graphs are rigid and not isomorphic, then x' contains information about a uniformly chosen random string of length at least θ, and hence $\mathrm{KT}(x') \geq \mathrm{K}_U(x') \gg \theta$ with high probability. (Here, $\mathrm{K}_U(x')$ denotes the *time-unbounded* Kolmogorov complexity.)

Now consider an arbitrary oracle A. We claim that the rigid graph isomorphism problem reduces to checking if $(x', \theta) \in \mathsf{MCSP}^A$. Suppose that the graphs are isomorphic; in this case, we have $\mathrm{K}_{I^A}(x') \leq \mathrm{KT}^A(x') + 2 \leq \mathrm{KT}(x') + 2 \ll \theta$. On the other hand, suppose that the graphs are rigid and not isomorphic. Since x' contains information about a uniformly chosen random string, an information-theoretic argument shows that $\mathrm{K}_{U^A}(x') \gg \theta$ with high probability (even relative to A). By the universality of U, we have $\mathrm{K}_{U^A}(x') \leq \mathrm{K}_{I^A}(x') + O(1)$. Therefore, $\mathrm{K}_{I^A}(x') \geq \mathrm{K}_{U^A}(x') - O(1) \gg \theta$. $\square$

To summarize, on one hand, relativization does not increase circuit complexity ($\mathrm{K}_{I^A}(x') \leq \mathrm{K}_I(x')$); on the other hand, we are prone to rely on the fact that a uniformly chosen random string requires high circuit complexity, which remains true for any MCSP^A.

We mention that, for a specific oracle A, an efficient reduction to MCSP^A is known. Allender, Buhrman, Koucký, van Melkebeek and Ronneburger [1] showed that $\mathsf{PSPACE} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}^{\mathsf{QBF}}}$. Since their proof relies on the fact that QBF is PSPACE-complete, the proof cannot be generalized to a reduction to MCSP; hence, their reduction cannot be regarded as an oracle-independent reduction to MCSP.

4 Limits of Oracle-independent Turing Reductions to MCSP

We show upper bounds for classes of languages that reduce to MCSP in an oracle-independent manner (*i.e.,* in a way that one does not use a property of MCSP rather than that of a relativized version MCSP^A). For example, we consider a situation where a language L is reducible to MCSP^A for any A via a polynomial-time Turing reduction; more precisely, for every A, there exists a polynomial-time Turing reduction from L to MCSP^A, *i.e.,* $L \in \bigcap_A \mathsf{P}^{\mathsf{MCSP}^A}$. That is, only properties common to MCSP^A for any oracle A are used to show that L is in $\mathsf{P}^{\mathsf{MCSP}^A}$. We would like to show that L is relatively easy in such situations.

In fact, we can indeed show that any language L in $\bigcap_A \mathsf{P}^{\mathsf{MCSP}^A}$ is in P.

Theorem 1 (restated)**.** *Let $L \subseteq \{0,1\}^*$ be a language such that for any oracle A, there exists a polynomial-time Turing reduction from L to MCSP^A. Then L is in P. In short, $\bigcap_A \mathsf{P}^{\mathsf{MCSP}^A} = \mathsf{P}$.*

We will prove this theorem as follows: We will argue that, for each polynomial-time reduction M, we can adversarially choose an oracle A_M so that the reduction M cannot query any truth-table of high circuit complexity (by encoding the truth-tables queried by M into the oracle A_M). However, the assumption of the theorem states that a reduction M can depend on an oracle A, and hence A cannot depend on M. We first get around this difficulty by swapping the order of quantifiers: we reduce our theorem to the following lemma, in which a machine M cannot depend on A.

Lemma 10. *Let $L \subseteq \{0,1\}^*$ be a language and A_0 be an arbitrary finite oracle. Suppose that there exists a polynomial-time oracle Turing machine M such that $M^{\mathsf{MCSP}^A}(x) = L(x)$ for any $x \in \{0,1\}^*$ and any oracle A consistent with A_0. Then, $L \in \mathsf{P}$.*

Note that, in this lemma, a *single* machine M is required to compute L with respect to *every* oracle version MCSP^A. We will later prove this lemma by choosing, for each reduction M and input x, an oracle $A_{M,x}$ so that the reduction M to $\mathsf{MCSP}^{A_{M,x}}$ can be simulated in polynomial time. Before its proof, we show that Lemma 10 implies Theorem 1 by using a simple diagonalization argument.

Proof of Theorem 1 based on Lemma 10. We prove the contraposition: Assuming $L \notin \mathsf{P}$, the aim is to construct an oracle A such that $L \notin \mathsf{P}^A$. Such an oracle $A = \bigcup_e B_e$ is constructed in stages. Let all the polynomial-time oracle Turing machines be $\{ M_1, M_2, \cdots \}$.

At stage e, we construct a finite oracle B_e. At stage 0, set $B_0(y) := \perp$ for all $y \in \{0,1\}^*$. At stage $e \geq 1$, we apply Lemma 10 for $M = M_e$ and $A_0 = B_{e-1}$: by the assumption that $L \notin \mathsf{P}$, there exist some string x_e and some oracle B_e consistent with B_{e-1} such that $M_e^{\mathsf{MCSP}^{B_e}}(x_e) \neq L(x_e)$. We may assume that B_e is a finite oracle: indeed, since the computation of $M_e^{\mathsf{MCSP}^{B_e}}$ on input x_e makes a finite number of queries to MCSP^{B_e}, the answers of the queries also depend on a finite portion of B_e. Define an oracle A as the union of all the oracles B_e whose $\perp$ is replaced by 0.

Since A is consistent with B_e, it holds that $M_e^{\mathsf{MCSP}^{B_e}}(x_e) = M_e^{\mathsf{MCSP}^A}(x_e)$ for each $e \geq 1$. By the definition of x_e, we have $M_e^{\mathsf{MCSP}^{B_e}}(x_e) \neq L(x_e)$. Therefore, $M_e^{\mathsf{MCSP}^A}(x_e) \neq L(x_e)$ holds for any e, and hence $L \notin \mathsf{P}^{\mathsf{MCSP}^A}$. $\square$

Now we give a proof of Lemma 10. The idea is as follows: For any reduction M and any input x, we simulate the reduction M by answering M's query by exhaustively searching all the circuits of size at most $O(\log n)$. On this specific computation path of M, we claim that there exists some oracle $A_{M,x}$ such that the simulated computation path coincides with the computation path of the reduction M to $\mathsf{MCSP}^{A_{M,x}}$, thereby showing that the output of the simulation of M is $L(x)$: Since M is a polynomial-time machine, the number of the queries on the computation path is at most $n^{O(1)}$. Thus, the index i of the queries can be described in $O(\log n)$ bits, and hence the description length of the oracle circuit $C^{A_{M,x}}(j) := A_{M,x}(j,i)$ is at most $O(\log n)$. By defining $A_{M,x}(j,i) := T_{ij}$ for each truth-table T_i queried by M, any truth-table T_i admits a circuit of size at most $O(\log n)$.

Let us turn to a formal proof. Let M be a polynomial-time oracle machine that computes L given oracle access to MCSP^A in time n^c for some constant c, where A denotes an arbitrary oracle consistent with A_0. We define a polynomial-time machine M_0 that simulates M without using MCSP^A as follows: On input $x \in \{0,1\}^*$ of length n, simulate M on input x, and accept if and only if M accepts. If M makes a query (T, s), then we try to compute the circuit complexity $\mathrm{K}_{I^{A_0}}(T)$ of the truth-table T relative to a finite oracle A_0, by an exhaustive search up to size at most $4c \log n$. (More specifically, we compute the shortest description d of length at most $4c \log n$ such that $I^{A_0}(d) = T$, where we regard $A_0 \subseteq \{0,1\}^*$ as an oracle by replacing $\perp$ by 0 in finite oracle A_0.) If the circuit complexity $\mathrm{K}_{I^{A_0}}(T)$ has turned out to be greater than $4c \log n$, then define $s' := 4c \log n$; otherwise define $s' := \mathrm{K}_{I^{A_0}}(T) \ \ (\leq 4c \log n)$. (*i.e.*, $s' := \min\{4c \log n, \mathrm{K}_{I^{A_0}}(T)\}$.) Answer "Yes" to the query if and only if $s' \leq s$.

It is easy to see that M_0 is indeed a polynomial-time machine, since there are only $2^{O(\log n)}$ circuits of size at most $O(\log n)$. (Recall that we regard a circuit size as a description length.) Thus, it is sufficient to prove the following:

Claim 11. *For all sufficiently large n and all inputs x of length n, there exists an oracle $A_{M,x}$ consistent with A_0 such that $M_0(x) = M^{\mathsf{MCSP}^{A_{M,x}}}(x)$.*

Note that the assumption of Lemma 10 implies that $M^{\mathsf{MCSP}^{A_{M,x}}}(x) = L(x)$. Thus, the claim implies that $M_0(x) = L(x)$ and hence $L \in \mathsf{P}$.

Proof of Claim 11. Fix n sufficiently large and an input $x \in \{0,1\}^n$. For $i \in [n^c]$, let T_i be the truth-table in the ith query that M makes on the computation path simulated by M_0 on input x.

We define an oracle $A_{M,x} = A$ as follows (here, $A_{M,x}$ is abbreviated as A for notational convenience): For any string $q \in \{0,1\}^*$ of length less than $4c \log n$, define $A(q) = 1$ if and only if $A_0(q) = 1$. For strings of length $4c \log n$, we encode T_i into oracle A so that the circuit complexity of T_i relative to A is at most $4c \log n$: Specifically, we would like to define a description d_i of length (*exactly* equal to) $4c \log n$ so that $I^A(d_i) = T_i$. To this end, let $a_i := \log |T_i|$ and define $d_i := \langle 1^{a_i}, \underline{i}_{k_i} \rangle$, where $k_i \in \mathbb{N}$ is defined so that $|d_i| = 2a_i + 1 + k_i = 4c \log n$. Here, $\underline{i}_{k_i}$ is well-defined: indeed, we have $a_i = \log |T_i| \leq c \log n$, which implies that $k_i := 4c \log n - 2a_i - 1 \geq c \log n$, and thus $i \leq 2^{c \log n} \leq 2^{k_i}$. Now define $A(\underline{j}_{a_i}, \underline{i}_{k_i}) := T_{ij}$ for each $j \in [2^{a_i}]$. By the definition of I^A, the truth-table T_i can be described succinctly: $I^A(d_i) = A(\underline{1}_{a_i}, \underline{i}_{k_i}) \cdots A(\underline{2^{a_i}}_{a_i}, \underline{i}_{k_i}) = T_i$; thus, the circuit complexity $\mathrm{K}_{I^A}(T_i)$ of T_i is at most $|d_i| = 4c \log n$.

It remains to show that, for each query (T_i, s) that M makes on the computation path simulated by M_0, circuit complexity $s' \ (= \min\{4c \log n, \mathrm{K}_{I^{A_0}}(T_i)\})$ calculated by M_0 coincides with $\mathrm{K}_{I^A}(T_i)$; note that this implies that $M_0(x) = M^{\mathsf{MCSP}^A}(x)$, because the computation path simulated by M_0 coincides with that of M relative to MCSP^A. In order to see $\mathrm{K}_{I^A}(T_i) = \min\{4c \log n, \mathrm{K}_{I^{A_0}}(T_i)\}$, first we note that A and A_0 are consistent up to length $4c \log n - 1$; thus, for small circuits, circuit complexity relative to A remains the same with circuit complexity relative to A_0, because small circuits cannot query long strings of length $4c \log n$. Formally, suppose that $\mathrm{K}_{I^{A_0}}(T_i) < 4c \log n$ (*i.e.*, $s' = \mathrm{K}_{I^{A_0}}(T_i)$). In this case, there exists some description d of length less than $4c \log n$ such that $I^{A_0}(d) = T_i$. Since the circuit described by d cannot make any query of length greater than $|d|$, it holds that $I^{A_0}(d) = I^A(d)$. Thus $\mathrm{K}_{I^A}(T_i) \leq \mathrm{K}_{I^{A_0}}(T_i) < 4c \log n$. Similarly, we have $\mathrm{K}_{I^{A_0}}(T_i) \leq \mathrm{K}_{I^A}(T_i)$, and hence $\mathrm{K}_{I^A}(T_i) = \mathrm{K}_{I^{A_0}}(T_i) = s'$. Now suppose that $\mathrm{K}_{I^{A_0}}(T_i) \geq 4c \log n$ (*i.e.*, $s' = 4c \log n$). We claim that $\mathrm{K}_{I^A}(T_i) = 4c \log n$. Since we have $\mathrm{K}_{I^A}(T_i) \leq 4c \log n$ by the definition of A, it is sufficient to show that $\mathrm{K}_{I^A}(T_i) < 4c \log n$ is not true. Assume, by way of contradiction, that $\mathrm{K}_{I^A}(T_i) < 4c \log n$. By the same argument above, it must be the case that $\mathrm{K}_{I^A}(T_i) \geq \mathrm{K}_{I^{A_0}}(T_i) \geq 4c \log n$, which is a contradiction. $\square$

This completes the proof of Lemma 10.

Remark. If we regard a size of a circuit as the number of its wires, then the upper bound P becomes $\mathsf{DTIME}(n^{O(\log \log n)})$. Specifically, let MCSP'^A denotes a version of MCSP^A in which a size of a circuit is measured by the number of its wires. Then we have $\bigcap_A \mathsf{P}^{\mathsf{MCSP}'^A} \subseteq \mathsf{DTIME}(n^{O(\log\log n)})$. This can be proved by simply changing M_0 in the proof above so that M_0 exhaustively search all the circuits of at most $O(\log n)$ *wires* in time $O(\log n)^{O(\log n)} = n^{O(\log\log n)}$.

5 Limits of Oracle-independent Randomized Reductions to MCSP

In this section, we discuss the limits of a randomized reduction to MCSP that can be generalized to a reduction to MCSP^A for an arbitrary oracle A. Our focus is a randomized reduction with negligible two-sided error that can make at most one query:

Definition. *Let $L, B \subseteq \{0,1\}^*$ be a language and an oracle, respectively. We say that L reduces to B via a* one-query BPP-reduction *and write $L \in \mathsf{BPP}^{B[1]}$ if there exist polynomial-time machines M, Q and a negligible function ϵ such that, for any $x \in \{0,1\}^*$,*

$$\Pr_{r \in \{0,1\}^{|x|^{O(1)}}}[M(x, r, B(Q(x,r))) = L(x)] \geq 1 - \epsilon(|x|).$$

Here, we say that a function ϵ is negligible if for all polynomials p, for all sufficiently large $n \in \mathbb{N}$, the function is bounded by the inverse of p: that is, $\epsilon(n) < \frac{1}{p(n)}$.

Note that we require the error probability to be negligible. Since the number of queries is restricted to one, we cannot apply the standard error-reduction argument; hence, this definition may be stronger than a definition whose error probability is a constant. We leave as an open problem improving our result to the case when the error probability is a constant.

We prove that there is no language outside $\mathsf{AM} \cap \mathsf{coAM}$ that can reduce to MCSP^A for an arbitrary oracle A via a one-query randomized reduction:

Theorem 2 (restated)**.** *Let $L \subseteq \{0,1\}^*$ be a language such that for any oracle A, there exists a one-query* BPP*-reduction from L to MCSP^A. Then L is in $\mathsf{AM} \cap \mathsf{coAM}$. In short,*

$$\bigcap_A \mathsf{BPP}^{\mathsf{MCSP}^A[1]} \subseteq \mathsf{AM} \cap \mathsf{coAM}.$$

As with Theorem 1, we first swap the order of quantifiers. However, in order to swap the order of quantifiers, we need to enumerate all the negligible functions, which is not countably many; thus, we sidestep this by requiring that the error probability is an inverse polynomial $1/q$ in the running time of machines M and Q. Also, since a one-query BPP-reduction is closed under complement, we only have to show that the target language is in AM.

Lemma 12. *There exists some universal polynomial q (specified later) that satisfies the following: Let L, A_0 be a language and a finite oracle, respectively. Suppose that there exist a polynomial p and Turing machines M, Q such that M and Q run in time $p(n)$ and*

$$\Pr_{r \in \{0,1\}^{p(n)}}[M(x, r, \mathsf{MCSP}^A(Q(x,r))) = L(x)] \geq 1 - \frac{1}{q(p(n))}$$

for any $x \in \{0,1\}^*$ of length n and any oracle $A \subseteq \{0,1\}^*$ consistent with A_0. Then, we have $L \in \mathsf{AM}$.

We prove that Lemma 12 implies Theorem 2:

Proof of Theorem 2 based on Lemma 12. We prove the contraposition: Assuming $L \notin \mathsf{AM}$, we will construct an oracle A such that $L \notin \mathsf{BPP}^{\mathsf{MCSP}^A[1]}$ by diagonalization.

Enumerate all the tuples $\{(M_e, Q_e, c_e)\}_{e \geq 1}$, where M_e and Q_e are polynomial-time machines and $c_e \in \mathbb{N}$. We assume that, for each tuple (M_e, Q_e, c_e), there exist infinitely many $e' \in \mathbb{N}$ such that $(M_e, Q_e, c_e) = (M_{e'}, Q_{e'}, c_{e'})$.

At stage $e \geq 1$, we construct a finite oracle B_e that fools a one-query BPP reduction (M_e, Q_e) that runs in time n^{c_e}: If M_e or Q_e does not run in time n^{c_e}, then we define $B_e := B_{e-1}$. Otherwise, we can apply the contraposition of Lemma 12 to M_e and Q_e: there exist some input x_e and some oracle B_e consistent with B_{e-1} such that $\Pr_r[M_e(x_e, r, \mathsf{MCSP}^{B_e}(Q_e(x_e, r))) = L(x_e)] < 1 - \frac{1}{q(n^{c_e})}$. We can make B_e a finite oracle, since M_e depends on only a finite portion of B_e. This completes stage e. Define A as the union of all the oracles B_e whose $\perp$ is replaced by 0.

We claim that $L \notin \mathsf{BPP}^{\mathsf{MCSP}^A[1]}$. Assume otherwise. Then there exist a constant $c > 1$, a negligible function ϵ, and machines M and Q that run in time n^c such that

$$\Pr_r[M(x, r, \mathsf{MCSP}^A(Q(x, r))) = L(x)] \geq 1 - \epsilon(|x|) \tag{1}$$

for all $x \in \{0,1\}^*$. Fix a sufficiently large $n_0 \in \mathbb{N}$ such that $\epsilon(n) < \frac{1}{q(n^{c+1})}$ for all $n \geq n_0$. Let M' be the Turing machine[3] that, on input x, outputs a hardwired answer $L(x)$ if $|x| \leq n_0$, and simulates M otherwise. Note that the running time of M' is at most n^{c+1}.

By the construction above, there exists $e \geq n_0$ such that $(M_e, Q_e, c_e) = (M', Q, c+1)$. By the definition of x_e, we have $\Pr_r[M'(x_e, r, \mathsf{MCSP}^A(Q(x_e, r))) = L(x_e)] < 1 - \frac{1}{q(|x_e|^{c+1})}$. Moreover, since M' outputs a correct answer with probability 1 on input x of length at most n_0, it holds that $|x_e| > n_0$; thus, we have $\epsilon(|x_e|) < \frac{1}{q(|x_e|^{c+1})}$; in addition, the machine M' behaves in the same way with M. Hence, the success probability of (M, Q) on input x_e is equal to that of (M', Q) on input x_e, which is bounded above by $1 - \frac{1}{q(|x_e|^{c+1})} < 1 - \epsilon(|x_e|)$. This contradicts (1). ☐

Now we outline the proof of Lemma 12.

We will first show that we may assume that all the queries that Q makes have a truth-table of a fixed length 2^t and a fixed size-parameter s for some $t, s \in \mathbb{N}$. There is no loss of generality in assuming this because there are only polynomially many possibilities: the number of all the possible lengths of a truth-table and size-parameters is at most n^c for some c. Moreover, we may fix how to use the answer of a query: specifically, for a random choice r, define $f : \{0,1\} \to \{0,1\}$ (which has 4 possible choices) so that $f(b) = M(x, r, b)$. (For example, $f(b) = b$ means that M accepts if and only if the query is a positive instance of MCSP^A.)

We classify the set of random choices r into $R_{f,t,s}$ according to these parameters (f, t, s). If $x \in L$, then there must exist some (f, t, s) such that $f(\mathsf{MCSP}^A(Q(x, r))) = 1$ with high probability over the choice of $r \in_R R_{f,t,s}$. On the other hand, if $x \notin L$, then any (f, t, s) must satisfy $f(\mathsf{MCSP}^A(Q(x, r))) = 0$ with high probability. Therefore, it is sufficient to prove that, for a specific (f, t, s), there exists an AM protocol that checks if $f(\mathsf{MCSP}^A(Q(x, r))) = 1$ with high probability conditioning on $r \in R_{f,t,s}$.

Let us assume that $f(b) = b$ for simplicity. Then, it is sufficient to estimate the probability

$$P_{f,t,s} := \Pr_{r \in_R R_{f,t,s}}[f(\mathsf{MCSP}^A(Q(x, r))) = 1] = \Pr_{r \in_R R_{f,t,s}}[Q(x, r) \in \mathsf{MCSP}^A]$$

[3] M' can be implemented by a Turing machine as follows: Read the first $n_0 + 1$ bits of the input (if any). If the input length is at most n_0, then output the hardwired answer. Otherwise, move the head of the input tape to the initial position, and continue the computation of M. This implementation costs at most $2n_0$ additional steps.

by an AM protocol. If the probability $P_{f,t,s}$ is close to 1, then the distribution induced by $Q(x,r)$ concentrates on a limited number of instances: indeed, since there are at most 2^{s+1} positive instances in MCSP^A for a size-parameter s, the query $Q(x,r)$ must be one of such instances with probability at least $P_{f,t,s}$. Conversely, suppose that the query distribution $Q(x,r)$ concentrates on a limited number of instances $\{(T_1,s),(T_2,s),\cdots\}$; we may encode T_i into an oracle A and force these instances to be positive (*i.e.,* $(T_i,s)\in\mathsf{MCSP}^A$); as a result, the probability $P_{f,t,s}$ is not small (since the instances (T_i,s) are positive). Therefore, the task reduces to checking whether the query distribution concentrates on a limited number of instances.

To this end, we will use the heavy samples protocol [6]. We say that an instance (T,s) is β-heavy if the probability that (T,s) is queried (*i.e.,* $(T,s)=Q(x,r)$) is at least β. The heavy samples protocol allows us to estimate the probability that $Q(x,r)$ is β-heavy.

Lemma 13 (The heavy samples protocol; Trevisan and Bogdanov [6])**.** *Let $\mathcal{D}=\{\mathcal{D}_n\}_{n\in\mathbb{N}}$ be a polynomial-time samplable distribution. There exist a universal constant c ($c=2^{11}$ will do) and an $\mathsf{AM}\cap\mathsf{coAM}$ protocol that solves the following promise problem: Given input 1^n and a threshold $\beta\in[0,1]$, accept if $\Pr_{y\sim\mathcal{D}_n}[\,y$ is $c\beta$-heavy$\,]\geq\frac{3}{4}$, and reject if $\Pr_{y\sim\mathcal{D}_n}[\,y$ is β-heavy$\,]\leq\frac{1}{4}$.*

Since the problem setting is slightly different from that of [6], we include a proof of this lemma based on the lower bound protocol (Goldwasser and Sipser [10]) and the upper bound protocol (Fortnow [9]) in Appendix A.

Now we give a formal proof of Lemma 12. For the proof, we need to show an AM protocol deciding whether $x\in L$; we will show the protocol by a sequence of claims.

We begin with clarifying our setting and introducing some notation. Let $p(n)$ be a polynomial that is an upper bound of the running time of M and Q. Fix a sufficiently large $n\in\mathbb{N}$ and an input $x\in\{0,1\}^n$.

Let $f\colon\{0,1\}\to\{0,1\}$ be a function, and $t,s\in\mathbb{N}$. We define $R_{f,t,s}\subseteq\{0,1\}^{p(n)}$ as the set of all the random choices $r\in\{0,1\}^{p(n)}$ such that $M(x,r,b)=f(b)$ for all $b\in\{0,1\}$ and $(T,s)=Q(x,r)$ and $|T|=2^t$. That is, f specifies how to use the answer from oracle MCSP^A, and t and s specify the length of the truth-table and the size-parameter in the query, respectively. Let $X:=\{(f,t,s)\mid R_{f,t,s}\neq\varnothing\}$. We may assume, without loss of generality, that $s\leq p(n)^2$ as otherwise $Q(x,r)$ is obviously a positive instance; hence, $|X|\leq 2^2\cdot\log p(n)\cdot p(n)^2\leq 4p(n)^3$.

Define $P_{f,t,s}:=\Pr_r[f(\mathsf{MCSP}^A(Q(x,r)))=1\mid r\in R_{f,t,s}]$ for $(f,t,s)\in X$. Let us divide the probability that M accepts x by conditioning on $r\in R_{f,t,s}$:

$$\Pr_r[M(x,r,\mathsf{MCSP}^A(Q(x,r))=1]=\sum_{(f,t,s)\in X}\Pr_r[r\in R_{f,t,s}]\cdot P_{f,t,s}. \tag{2}$$

Since there are polynomially many choices for (f,t,s), there must be some $(f,t,s)\in X$ that can be used as a "witness" for $x\in L$ in our AM protocol. Specifically, the following claim holds:

Claim 14. *Let $\delta(n):=\sqrt{1/q(p(n))}$ and $\delta'(n):=9p(n)^3\delta(n)$.*

1. *If $x\in L$, then there exists $(f,t,s)\in X$ such that $\Pr[r\in R_{f,t,s}]\geq 2\delta(n)$ and $P_{f,t,s}\geq 1-\delta'(n)$.*
2. *If $x\notin L$, then $P_{f,t,s}\leq\delta(n)$ for all $(f,t,s)\in X$ such that $\Pr[r\in R_{f,t,s}]\geq\delta(n)$.*

Proof of Claim 14.

1. Suppose that $x \in L$; then, the probability (2) is at least $1 - \delta(n)^2$. Assume, by way of contradiction, that $P_{f,t,s} < 1 - \delta'(n)$ for all $(f, t, s) \in X$ such that $\Pr[r \in R_{f,t,s}] \geq 2\delta(n)$. Then,

$$\begin{aligned}1 - \delta(n)^2 &\leq \sum_{(f,t,s)\in X} \Pr_r[r \in R_{f,t,s}] \cdot P_{f,t,s} \\ &= \sum_{\substack{(f,t,s)\in X \\ \Pr[r\in R_{f,t,s}]\geq 2\delta(n)}} \Pr_r[r \in R_{f,t,s}] \cdot P_{f,t,s} \quad + \sum_{\substack{(f,t,s)\in X \\ \Pr[r\in R_{f,t,s}]< 2\delta(n)}} \Pr_r[r \in R_{f,t,s}] \cdot P_{f,t,s} \\ &\leq 1 - \delta'(n) + 2|X|\delta(n) \leq 1 - 9p(n)^3\delta(n) + 8p(n)^3\delta(n) = 1 - p(n)^3\delta(n).\end{aligned}$$

   Thus $p(n)^3 \leq \delta(n) < 1$, which is a contradiction.

2. Let X' be the set of all $(f, t, s) \in X$ such that $\Pr[r \in R_{f,t,s}] \geq \delta(n)$. Suppose that $x \notin L$; then,

$$\delta(n)^2 \geq \sum_{(f,t,s)\in X} \Pr_r[r \in R_{f,t,s}] \cdot P_{f,t,s} \geq \delta(n) \cdot \sum_{(f,t,s)\in X'} P_{f,t,s},$$

   which clearly implies that $P_{f,t,s}$ is at most $\delta(n)$ for each $(f, t, s) \in X'$.

□

In our AM protocol, the prover first sends (f, t, s) to the verifier; an honest prover is supposed to send $(f, t, s) \in X$ that satisfies the first condition in Claim 14 above. Then, what we need is to show a verifier of an AM protocol as stated in the following claim.

Claim 15. *There exists a verifier V of an* AM *protocol such that, for a given $x \in \{0, 1\}^*$ and $(f, t, s) \in X$,*

1. *if $\Pr[r \in R_{f,t,s}] \geq 2\delta(n)$ and $P_{f,t,s} \geq 1 - \delta'(n)$, then V accepts with high probability by communicating with some prover, and*

2. *if $\Pr[r \in R_{f,t,s}] < \delta(n)$ or $P_{f,t,s} \leq \delta(n)$, then V rejects with high probability with any prover.*

We explain below how to define this verifier V. Recall that $1/\delta(n) = n^{O(1)}$; thus, it is easy to distinguish the case when $\Pr[r \in R_{f,t,s}] \geq 2\delta(n)$ and the case when $\Pr[r \in R_{f,t,s}] < \delta(n)$. The following claim states this formally.

Claim 16. *There exists a randomized polynomial-time algorithm that, given $x \in \{0, 1\}^*$ and $(f, t, s) \in X$,*

1. *accepts with high probability if $\Pr[r \in R_{f,t,s}] \geq 2\delta(n)$, and*

2. *rejects with high probability if $\Pr[r \in R_{f,t,s}] < \delta(n)$.*

Proof Sketch. Sample $r_1, \cdots, r_m \in_R \{0, 1\}^{p(n)}$ uniformly at random for $m = O(n/\delta(n)^2)$. Accept if and only if the number of i's such that $r_i \in R_{f,t,s}$ is at least $1.5 \cdot \delta(n)m$. By applying the Chernoff bound, this algorithm distinguishes the two cases with probability at least $1 - 2^{-n}$. □

Therefore in our AM protocol, verifier V first uses this algorithm to check whether $\Pr[r \in R_{f,t,s}] \geq 2\delta(n)$ or $\Pr[r \in R_{f,t,s}] < \delta(n)$. If $\Pr[r \in R_{f,t,s}] < \delta(n)$ is confirmed by the algorithm, then

V can immediately reject (f, t, s) and halt. Thus, it remains to design a part where V determines whether $P_{f,t,s} \geq 1 - \delta'(n)$ or $P_{f,t,s} \leq \delta(n)$ $(\leq \delta'(n))$ holds, assuming that $\Pr[r \in R_{f,t,s}] \geq \delta(n)$. Note that this assumption implies that the uniform distribution on $R_{f,t,s}$ can be sampled efficiently: indeed, sample $r \in_R \{0,1\}^{p(n)}$ until we obtain an element r such that $r \in R_{f,t,s}$; this sampling algorithm succeeds within $O(1/\delta(n))$ steps in expectation.

Now our task is to define an AM protocol determining whether $P_{f,t,s}$ is close to 1 or smaller than $\delta'(n)$, assuming that the query distribution $Q(x, r)$ where $r \in_R R_{f,t,s}$ can be sampled efficiently. Note that $P_{f,t,s}$ may depend on x and MCSP^A. We will show that the task above can be reduced to checking whether a certain concentration occurs, by defining A so that heavy queries become positive instances. Then we will check if such a concentration occurs by the heavy samples protocol of Lemma 13.

In order to reduce Claim 15 to the heavy samples protocol, we introduce some notation: Fix $(f, t, s) \in X$. Let us sort all the truth-tables $\{ T_1, \cdots, T_{2^{2^t}} \} = \{0,1\}^{2^t}$ of length 2^t in the order of heaviness: namely, let $p_i := \Pr_{r \in_R R_{f,t,s}}[Q(x,r) = (T_i, s)]$ and $p_1 \geq p_2 \geq \cdots \geq p_{2^{2^t}}$. Let $p(I)$ denote $\sum_{i \in I} p_i$ for $I \subseteq [2^{2^t}]$. Define the set of α-heavy indices (with respect to the distribution induced by $Q(x, r)$ where $r \in_R R_{f,t,s}$) as $I_\alpha := \{ i \in [2^{2^t}] \mid p_i \geq \alpha \}$ for $\alpha \geq 0$. Note that $p(I_\alpha) = \Pr_{r \in_R R_{f,t,s}}[\, Q(x,r) \text{ is } \alpha\text{-heavy} \,]$. We also define $P_{\mathrm{id},t,s} := \Pr_{r \in_R R_{f,t,s}}[\, Q(x,r) \in \mathsf{MCSP}^A \,]$.

We will show that the condition that $P_{\mathrm{id},t,s}$ is close to 1 is (almost) characterized by the fact that the query distribution is concentrated on $\{ (T_i, s) \mid i \in I_\beta \}$, namely the set of β-heavy instances for some threshold $\beta > 0$.

Claim 17. *There exists an oracle A consistent with A_0 up to length $7 \log p(n)$ that satisfies the following: for any $(f, t, s) \in X$ such that $\Pr[r \in R_{f,t,s}] \geq \delta(n)$ and $s > 7 \log p(n)$ hold,*

1. *if $P_{\mathrm{id},t,s} \geq 1 - \delta'(n)$, then $p(I_{c\beta}) \geq 1 - 3c\delta''(n)$, and*
2. *if $P_{\mathrm{id},t,s} \leq \delta'(n)$, then $p(I_\beta) \leq \delta''(n)$.*

Here, we define $\delta''(n) := 2\,p(n)^7\,\delta'(n) = 18\,p(n)^{10}\,\delta(n)$ and $\beta := \delta''(n)\,2^{-s}$, and c denotes the universal constant in Lemma 13.

This claim allows us to apply the heavy samples protocol. Let us complete the proof of Claim 15 before proving Claim 17.

Proof of Claim 15. As explained above, it is sufficient to show that our verifier V can check whether $P_{f,t,s} \geq 1 - \delta'(n)$ or $P_{f,t,s} \leq \delta'(n)$, assuming that $\Pr[r \in R_{f,t,s}] \geq \delta(n)$.

If $f \equiv 1$ or $f \equiv 0$, then the task is trivial: in the former case, $P_{f,t,s} = 1$ and hence V may immediately accept; in the latter case, V rejects.

If $s \leq 7 \log p(n)$, then we may decide whether $Q(x, r) \in \mathsf{MCSP}^{A_0}$ or not by an exhaustive search in time $2^{O(s)} = n^{O(1)}$. Since A and A_0 are consistent up to length $7 \log p(n)$, as in the proof of Lemma 10, it holds that $Q(x, r) \in \mathsf{MCSP}^{A_0}$ if and only if $Q(x, r) \in \mathsf{MCSP}^A$. Therefore, we may estimate $P_{f,t,s}$ by sampling $r \in_R R_{f,t,s}$ and then decide whether $Q(x, r) \in \mathsf{MCSP}^A$ by the exhaustive search.

Otherwise, we have $s > 7 \log p(n)$ and hence Claim 17 can be applied. Now suppose that $f(b) = b$ for any $b \in \{0, 1\}$. In this case, it holds that $P_{f,t,s} = P_{\mathrm{id},t,s}$; thus Claim 17 states that, if $P_{f,t,s} \geq 1 - \delta'(n)$ then $p(I_{c\beta}) \geq 1 - 3c\delta''(n)$, and if $P_{f,t,s} \leq \delta'(n)$ then $p(I_\beta) \leq \delta''(n)$. Now we may apply the heavy samples protocol for the query distribution induced by $Q(x, r)$ where $r \in_R R_{f,t,s}$, in order to check whether $p(I_{c\beta}) \geq 1 - 3c\delta''(n)$ or $p(I_\beta) \leq \delta''(n)$: more specifically, V accepts in the former case by running the AM protocol of Lemma 13.

Similarly, if $f(b) = 1-b$, then we have $P_{f,t,s} = 1 - P_{\mathrm{id},t,s}$. This implies the same condition except for flipping YES and NO: if $P_{f,t,s} \geq 1 - \delta'(n)$ then $p(I_\beta) \leq \delta''(n)$; if $P_{f,t,s} \leq \delta'(n)$ then $p(I_{c\beta}) \geq 1 - 3c\delta''(n)$. Thus, we may apply the heavy samples protocol to check whether $p(I_\beta) \leq \delta''(n)$ or $p(I_{c\beta}) \geq 1 - 3c\delta''(n)$: specifically, V accepts in the former case by running the coAM protocol of Lemma 13.

Note that we may pick the polynomial q that specifies the error probability so that $3c\delta''(n) \leq \frac{1}{4}$ (which allows us to use Lemma 13): indeed, if we define $q(n) := O(n^{22})$ then we have $\delta(n) = \sqrt{1/q(p(n))} = O(p(n)^{-11})$ and hence $3c\delta''(n) = O(p(n)^{10}\,\delta(n)) = o(1)$. $\square$

All that remains is to show Claim 17. The intuition is as follows: Suppose that the probability that a positive instance is queried is large (*i.e.,* $P_{\mathrm{id},t,s} \geq 1 - \delta'(n)$). Since there are at most 2^{s+1} truth-tables that have circuit complexity at most s, the query distribution must concentrate on such positive instances; thus $p([2^{s+1}])$ is also large, which in particular implies that $p(I_{c\beta})$ is large (since β is in fact chosen so that $p(I_\beta)$ is roughly equal to $p([2^{s+1}])$).

Conversely, suppose that the query distribution concentrates on heavy instances $\{\,(T_1, s), \ldots, (T_{2^k}, s)\,\}$ (*i.e.,* $p([2^k])$ is large) for some k. In this case, we may encode the heavy truth-tables into the oracle A; thereby we can force these truth-tables to be positive instances, which implies that $p([2^k]) \leq P_{\mathrm{id},t,s}$; hence $P_{\mathrm{id},t,s}$ cannot be small. The details follow:

Proof of Claim 17.

1. Suppose that $P_{\mathrm{id},t,s} \geq 1 - \delta'(n)$. Then,

$$\begin{aligned} 1 - \delta'(n) &\leq \Pr_{r \in_R R_{f,t,s}}[Q(x,r) \in \mathsf{MCSP}^A] \\ &= \sum_{i\colon (T_i,s) \in \mathsf{MCSP}^A} p_i \leq \sum_{i=1}^{2^{s+1}} p_i = p([2^{s+1}]), \end{aligned}$$

   where in the last inequality we used the fact that there are at most 2^{s+1} positive instances in MCSP^A. Now,

$$1 - \delta'(n) \leq p([2^{s+1}]) = p([2^{s+1}] \cap I_{c\beta}) + p([2^{s+1}] \setminus I_{c\beta}) \leq p(I_{c\beta}) + 2^{s+1} \cdot c\beta,$$

   which implies that $p(I_{c\beta}) \geq 1 - 2^{s+1} \cdot c\beta - \delta'(n) \geq 1 - 2c\delta''(n) - \delta'(n) \geq 1 - 3c\delta''(n)$.

2. Note that an oracle A can depend on input x, but A must not depend on a specific (f, t, s). Thus, we define A so that, for *all* $(f, t, s) \in X$, the heavy queries $\{\,(T_1, s), \cdots, (T_{2^k}, s)\,\}$ (with respect to the distribution induced by $Q(x, r)$ where $r \in_R R_{f,t,s}$) become positive instances. (Note that truth-tables T_i depend on (f, t, s).) For any string y of length at most $7 \log p(n)$, we define $A(y) := 1$ if and only if $A_0(y) = 1$, which ensures that A and A_0 are consistent up to length $7 \log p(n)$. For each $(f, t, s) \in X$ such that $s > 7 \log p(n)$, define $k := s - 7 \log p(n)$; for each $i \in [2^k]$, we would like to define A so that $I^A(d) = T_i$ for some description d of length exactly equal to s. To this end, define $d := \langle 1^t, \langle f, s, \underline{i}_{k_i} \rangle \rangle$ and $A(\underline{j}_t, \langle f, s, \underline{i}_{k_i} \rangle) = T_{ij}$ for all $j \in [2^t]$, where k_i is chosen so that $|d| = 2t + 2\log s + k_i + O(1) = s$. Thus, $k_i := s - 2t - 2\log s - O(1) \geq s - 7 \log p(n) = k$. This ensures that $\underline{i}_{k_i}$ is well-defined. These imply that $\mathrm{K}_{I^A}(T_i) \leq s$; hence, $(T_i, s) \in \mathsf{MCSP}^A$ for all $i \in [2^k]$ and therefore $p([2^k]) \leq P_{\mathrm{id},t,s}$.

Now fix $(f, t, s) \in X$ such that $k = s - 7\log p(n) > 0$ and $P_{\mathrm{id},t,s} \le \delta'(n)$ hold. Since k and s are close, it holds that

$$\begin{aligned} p([2^{s+1}]) &= p(\{1, \cdots, 2^k\}) + p(\{2^k + 1, \cdots, 2 \cdot 2^k\}) + \cdots + p(\{2^{s+1} - 2^k + 1 \cdots, 2^{s+1}\}) \\ &\le 2^{s+1}/2^k \cdot p([2^k]) \le 2^{s+1-k} \cdot P_{\mathrm{id},t,s} = 2p(n)^7 \cdot P_{\mathrm{id},t,s} \le 2p(n)^7 \cdot \delta'(n) = \delta''(n). \end{aligned}$$

We claim that this implies $p(I_\beta) \le \delta''(n)$: Indeed, let $j := \max I_\beta$. If $j > 2^s$, then $\delta''(n) \ge p([2^{s+1}]) \ge p([2^{s+1}] \cap I_\beta) \ge \beta \cdot \min\{2^{s+1}, j\} > \beta \cdot 2^s = \delta''(n)$, which is a contradiction. Thus, we have $j \le 2^s$, and hence $p(I_\beta) \le p([2^{s+1}]) \le \delta''(n)$ as desired.

$\square$

This completes the proof of Lemma 12.

6 Hardness of MCSP Implies Separations of Complexity Classes

In this section, we give a reinterpretation of the results of Murray and Williams [17] by using Levin's Kolmogorov complexity, and extend these results to the case of polynomial-time nonadaptive reductions and polynomial-time Turing reductions. Our proofs are based on the firm links between circuit complexity and resource-bounded Kolmogorov complexity, which have been established by a line of work [1, 5]. First, we introduce Levin's Kt-complexity.

Definition (Levin's Kolmogorov Complexity [16])**.** *Fix an efficient universal Turing machine U. The* Levin's Kolmogorov complexity $\mathrm{Kt}(x)$ *of a string x is defined as*

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

Our proof is principally based on the fact that $\mathsf{EXP} \subseteq \mathsf{P/poly}$ if and only if circuit complexity K_I is polynomially related to Levin's Kolmogorov complexity Kt.

Lemma 18 (Allender, Koucký, Ronneburger and Roy [5])**.** $\mathsf{EXP} \subseteq \mathsf{P/poly}$ *if and only if there exists a polynomial* poly *in two variables such that* $\mathrm{K}_I(x) \le \mathsf{poly}(\mathrm{Kt}(x), \log|x|)$*.*

We would like to separate the class of languages reducible to MCSP from EXP, under the assumption that $\mathsf{EXP} \subseteq \mathsf{P/poly}$. Under this assumption, Lemma 18 suggests that circuit complexity and Kt-complexity are essentially the same (in the sense that these are polynomially related to each other). Therefore, we will first separate the class of languages reducible to Kt from EXP, and then, based on Lemma 18, translate the property of Kt into that of MCSP, assuming $\mathsf{EXP} \subseteq \mathsf{P/poly}$.

6.1 The Case of Nonadaptive Reductions

In the case of polynomial-time nonadaptive reductions, it is well known that $\mathsf{P}_{||}^{\mathrm{Kt}} \ne \mathsf{EXP}$.

Proposition 19 (folklore)**.** $\mathsf{EXP}_{||}^{\mathrm{Kt}} = \mathsf{EXP}$*. (Here,* Kt *is identified with the oracle $\{(x, s) \in \{0,1\}^* \times \mathbb{N} \mid \mathrm{Kt}(x) \le s\}$.)*

Note that this implies $\mathsf{P}_{||}^{\mathrm{Kt}} \ne \mathsf{EXP}$ by the time hierarchy theorem.

Proof. Let M be any $\mathsf{EXP}_{||}^{\mathrm{Kt}}$ machine. Given input $x \in \{0,1\}^*$ of length n, let $Q(x)$ be the set of queries (without size-parameter s) that M makes. Since M is a nonadaptive oracle machine, $Q(x)$ can be computed in exponential time. Therefore, any query $q \in Q(x)$ can be described by the input x and an index $i \in [2^{n^{O(1)}}]$ in exponential time; hence, $\mathrm{Kt}(q) \le |x| + n^{O(1)} + \log 2^{n^{O(1)}} = n^{O(1)}$.

Given the fact that $\mathrm{Kt}(q) \le n^{O(1)}$, we may compute $\mathrm{Kt}(q)$ by an exhaustive search in exponential time. Thus, by answering M's queries by the exhaustive search, we can compute M's output in exponential time. $\square$

Under the assumption that $\mathsf{EXP} \subseteq \mathsf{P/poly}$, we can translate the property of Kt into that of circuit complexity:

Theorem 20. *If* $\mathsf{EXP} \subseteq \mathsf{P/poly}$ *then* $\mathsf{EXP}_{||}^{\mathsf{MCSP}} = \mathsf{EXP}$*.*

Proof Sketch. Let (T, s) be any query of an $\mathsf{EXP}_{||}^{\mathsf{MCSP}}$ machine. Since $\mathrm{Kt}(T)$ is $n^{O(1)}$, the circuit complexity $\mathrm{K}_I(T)$ of T is also bounded above by $n^{O(1)}$ by Lemma 18. Thus, the circuit complexity of all the queries can be computed by an exhaustive search in time exponential in n. $\square$

This theorem allows us to obtain a nontrivial separation of $\mathsf{P}_{||}^{\mathsf{MCSP}} \cap \mathsf{P/poly}$ from EXP:

Corollary 21. $\mathsf{P}_{||}^{\mathsf{MCSP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$*.*

Proof. Assume, by way of contradiction, that $\mathsf{P}_{||}^{\mathsf{MCSP}} \cap \mathsf{P/poly} = \mathsf{EXP}$. In particular, $\mathsf{EXP} \subseteq \mathsf{P/poly}$. Thus, by Theorem 20, we have $\mathsf{EXP}_{||}^{\mathsf{MCSP}} = \mathsf{EXP}$. Therefore, $\mathsf{EXP}_{||}^{\mathsf{MCSP}} = \mathsf{EXP} = \mathsf{P}_{||}^{\mathsf{MCSP}}$, which contradicts the (relativized) time hierarchy theorem [11]. $\square$

This result exhibits a singular property of MCSP. In particular, reducing a language L to MCSP via a polynomial-time nonadaptive reduction implies a separation of $\mathsf{P}_{||}^{L} \cap \mathsf{P/poly}$ from EXP.

Corollary 22. *If* $L \le_{tt}^{p} \mathsf{MCSP}$*, then* $\mathsf{P}_{||}^{L} \cap \mathsf{P/poly} \neq \mathsf{EXP}$*.*

Proof. The hypothesis implies that $\mathsf{P}_{||}^{L} \subseteq \mathsf{P}_{||}^{\mathsf{MCSP}}$, and by the previous corollary it holds that $\mathsf{EXP} \not\subseteq \mathsf{P}_{||}^{\mathsf{MCSP}} \cap \mathsf{P/poly}$, from which the result follows. $\square$

We give some specific remarks:

Remark.

1. If MCSP is ZPP-hard under polynomial-time nonadaptive reductions, then $\mathsf{ZPP} \neq \mathsf{EXP}$, which is a notorious open problem.
2. If MCSP is NP-complete under polynomial-time nonadaptive reductions, then $\mathsf{P}_{||}^{\mathsf{NP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$. (The consequence is also a tiny improvement of Murray and Williams [17], who showed that $\mathsf{NP} \cap \mathsf{P/poly} \neq \mathsf{EXP}$ under the assumption that $\mathsf{NP} \le_m^p \mathsf{MCSP}$.)

6.2 On Hardness of Approximating Kt-complexity and Circuit Complexity

Now we turn to the case of polynomial-time Turing reductions. We first introduce some definitions about promise problems:

Definition. *1. A* promise problem $\Pi = (\Pi_Y, \Pi_N)$ *is a pair of disjoint languages* Π_Y *and* Π_N*, where* Π_Y *is the set of YES instances and* Π_N *is the set of NO instances.*

2. We say that an oracle A satisfies the promise of $\Pi = (\Pi_Y, \Pi_N)$ *if, for any* $x \in \{0,1\}^*$*, it holds that* $x \in \Pi_Y$ *implies* $x \in A$*, and that* $x \in \Pi_N$ *implies* $x \notin A$*.*

3. We say that a language L is reducible to a promise problem Π *via a polynomial-time Turing reduction M and write* $L \le_T^p \Pi$ *if* $M^A(x) = L(x)$ *for any* $x \in \{0,1\}^*$ *and any oracle A that satisfies the promise of* Π*.*

We show that approximating Kt-complexity within additive error $g(n) = \omega(\log n)$ is not EXP-complete under polynomial-time Turing reductions. We denote such a promise problem by $\mathrm{Gap}_g\mathrm{Kt}$:

Definition. *For a function* $g \colon \mathbb{N} \to \mathbb{N}$*, define a promise problem* $\mathrm{Gap}_g\mathrm{Kt} := (\Pi_Y, \Pi_N)$ *by*

$$\Pi_Y := \{ (x, s) \in \{0,1\}^* \times \mathbb{N} \mid \mathrm{Kt}(x) \le s \},$$
$$\Pi_N := \{ (x, s) \in \{0,1\}^* \times \mathbb{N} \mid \mathrm{Kt}(x) > s + g(|x|) \}.$$

For this promise problem, we prove:

Theorem 4 (restated)**.** *For any nondecreasing function* $g(n) = \omega(\log n)$*, it holds that* $\mathsf{P}^{\mathrm{Gap}_g\mathrm{Kt}} \neq \mathsf{EXP}$.

The proof is similar to a simplified proof in [1, Corollary 40] showing that resource-bounded Kolmogorov complexity K^t for a fixed exponential time $t(n) \ge 2^{n^2}$ is not EXP-hard (originally proved by Buhrman and Mayordomo [8]).

Proof. It is sufficient to prove that every unary language in $\mathsf{P}^{\mathrm{Gap}_g\mathrm{Kt}}$ can be solved in a fixed exponential time. Indeed, by the time hierarchy theorem, there exists a unary language in EXP that requires time complexity larger than the fixed exponential time, which implies that $\mathsf{P}^{\mathrm{Gap}_g\mathrm{Kt}} \neq \mathsf{EXP}$.

We first note that $\mathrm{Kt}(x) \le |x| + O(\log |x|)$ for any $x \in \{0,1\}^*$, since every string can be described by itself in polynomial time. Let $l(n)$ be such a (nondecreasing) upper bound (*i.e.*, $l(n) = n + O(\log n)$).

Let $L \subseteq \{0\}^*$ be an arbitrary unary language in $\mathsf{P}^{\mathrm{Gap}_g\mathrm{Kt}}$, and M be a polynomial-time machine that witnesses $L \in \mathsf{P}^{\mathrm{Gap}_g\mathrm{Kt}}$.

The proof idea is as follows: We would like to simulate M on input 0^n without oracle access to $\mathrm{Gap}_g\mathrm{Kt}$ in time $2^{2n} \ll 2^{n^{O(1)}}$. To this end, we try to answer M's query q by exhaustively searching up to Kt-complexity $l(n)$. While we cannot obtain the correct value $\mathrm{Kt}(q)$ for a query q such that $\mathrm{Kt}(q) > l(n)$, we guess the value $\mathrm{Kt}(q)$ to be $l(n)$. Then, we will argue that each query q can be computed efficiently and hence $\mathrm{Kt}(q)$ is relatively small; therefore, the guessed value of Kt-complexity gives a good approximation. A formal proof follows.

We define a machine M_0 that simulates M on input 0^n (without oracle access to $\mathrm{Gap}_g\mathrm{Kt}$): On input 0^n, M_0 simulates M on the same input, and accepts if and only if M accepts. If the machine M makes a query $(q, s) \in \{0,1\}^* \times \mathbb{N}$ to a $\mathrm{Gap}_g\mathrm{Kt}$ oracle, then we perform an exhaustive search up to Kt-complexity $l(n)$, which allows us to compute $\sigma_n(q) := \min\{\,\mathrm{Kt}(q), l(n)\,\}$. (Namely, for each $d \in \{0,1\}^*$ of length at most $l(n)$, run the universal Turing machine U on input d for time $2^{l(n)-|d|}$, which takes overall $2^{l(n)}n^{O(1)}$ time.) We answer "Yes" to the query q if and only if $\sigma_n(q) \le s$. The machine M_0 runs in time $2^{l(n)}n^{O(1)} \le 2^{2n}$ (*i.e.,* a fixed exponential time). Hence, it remains to prove that, for each $n \in \mathbb{N}$, there exists an oracle A that satisfies the promise of $\mathrm{Gap}_g\mathrm{Kt}$ such that $M_0(0^n) = M^A(0^n)$, which in particular implies that $M_0(0^n) = L(0^n)$.

A crucial observation here is that each query that M makes on the computation path simulated by M_0 can be described succinctly in terms of Kt-complexity: Specifically, fix an input 0^n and define the set $Q_n = \{\,(q_1, s_1), \cdots, (q_m, s_m)\,\}$ of queries that M makes on the computation path simulated by M_0, where $m = n^{O(1)}$ is the number of the queries. Then, the ith query (q_i, s_i) can be described by n and an index $i \in [m]$ in time $2^{l(n)}n^{O(1)}$. Therefore, it holds that $\mathrm{Kt}(q_i) \le O(\log n) + \log 2^{l(n)}n^{O(1)} = l(n) + O(\log n)$. By the assumption, we have $O(\log n) \le g(n)$ for all large n; hence, $\mathrm{Kt}(q_i) \le l(n) + g(n)$. This means that the difference between $\mathrm{Kt}(q_i)$ and the threshold $l(n)$ up to which we performed an exhaustive search is at most $g(n)$.

Now, for each $n \in \mathbb{N}$, define an oracle A as follows: $(q, s) \in A$ if and only if $\sigma_n(q) \le s$ for any $(q, s) \in Q_n$, and $(q, s) \in A$ if and only if $\mathrm{Kt}(q) \le s$ for any $(q, s) \notin Q_n$. (Here, $\sigma_n(q)$ denotes $\min\{\,\mathrm{Kt}(q), l(n)\,\}$.) By this definition, it holds that $M^A(0^n) = M_0(0^n)$; thus all that remains is to show that A satisfies the promise of $\mathrm{Gap}_g\mathrm{Kt}$ (which implies that $M^A(0^n) = L(0^n)$).

Namely, for all $(q, s) \in Q_n$, we would like to claim that $(q, s) \in A$ holds if (q, s) is a YES instance of $\mathrm{Gap}_g\mathrm{Kt}$ (*i.e.,* $\mathrm{Kt}(q) \le s$), and that $(q, s) \notin A$ holds if (q, s) is a NO instance of $\mathrm{Gap}_g\mathrm{Kt}$ (*i.e.,* $\mathrm{Kt}(q) \ge s + g(|q|)$). Note that if $\mathrm{Kt}(q) \le l(n)$ then $\sigma_n(q) = \mathrm{Kt}(q)$; hence in this case, the claim is obviously satisfied. In what follows, we may assume that $\mathrm{Kt}(q) > l(n)$ (and thus $\sigma_n(q) = l(n)$). In particular, this implies that $n \le |q|$: indeed, by the definition of $l(n)$, we have $\mathrm{Kt}(q) \le l(|q|)$, which implies $l(n) < \mathrm{Kt}(q) \le l(|q|)$; hence, $n \le |q|$ follows. Therefore, $\mathrm{Kt}(q) \le l(n) + g(n) \le l(n) + g(|q|)$. Now assume that $\mathrm{Kt}(q) > s + g(|q|)$ (*i.e.,* (q, s) is a NO instance). This implies that $\sigma_n(q) = l(n) \ge \mathrm{Kt}(q) - g(|q|) > s$, and hence $(q, s) \notin A$ as desired. On the other hand, if $\mathrm{Kt}(q) \le s$ (*i.e.,* (q, s) is an YES instance), then we have $\sigma_n(q) \le \mathrm{Kt}(q) \le s$, and hence $(q, s) \in A$. □

Next, assuming that $\mathsf{EXP} \subseteq \mathsf{P/poly}$, we translate the property of Kt-complexity into that of MCSP. However, since these two measures are just *polynomially* related, the narrow gap of Kt does not seem to be translated into a narrow gap of MCSP. Thus, we define $\mathrm{Gap}^k\mathrm{MCSP}$ as a promise problem that asks for approximating the *logarithm* of circuit complexity within a factor of k:

Definition. *For a constant $k \ge 1$, define a promise problem $\mathrm{Gap}^k\mathrm{MCSP} := (\Pi_Y, \Pi_N)$ by*

$$\Pi_Y := \{\,(T, s) \in \{0,1\}^* \times \mathbb{N} \mid \log \mathrm{K}_I(T) \le s\,\},$$
$$\Pi_N := \{\,(T, s) \in \{0,1\}^* \times \mathbb{N} \mid \log \mathrm{K}_I(T) > ks\,\}.$$

We can apply the same proof idea to $\mathrm{Gap}^k\mathrm{MCSP}$. In fact, thanks to the fact that the gap between Π_Y and Π_N is wide, we can prove a somewhat strong consequence:

Theorem 23. *If $\mathsf{EXP} \subseteq \mathsf{P/poly}$, then for any $\epsilon > 0$, there exists a constant $k \ge 1$ such that $\mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}} \subseteq \mathsf{DTIME}(2^{n^\epsilon})$. In particular, $\mathsf{EXP} \ne \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}} \cap \mathsf{P/poly}$ for some k.*

Proof. The proof idea is exactly the same with that of Theorem 4: We first simulate a $\mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}}$ machine by answering its query T by an exhaustive search up to circuit complexity $l(n)$ for some

$l(n)$. Then, since any query T can be described succinctly in terms of Kt-complexity, the circuit complexity $\mathrm{K}_I(T)$ of the query T is also relatively small by Lemma 18; hence, the incomplete exhaustive search gives a somewhat good approximation. While the theorem can be proved based on Lemma 18, we incorporate a proof of Lemma 18 and give an entire proof below for completeness.

Let us define an EXP-complete language $B \subseteq \{0,1\}^*$ as all the tuples $\langle Q, x, t\rangle$ such that the Turing machine Q accepts x in time t. Since $B \in \mathsf{EXP} \subseteq \mathsf{P/poly}$, there exist some constant $k_0 \in \mathbb{N}$ and some family of circuits $\{ C_m \}_{m\in\mathbb{N}}$ of size at most m^{k_0} that computes B on input length m.

Fix a small constant $\epsilon > 0$. Define $k := (k_0+1)/\epsilon$. Let $L \in \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}}$ and M be a polynomial-time oracle machine that witnesses $L \in \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}}$.

Define $l(n) := n^\epsilon$. As in the proof of Theorem 4, we define a machine M_0 that simulates M (without oracle access to $\mathrm{Gap}^k\mathrm{MCSP}$) as follows: M_0 takes input $x \in \{0,1\}^*$ of length n, simulates M on input x, and accepts if and only if M accepts. If M makes a query (T, s), then answer to the query by an exhaustive search up to circuit size $l(n)$. (Specifically, compute $\sigma_x(T) := \min\{ \mathrm{K}_I(T), l(n) \}$ and answer "Yes" if and only if $\sigma_x(T) \le s$.) The machine M_0 runs in time $2^{l(n)} n^{O(1)} \le 2^{n^{2\epsilon}}$ for all large n.

Fix input $x \in \{0,1\}^*$ of length n. Let $Q_x = \{ (T_1, s_1), \cdots, (T_{n^{O(1)}}, s_{n^{O(1)}}) \}$ be the set of all the queries that M makes on the computation path simulated by M_0. We claim that for each $(T_i, s_i) \in Q_x$, the circuit complexity $\mathrm{K}_I(T_i)$ is relatively small: Indeed, each truth-table T_i in Q_x can be computed in time $t(n) := 2^{n^{2\epsilon}}$, by simulating M in the same way with M_0. Let Q be the Turing machine that takes as input $x \in \{0,1\}^*$ of length n and indices $i, j \in [n^{O(1)}]$, and outputs T_{ij}. By the definition of B, it holds that $B(Q, \langle x, i, j\rangle, t(n)) = Q(x, i, j) = T_{ij}$. Also, by the definition of C_m, we have $B(Q, \langle x, i, j\rangle, t(n)) = C_m(Q, \langle x, i, j\rangle, t(n))$ for $m = |\langle Q, \langle x, i, j\rangle, t(n)\rangle|$. Note that $m = 4n + O(\log n) + \log t(n) \le 5n$ for all large n. Now let us fix $x \in \{0,1\}^n$ and $i \in [n^{O(1)}]$: namely, define $D_{x,i}(j) = C_m(Q, \langle x, i, j\rangle, t(n))$; then, the truth-table of $D_{x,i}$ coincides with T_i. Therefore,

$$\mathrm{K}_I(T_i) \le |D_{x,i}| \le |C_m| \le m^{k_0} \le (5n)^{k_0} \le n^{k\epsilon} = l(n)^k$$

for all large n. (Here, $|C_m|$ denotes the circuit size of C_m.)

Now we claim that $\sigma_x(T_i) = \min\{ \mathrm{K}_I(T_i), l(n) \}$ approximates $\mathrm{K}_I(T_i)$ for all $(T_i, s_i) \in Q_x$: specifically, we claim that $\log \sigma_x(T_i) \le \log \mathrm{K}_I(T_i) < k \log \sigma_x(T_i)$. If $\mathrm{K}_I(T_i) \le l(n)$, then $\sigma_x(T_i) = \mathrm{K}_I(T_i)$ and the claim is obvious. Now assume that $\mathrm{K}_I(T_i) > l(n)$, which implies that $\sigma_x(T_i) = l(n)$. Thus we have $\sigma_x(T_i) = l(n) < \mathrm{K}_I(T_i) < l(n)^k = \sigma_x(T_i)^k$.

From the inequalities above, for all but finitely many $x \in \{0,1\}^*$, it is easy to see that there exists an oracle A such that A satisfies the promise of $\mathrm{Gap}^k\mathrm{MCSP}$ and $M_0(x) = M^A(x) = L(x)$. □

As in Corollary 22, we obtain:

Corollary 24. *If $L \le_T^p \mathrm{Gap}^k\mathrm{MCSP}$ for all $k \ge 1$, then $\mathsf{P}^L \cap \mathsf{P/poly} \neq \mathsf{EXP}$.*

Proof. The hypothesis implies that $\mathsf{P}^L \subseteq \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}}$ for all $k \ge 1$, and Theorem 23 shows $\mathsf{EXP} \not\subseteq \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}} \cap \mathsf{P/poly}$ for some $k \ge 1$, from which the result follows. □

Remark. 1. As in the case of nonadaptive reductions, establishing NP-hardness of $\mathrm{Gap}^k\mathrm{MCSP}$ for all $k \ge 1$ via a polynomial-time Turing reduction implies that $\mathsf{P}^{\mathsf{NP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$.

2. One interesting consequence is that if MCSP itself is reducible to $\mathrm{Gap}^k\mathrm{MCSP}$ for all $k \geq 1$ via a polynomial-time Turing reduction, then $\mathsf{P}^{\mathsf{MCSP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$, which we do not know how to prove. Thus, establishing such "robustness" of MCSP via a polynomial-time Turing reduction is at least as hard as separating $\mathsf{P}^{\mathsf{MCSP}} \cap \mathsf{P/poly}$ from EXP.

Finally, we observe that every language in statistical zero knowledge is reducible to $\mathrm{Gap}^k\mathrm{MCSP}$ via a BPP-reduction. As observed in [5], hardness of statistical zero knowledge implies hardness of approximating the minimum circuit complexity of a truth-table T within a factor of $|T|^{1-\epsilon}$ for any $\epsilon \in (0,1)$. Similarly, it implies hardness of $\mathrm{Gap}^k\mathrm{MCSP}$ for all $k \geq 1$ (*i.e.,* a problem of approximating the logarithm of the circuit complexity within an arbitrary constant factor).

Theorem 6 (restated)**.** *For all $k \geq 1$, every language in statistical zero knowledge is reducible to* $\mathrm{Gap}^k\mathrm{MCSP}$ *via a* BPP*-Turing reduction.*

Proof. Let A be an arbitrary oracle that satisfies the promise of $\mathrm{Gap}^k\mathrm{MCSP}$. Let $s(n) := \frac{1}{2k}\log n$. Define an oracle $B := \{\, x \in \{0,1\}^* \mid (x, s(|x|)) \notin A \,\}$. It is sufficient to show that B satisfies the hypothesis of Theorem 7.

First, we claim that B does not contain any string of low circuit complexity. Suppose that $x \in B$. Then we have $(x, s(|x|)) \notin A$, which implies that $(x, s(|x|))$ is not an YES instance of $\mathrm{Gap}^k\mathrm{MCSP}$. This means that $\log \mathrm{K}_I(x) > s(|x|)$; hence, $\mathrm{K}_I(x) > |x|^{1/2k}$.

Next, we claim that the oracle B is of polynomial density. It is sufficient to prove that $\{\, x \in \{0,1\}^* \mid \mathrm{K}_I(x) > |x|^{1/2} \,\} \subseteq B$: Indeed, suppose that $\mathrm{K}_I(x) > |x|^{1/2}$ for a string $x \in \{0,1\}^*$; then we have $\log \mathrm{K}_I(x) > ks(|x|)$, which implies that $(x, s(|x|))$ is a NO instance of $\mathrm{Gap}^k\mathrm{MCSP}$; hence, $x \in B$. $\square$

Acknowledgements We would like to thank anonymous referees of CCC 2016 for their helpful comments and suggestions. This work is supported in part by MEXT KAKENHI No. 24106008.

References

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

[2] Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 25–32, 2014.

[3] Eric Allender, Joshua Grochow, and Cristopher Moore. Graph isomorphism and circuit size. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:162, 2015.

[4] Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. In *Proceedings of 32nd International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 21–33, 2015.

[5] 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.

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

[7] Ravi Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Inf. Process. Lett.*, 25(2):127–132, 1987.

[8] Harry Buhrman and Elvira Mayordomo. An excursion to the Kolmogorov random strings. *J. Comput. Syst. Sci.*, 54(3):393–399, 1997.

[9] Lance Fortnow. The complexity of perfect zero-knowledge. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 204–209, 1987.

[10] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 59–68, 1986.

[11] Juris Hartmanis and Richard Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, pages 285–306, 1965.

[12] Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[13] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 220–229, 1997.

[14] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.

[15] Ker-I Ko. On the complexity of learning minimum time-bounded turing machines. *SIAM J. Comput.*, 20(5):962–986, 1991.

[16] Leonid Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.

[17] Cody Murray and Ryan Williams. On the (non) NP-hardness of computing circuit complexity. In *Proceedings of 30th Conference on Computational Complexity (CCC)*, pages 365–380, 2015.

A Heavy Samples Protocol

For completeness, we show how to use the lower bound protocol and the upper bound protocol in order to estimate the probability that a sampled element is heavy.

Lemma 13 (The heavy samples protocol; Trevisan and Bogdanov [6] (restated))**.** *Let $\mathcal{D} = \{ \mathcal{D}_n \}_{n\in\mathbb{N}}$ be a polynomial-time samplable distribution. There exist a universal constant c ($c = 2^{11}$ will do) and an $\mathsf{AM} \cap \mathsf{coAM}$ protocol that solves the following promise problem: Given input 1^n and a threshold $\beta \in [0, 1]$, accept if $\Pr_{y\sim\mathcal{D}_n}[\ y \text{ is } c\beta\text{-heavy}\ ] \geq \frac{3}{4}$, and reject if $\Pr_{y\sim\mathcal{D}_n}[\ y \text{ is } \beta\text{-heavy}\ ] \leq \frac{1}{4}$.*

Note that we may assume that β is a discrete value: indeed, since $\mathcal{D}$ has a polynomial-time sampler, the probability that an element is sampled is a multiple of $2^{-\mathsf{poly}(n)}$ for some polynomial $\mathsf{poly}(n)$.

Proof. Let S be a polynomial-time sampler of $\mathcal{D}$. Namely, given an input 1^n and a random coin flip of length m, the probabilistic polynomial-time machine S outputs a sample of length n from $\mathcal{D}_n$.

First, we show an AM protocol by using the lower bound protocol. The AM protocol works as follows: Sample $y \sim \mathcal{D}_n$. Run the lower bound protocol to check whether $|S_n^{-1}(y)| \geq 2^m \cdot c\beta$ or $|S_n^{-1}(y)| \leq 2^m \cdot \beta$, and accept if and only if the lower bound protocol succeeds. Here, $S_n^{-1}(y)$ denotes $\{ r \in \{0,1\}^m \mid S(1^n, r) = y \}$. The correctness follows from the following lemma:

Lemma 25 (Lower bound protocol; Goldwasser and Sipser [10])**.** *There exists an* AM *protocol such that, given an input 1^n, a threshold $s \in \mathbb{N}$ and a string $y \in \{0,1\}^n$,*

1. *if $|S_n^{-1}(y)| \geq cs$, then the verifier accepts with high probability for some prover, and*
2. *if $|S_n^{-1}(y)| \leq s$, then the verifier rejects with high probability for any prover.*

We claim the correctness of the AM protocol. If $\Pr_{y \sim \mathcal{D}_n}[\, y \text{ is } c\beta\text{-heavy} \,] \geq \frac{3}{4}$, then with probability at least $\frac{3}{4}$, the sampled string $y \sim \mathcal{D}_n$ is $c\beta$-heavy. Conditioning on this, the lower bound protocol accepts with high probability (say, with probability at least $\frac{8}{9}$). Therefore, the overall acceptance probability is at least $\frac{3}{4} \cdot \frac{8}{9} = \frac{2}{3}$. A similar argument applies to the case when $\Pr_{y \sim \mathcal{D}_n}[\, y \text{ is } \beta\text{-heavy} \,] \leq \frac{1}{4}$.

Next, we show a coAM protocol by using the upper bound protocol. In order to apply the upper bound protocol for checking whether $|S_n^{-1}(y)| \leq s$ for $s = 2^m \beta$, we need to sample a random element $r \in_R S_n^{-1}(y)$ that is not known to a prover. Thus, we first sample a random element $r \in_R \{0,1\}^m$ privately, and then define $y := S(1^n, r)$. (Note that y is distributed according to $\mathcal{D}_n$.) Run the upper bound protocol to check whether $|S_n^{-1}(y)| \leq s$ or $|S_n^{-1}(y)| \geq cs$, and accept if and only if the upper bound protocol succeeds.

The correctness of the coAM protocol follows from the same argument as the AM protocol and the following lemma.

Lemma 26 (Upper bound protocol; Fortnow [9])**.** *There exists an* AM *protocol satisfying the following: Suppose that a verifier has a random element $r \in_R S_n^{-1}(y)$ that is not known by a prover. Given an input 1^n, a threshold $s \in \mathbb{N}$, and a string $y \in \{0,1\}^n$,*

1. *if $|S_n^{-1}(y)| \leq s$, then the verifier accepts with probability at least $\frac{15}{16}$ for some prover, and*
2. *if $|S_n^{-1}(y)| \geq cs$, then the verifier rejects with probability at least $\frac{15}{16}$ for any prover.*

Since a random element r is chosen uniformly at random, it is also distributed uniformly on $S_n^{-1}(y)$, conditioning on the event that $S(1^n, r) = y$. Thus, the hypothesis of the upper bound protocol is satisfied. $\square$

ECCC ISSN 1433-8092

http://eccc.hpi-web.de