Electronic Colloquium on Computational Complexity, Report No. 10 (2021)

Cryptographic Hardness under Projections for Time-Bounded Kolmogorov Complexity

Eric Allender
Rutgers University, NJ, USA
http://www.cs.rutgers.edu/~allender
allender@cs.rutgers.edu

John Gouwar
Grinnell College, Iowa, USA
gouwarjo@grinnell.edu

Shuichi Hirahara
National Institute of Informatics, Japan
https://researchmap.jp/shuichi.hirahara/?lang=english
s_hirahara@nii.ac.jp

Caleb Robelle
University of Maryland, Baltimore County, USA
carobel1@umbc.edu

Abstract

A version of time-bounded Kolmogorov complexity, denoted KT, has received attention in the past several years, due to its close connection to circuit complexity and to the Minimum Circuit Size Problem MCSP. Essentially all results about the complexity of MCSP hold also for MKTP (the problem of computing the KT complexity of a string). Both MKTP and MCSP are hard for SZK (Statistical Zero Knowledge) under BPP-Turing reductions; neither is known to be NP-complete.

Recently, some hardness results for MKTP were proved that are not (yet) known to hold for MCSP. In particular, MKTP is hard for DET (a subclass of P) under nonuniform $\leq_m^{\mathsf{NC}^0}$ reductions. In this paper, we improve this, to show that $\overline{\mathsf{MKTP}}$ is hard for the (apparently larger) class $\mathsf{NISZK_L}$ under not only $\leq_m^{\mathsf{NC}^0}$ reductions but even under projections. Also $\overline{\mathsf{MKTP}}$ is hard for NISZK under $\leq_m^{\mathsf{P/poly}}$ reductions. Here, NISZK is the class of problems with non-interactive zero-knowledge proofs, and $\mathsf{NISZK_L}$ is the non-interactive version of the class $\mathsf{SZK_L}$ that was studied by Dvir et al.

As an application, we provide several improved worst-case to average-case reductions to problems in NP.

2012 ACM Subject Classification Complexity Classes; Problems, reductions and completeness; Circuit complexity

Keywords and phrases Kolmogorov Complexity, Interactive Proofs, Minimum Circuit Size Problem, Worst-case to Average-case Reductions

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Eric Allender*: Supported in part by NSF Grant CCF-1909216.

Acknowledgements EA was supported in part by NSF Grant CCF-1909216. JG and CR were supported in part by the DIMACS 2020 REU program under NSF grant CCF-1836666. We thank Oded Goldreich, Salil Vadhan, Harsha Tirumala, Dieter van Melkebeek and Andrew Morgan for helpful discussions.

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:25
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

ISSN 1433-8092

1 Introduction

The study of time-bounded Kolmogorov complexity is tightly connected to the study of circuit complexity. Indeed, the measure that we study most closely in this paper, denoted KT, was initially defined in order to capitalize on the framework of Kolmogorov complexity in investigations of the Minimum Circuit Size Problem (MCSP) [4]. If f is a bit string of length 2^k representing the truth-table of a k-ary Boolean function, then $\mathsf{KT}(f)$ is polynomially related to the size of the smallest circuit computing f. Thus the problem of computing KT complexity (denoted MKTP) was initially viewed as a more-or-less equivalent encoding of MCSP, and it is still the case that all theorems that have been proved about the complexity of MCSP hold also for MKTP (such as those in [5, 8, 9, 14, 18–21, 26, 27, 29, 31]).

In recent years, however, a few hardness results were proved for MKTP that are not yet known to hold for MCSP [6, 7]. We believe that these results can be taken as an indication of what is likely to be true also for MCSP. The present work gives significantly improved hardness results for MKTP.

Reducibility and completeness are the most effective tools in the arsenal of complexity theory for giving evidence of intractability. However, it is not clear whether MCSP or MKTP is NP-complete; neither can be shown to be NP-complete – or even hard for ZPP – under the usual $\leq_m^{\mathsf{P}}$ reductions without first showing that $\mathsf{EXP} \neq \mathsf{ZPP}$, a long-standing open problem [14, 27].

The strongest hardness results that have been proved thus far for MCSP and MKTP are that both are hard for SZK under BPP-Turing reductions [5]. SZK is the class of problems that have Statistical Zero Knowledge Interactive Proofs, and contains many problems of interest to cryptographers. Indeed, if MCSP (or MKTP) is in P/poly, then there are no cryptographically-secure one-way functions [23].

SZK is not known to be contained in NP; until such a containment can be established, there is no hope of improving the BPP-Turing reduction of [5] to a $\leq_m^{\mathsf{P}}$ reduction. But we come close in this paper. NISZK is the "non-interactive" subclass of SZK; it contains intractable problems if and only if SZK does [15]. We show that $\overline{\mathsf{MKTP}}$ is hard for NISZK under $\leq_m^{\mathsf{P/poly}}$ reductions. (Thus, instead of asking many queries, as in [5], a single query suffices.) Our proof also shows that MKTP is hard for NISZK under BPP reductions that ask only one query. Combined with [15], this shows that MKTP is hard for SZK under *non-adaptive* BPP reductions, yielding a modest improvement over [5]; this has implications regarding the study of worst-case to average-case reductions. (See Section 1.1.)

But $\leq_m^{\mathsf{P/poly}}$ reductions are still quite powerful. There is great interest currently in proving lower bounds for MCSP, MKTP, and related problems such as MKtP (the problem of computing a different kind of time-bounded Kolmogorov complexity, due to Levin [25]) on very limited classes of circuits and formulae, as part of the "hardness magnification" program. For instance, if modest lower bounds can be shown on the size required to compute MKtP on de Morgan formulae augmented with PARITY gates at the leaves, then EXP is not contained in non-uniform NC^1 [28]. Also, there is great interest in finding lower bounds against a variety of other models, such as depth-three threshold gates, or circuits consisting of polynomial threshold gates [24]. If a lower bound is known against one of these limited classes of circuits for some problem A that is reducible to, say, MKTP or MKtP under $\leq_m^{\mathsf{P/poly}}$ reductions, it implies nothing about the complexity of MKTP or MKtP, since the circuitry involved in computing the reduction is much more powerful than the circuitry in the class of circuits for which the lower bound is known.

Thus there is a great deal of interest in considering reductions that are much less powerful than $\leq_m^{\mathsf{P/poly}}$ reductions. For extremely weak (uniform) notions of reducibility (such as log-time reductions), it is known that MCSP and MKTP are *not* hard for any complexity class that contains the PARITY function [27]. However, this non-hardness result relies on uniformity; it was later shown that MKTP is hard for the complexity class DET under *nonuniform* $\leq_m^{\mathsf{NC}^0}$ reductions [7].

However, even $\leq_m^{\mathsf{NC}^0}$ reductions are too powerful a tool, when one is interested in lower bounds against the classes of circuits discussed above, since they do not seem to be closed under $\leq_m^{\mathsf{NC}^0}$ reductions. This motivates consideration of the most restrictive type of reduction that we will be considering: projections.

A projection is a reduction that is computed by a circuit consisting only of wires and NOT gates. Each output bit is either a constant, or is connected by a wire to a (possibly negated) input bit. All of the classes of circuits mentioned above (and – indeed – most conceivable classes of circuits) are closed under projections.

Prior to our work, the result of [7] showing that MKTP is hard for DET under $\leq_m^{\mathsf{NC}^0}$ reductions was improved, to show that MKTP is hard for DET even under projections [3]. Since DET is a subclass of P, this provides little ammunition when one is seeking to prove that MKTP is intractable. One of our main contributions is to show that $\overline{\mathsf{MKTP}}$ is hard for $\mathsf{NISZK_L}$ under projections.

The reader will not be familiar with $\mathsf{NISZK_L}$; this complexity class makes its first appearance in the literature here. It is the "non-interactive" counterpart to the complexity class $\mathsf{SZK_L}$ that was studied previously by Dvir et al. [13], and was shown there to contain several important natural problems of interest to cryptographers (such as Discrete Log and Decisional Diffie-Hellman). $\mathsf{NISZK_L}$ contains intractable problems if and only if $\mathsf{SZK_L}$ does (see Section 2). Thus, for the first time, we show that MKTP is hard under projections for a complexity class that is widely believed to contain intractable problems. Our hardness results carry over immediately to MKtP and to similar problems defined in terms of general Kolmogorov complexity; no hardness results under projections had been known previously for those problems. We present some complete problems for $\mathsf{NISZK_L}$ and establish some other basic facts about this class in Section 4.

1.1 Average-Case Complexity

Building on the techniques introduced in [17], we are able to establish new insights regarding the relationship between worst-case and average-case complexity. In Theorem 47, capitalizing on the fact that essentially every circuit complexity class $\mathcal{C}$ is closed under projections, we show that if $\mathsf{NISZK_L}$ does not lie in $\mathsf{OR} \circ \mathcal{C}$, then there are problems A in NP that cannot be solved *in the average case* by errorless heuristics in $\mathcal{C}$. For instance, if one were able to show that some of the candidate one-way functions in $\mathsf{NISZK_L}$ cannot be solved by depth-four ACC^0 circuits, it would follow that there are problems in NP that are hard-on-average for depth-three ACC^0 circuits. Such conclusions would *not* follow if our reductions to MKTP had merely been computable in AC^0 or NC^0.

We are also able to shed more light on worst-case to average-case reductions, in the form that they were studied by Bogdanov and Trevisan [12]. Bogdanov and Trevisan showed that there were severe limits on the complexity of problems whose worst-case complexity could be reduced to the average-case complexity of problems in NP via *non-adaptive* reductions; all such problems lie in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. But it was not known how large this class of problems could be. Hirahara showed that every problem in SZK has an *adaptive* worst-case to average-case reduction to a problem in NP, but the upper bound of $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ proved by Bogdanov and Trevisan does not apply for adaptive reductions. As a consequence of our Corollary 19, showing that MKTP is hard for SZK under nonadaptive BPP reductions, we are able to show (in Corollary 50) that the class identified by Bogdanov and Trevisan lies in the narrow range between SZK and NP/poly ∩ coNP/poly.

Remark: This is an illustration of the utility of studying MKTP, as an example of a theorem that does not explicitly mention MKTP or MCSP, but which was proved via the study of MKTP. No such argument based on MCSP is known. We believe that MKTP can in fact be viewed as a *particularly convenient* formulation of MCSP, since (a) KT complexity is closely related to circuit size, (b) essentially all theorems known to hold for MCSP also hold for MKTP, (c) some arguments that one might intend to formulate in terms of MCSP elude current approaches, but can instead be successfully carried through by use of MKTP instead. Furthermore, theorems proved for MKTP may serve as an indication of what is likely to be true for MCSP as well.

The rest of the paper is organized as follows: Our $\leq_m^{\mathsf{P/poly}}$-hardness theorem for MKTP is proved in Section 3. Then, after establishing some basic facts about $\mathsf{NISZK_L}$ in Section 4, in Section 5 we show that $\overline{\mathsf{MKTP}}$ is hard for $\mathsf{NISZK_L}$ under projections. We present applications of our reductions and implications for average-case complexity in Section 6.

2 Preliminaries

2.1 Complexity Classes and Reducibilities

We assume familiarity with the complexity classes P, NP, L, BPP, and P/poly. We also make use of the circuit complexity classes AC^0 and NC^0. For the purposes of this paper, AC^0 can be understood as the set of problems for which there is a family of circuits $\{C_n : n \in \mathbb{N}\}$ with unbounded-fan-in AND and OR gates (and NOT gates of fan-in 1) of polynomial size and constant depth. NC^0 is defined similarly, but with AND and OR gates of bounded fan-in (and thus each output bit depends on only a constant number of bits of the input). We deal primarily with the "nonuniform" versions of these complexity classes (which means that the mapping $n \mapsto C_n$ need not be computable).

Branching programs are a circuit-like model of computation that can be used to characterize logspace computation. A *branching program* is a directed acyclic graph with a single source and two sinks labeled 1 and 0, respectively. Each non-sink node in the graph is labeled with a variable in $\{x_1, \ldots, x_n\}$ and has two edges leading out of it: one labeled 1 and one labeled 0. A branching program computes a Boolean function f on input $x = x_1 \ldots x_n$ by first placing a pebble on the source node. At any time when the pebble is on a node v labeled x_i, the pebble is moved to the (unique) vertex u that is reached by the edge labeled 1 if $x_i = 1$ (or by the edge labeled 0 if $x_i = 0$). If the pebble eventually reaches the sink labeled b, then $f(x) = b$. Branching programs can also be used to compute functions $f : \{0,1\}^m \to \{0,1\}^n$, by concatenating n branching programs $p_1, \ldots, p_n$, where p_i computes the function $f_i(x) =$ the i-th bit of $f(x)$. For more information on the definitions, backgrounds, and nuances of these complexity classes, circuits, and branching programs, see the text by Vollmer [32].

A *promise problem* Π is a pair of disjoint sets (Π_{YES}, Π_{NO}). A *solution* to a promise problem is any set A such that $\Pi_{YES} \subseteq A$ and $\Pi_{NO} \subseteq \overline{A}$. A *don't-care instance* of Π is any string that is not in $\Pi_{YES} \cup \Pi_{NO}$. A *language* A can be viewed as a promise problem that has no don't-care instances.

Given any class $\mathcal{C}$ of functions, there is an associated notion of *m-reducibility* or *many-one reducibility*: For two languages A and B, we say that $A \leq_m^{\mathcal{C}} B$ if there is a function f in $\mathcal{C}$ such that $x \in A$ iff $f(x) \in B$. This notion of reducibility extends naturally to promise problems, mapping yes-instances to yes-instances, and no-instances to no-instances. The most familiar notion of m-reducibility is Karp reducibility: $\leq_m^{\mathsf{P}}$; NP-completeness is most commonly defined in terms of Karp reducibility. However, in this paper, we will frequently be reducing problems that are not known to reside in NP to MKTP, which does lie in NP. Thus it is clear that a more powerful notion of reducibility is required. Some of our results are most conveniently stated in terms of $\leq_m^{\mathsf{P/poly}}$ reductions (i.e., reductions computed by nonuniform polynomial-size circuits). We also consider restrictions of $\leq_m^{\mathsf{P/poly}}$ reductions, computed by nonuniform AC^0 and NC^0 circuits: $\leq_m^{\mathsf{AC}^0}$ and $\leq_m^{\mathsf{NC}^0}$. Finally we also consider *projections* ($\leq_m^{\mathsf{proj}}$), which are functions computed by NC^0 circuits that have only NOT gates. That is, in a projection, each output bit is either a constant 0 or 1, or is connected by a wire to an input bit or its negation.

We will also make reference to various types of *Turing reducibility*, which are defined in terms of oracle Turing machines, or in terms of circuit families that are augmented with "oracle gates". For instance, we say that $A \leq_T^{\mathsf{BPP}} B$ if there is a probabilistic polynomial time oracle Turing machine M with oracle B that accepts every $x \in A$ with probability $\frac{2}{3}$ and rejects every $x \in \overline{A}$ with probability $\frac{2}{3}$. Note that the computation tree of such a BPP-Turing reduction can contain an exponential number of queries to different elements of B. Just as $\mathsf{BPP} \subseteq \mathsf{P/poly}$, it also holds that $A \leq_T^{\mathsf{BPP}} B$ implies $A \leq_T^{\mathsf{P/poly}} B$. Thus, on any input x, the circuit computing the P/poly-Turing reduction queries only a polynomial number of elements of B. It was shown in [5] that every problem in SZK (that is, every problem with a statistical zero knowledge proof system) is $\leq_T^{\mathsf{BPP}}$-reducible (and hence $\leq_T^{\mathsf{P/poly}}$-reducible) to MCSP and to MKTP. The question of interest to us here is: Is it necessary to ask so many queries? What can we do if we ask only one query? What can be reduced to MKTP via a $\leq_m^{\mathsf{P/poly}}$ reduction?

The complexity class with which we are primarily concerned in this paper is the class of problems that have non-interactive statistical zero knowledge proof systems: NISZK. NISZK was originally defined and studied by Blum et al. [11]. The definition below (in terms of promise problems) is due to Goldreich et al. [15].

▶ **Definition 1.** *A* non-interactive statistical zero-knowledge proof system *for a promise problem Π is defined by a triple of probabilistic machines P, V, and S, where V and S are polynomial-time and P is computationally unbounded, and a polynomial $r(n)$ (which will give the size of the random reference string σ), such that:*

1. *(Completeness:) For all $x \in \Pi_{YES}$, the probability that $V(x, \sigma, P(x, \sigma))$ accepts is at least $1 - 2^{-|x|}$.*
2. *(Soundness:) For all $x \in \Pi_{NO}$, the probability that $V(x, \sigma, P(x, \sigma))$ accepts is at most $2^{-|x|}$.*
3. *(Zero Knowledge:) For all $x \in \Pi_{YES}$, the statistical distance between the following two distributions bounded by $1/\beta(|x|)$*

(A) Choose σ uniformly from $\{0,1\}^{r(|x|)}$, sample p from $P(x, \sigma)$, and output (p, σ).
(B) S(x) (where the coins for S are chosen uniformly at random.)

where $\beta(n)$ is superpolynomial, and the probabilities in Conditions 1 and 2 are taken over the random coins of V and P, and the choice of σ uniformly from $\{0,1\}^{r(n)}$.

NISZK *is the class of promise problems for which there is a non-interactive statistical zero knowledge proof system.*

NISZK is not known to be closed under complementation; co-NISZK is defined as the class of promise problems $\Pi = (\Pi_{YES}, \Pi_{NO})$ such that (Π_{NO}, Π_{YES}) is in NISZK. It is known that SZK = NISZK iff NISZK = co-NISZK, and that every promise problem in SZK efficiently (and non-adaptively) Turing-reduces to a problem in NISZK [15]. Thus NISZK contains intractable problems if and only if SZK does.

A subclass of SZK, which we will denote by $\mathsf{SZK_L}$, in which the verifier V and simulator S are restricted to being logspace machines, was defined and studied by Dvir et al. [13]. Among other things, they showed that many of the important natural problems in SZK lie in $\mathsf{SZK_L}$, including Graph Isomorphism, Quadratic Residuosity, Discrete Log, and Decisional Diffie-Helman. The non-interactive version of $\mathsf{SZK_L}$, which we denote by $\mathsf{NISZK_L}$, has not been studied previously, but it figures prominently in our results.

▶ **Definition 2.** *The formal definition of* $\mathsf{NISZK_L}$ *is obtained by replacing each occurrence of "polynomial-time" in Definition 1 with "logspace". (It is important to note that, in this model, the logspace-bounded verifier V and simulator S are allowed two-way access to the reference string σ and to their polynomially-long sequences of probabilistic coin flips.)*

The reduction presented in [15] carries over directly to the logspace setting, showing that $\mathsf{NISZK_L}$ contains intractable problems if and only if $\mathsf{SZK_L}$ does. In particular, we have:

▶ **Proposition 3.** *Every promise problem in* $\mathsf{SZK_L}$ *is non-adaptively* AC^0*-Turing-reducible a problem in* $\mathsf{NISZK_L}$*.*

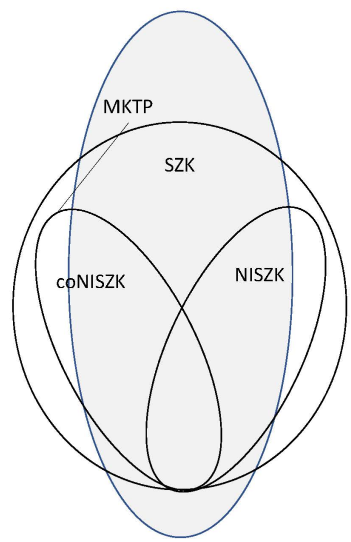

Figure 1 Diagram showing the classes NISZK, co-NISZK, and SZK. The shaded oval represents NP. Every problem in co-NISZK is $\leq_{\mathrm{m}}^{\mathsf{P/poly}}$-reducible to MKTP.

2.2 KT Complexity

The measure KT was defined in [4]. We provide a reproduction of that definition below.

▶ **Definition 4** (KT). *Let U be a universal Turing machine. For each string x, define* $\mathrm{KT}_U(x)$ *to be*

$$\min\{|d| + T : (\forall \sigma \in \{0, 1, *\})\ (\forall i \leq |x| + 1)\ U^d(i, \sigma) \textit{ accepts in } T \textit{ steps iff } x_i = \sigma\}$$

We define $x_i = *$ if $i > |x|$; thus, for $i = |x| + 1$ the machine accepts iff $\sigma = *$. The notation U^d indicates that the machine U has random access to the description d.

To understand the motivation for this definition, see [4]. The minimum KT problem, henceforth MKTP, is defined below.

▶ **Definition 5** (MKTP). *Suppose $y \in \{0,1\}^n$ and $\theta \in \mathbb{N} \setminus \{0\}$, then*

$$\mathsf{MKTP} = \{(y, \theta) \mid \mathrm{KT}(y) \leq \theta\}.$$

In this paper when we view MKTP as a promise problem, yes-instances will be considered those that are in the language, and no-instances those that are not in the language.

2.3 Discrete Probability and Entropy

▶ **Definition 6.** *Discrete Random Variables and Distributions*

- *A* random variable *$R : S \to T$ is a function where S is a finite set with a probability distribution on its elements. We will refer to S as the* sample space. *R with a uniform distribution on S will induce a distribution p on T.*
- *The* support *of a distribution is the set of elements in the distribution with positive probability. Alternatively, the support of a random variable R can be understood as the set $\mathrm{Im}(R)$.*
- *In an abuse of notation, often given a distribution X, we will refer to X as both the random variable that induces the distribution, and the distribution itself.*
- *Given a distribution X, we will use the notation X^k to denote the $k-$fold direct product of X. Alternatively, this can be understood as the concatenation of k independent copies of X.*

Given a function $f : \{0,1\}^m \to \{0,1\}^n$ we write U_m to denote the uniform distribution on m bits, and $f(U_m)$ for the output distribution of f when evaluated on a uniformly chosen element of $\{0,1\}^m$. Throughout this paper, our random variables, and in turn the distributions they induce, will be of the form $C(U_m)$, where C is a multi-output Boolean circuit $C : \{0,1\}^m \to \{0,1\}^n$.

The entropy of a distribution can be understood informally as measuring how much "randomness" is present in the distribution.

▶ **Definition 7.** *Suppose X is a distribution. The Shannon entropy of X (denoted $H(X)$) is the expected value of $1/\log(\Pr[X = x])$.*

3 $\overline{\mathsf{MKTP}}$ is Hard For NISZK

In this section, we prove our first hardness result for MKTP; MKTP is hard for co-NISZK under $\leq_{\mathrm{m}}^{\mathsf{P/poly}}$ reductions. In order to prove hardness, it suffices to provide a reduction from the *entropy approximation* problem: EA, which is known to be complete for NISZK under $\leq_{\mathrm{m}}^{\mathsf{P}}$ reductions [15].

▶ **Definition 8** (Promise-EA). *Let a circuit $C : \{0,1\}^m \to \{0,1\}^n$ represent a probability distribution X on $\{0,1\}^n$ induced by the uniform distribution on $\{0,1\}^m$. We define Promise-EA to be the promise problem*

$$\mathsf{EA}_{YES} = \{(C, k) \mid H(X) > k + 1\}$$
$$\mathsf{EA}_{NO} = \{(C, k) \mid H(X) < k - 1\}$$

where $H(X)$ denotes the entropy of X.

We will make use of some machinery that was developed in [6], in order to relate the entropy of a distribution to the KT complexity of samples taken from the distribution. However, these tools are only useful when applied to distributions that are sufficiently "flat". The next subsection provides the necessary tools to "flatten" a distribution.

3.1 Flat Distributions

A distribution is considered *flat* if it is uniform on its support. Goldreich et al. [15] formalized a relaxed notion of flatness, termed Δ-flatness, which relies on the concept of Δ-typical elements. The definitions of both concepts follow:

▶ **Definition 9** (Δ-typical elements). *Suppose X is a distribution with element x in its support. We say that x is Δ-typical if,*

$$2^{-\Delta} \cdot 2^{-H(X)} < \Pr[X = x] < 2^{\Delta} \cdot 2^{-H(X)}.$$

▶ **Definition 10** (Δ-flatness). *Suppose X is a distribution. We say that X is Δ-flat if for every $t > 0$ the probability that an element of the support, x, is $t \cdot \Delta$-typical is at least $1 - 2^{-t^2+1}$.*

▶ **Lemma 11** (Flattening Lemma). *[15] Suppose X is a distribution such that for all x in its support $\Pr[X = x] \geq 2^{-m}$. Then X^k is $(\sqrt{k} \cdot m)$-flat.*

Observe that if X is a distribution represented by a circuit $C : \{0,1\}^m \to \{0,1\}^n$, then the hypothesis of the Flattening Lemma holds for m. Note also that, for any distribution X, $H(X^k) = k \cdot H(X)$. Thus the entropy of the distribution X^k grows linearly with respect to k, while the deviation from flatness diminishes much more rapidly with respect to k.

3.2 Encoding and Blocking

The *Encoding Lemma* is the primary tool that was developed in [6] to give short descriptions of samples from a given distribution. Below, we give a precise statement of the version of the Encoding Lemma that is stated informally as Remark 4.3 of [6]. (Although the statement there is informal, the proof of the Encoding Lemma that is given there does yield our Lemma 13.) First, we need to define Λ-encodings.

▶ **Definition 12** (Λ-encodings). *Let $R : S \to T$ be a random variable that induces a distribution X. The* Λ-heavy *elements of T are those elements λ such that $\Pr[X = \lambda] > 1/2^{\Lambda}$. A Λ-encoding of R is given by a mapping $D : [N] \to S$ such that for every Λ-heavy element λ, there exists $i \in [N]$ such that $R(D(i)) = \lambda$. We refer to $\lceil \log(N) \rceil$ as the length of the encoding. The function D is also called the* decoder *for the encoding.*

▶ **Lemma 13** (Encoding Lemma). *[6, Lemma 4.1] Consider an ensemble $\{R_x\}$ of random variables that sample distributions on strings of some length $\mathrm{poly}_1(|x|)$, where there are circuits C_x of size $\mathrm{poly}_2(|x|)$ representing each R_x. Then there is a polynomial poly_3 such that, for every integer Λ, each R_x has a Λ-encoding of length $\Lambda + \log(\Lambda) + O(1)$ that is decodable by circuits of size $\mathrm{poly}_3(|x|)$.*

By itself, the Encoding Lemma says nothing about KT complexity. The other important ingredient in the toolbox developed in [6] is the *Blocking Lemma*, which refers to the process of chopping a string into blocks. Let y be a string of length tn, which we think of as being the concatenation of t samples y_i of a distribution X on strings of length n. Thus $y = y_1 \ldots y_t$.

Let $r = \lceil t/b \rceil$. Equivalently, we consider y to be equal to $z_1 \ldots z_r$ where each z_i is a string of length bn sampled according to X^b. (In the case when $|y|$ is not a multiple of b, z_r is shorter; this does not affect the analysis. We call the strings z_i the *blocks* of y.)

▶ **Lemma 14** (Blocking Lemma). *[6, Lemma 3.3] Let $\{T_x\}$ be an ensemble of sets of strings such that all strings in T_x have the same length $\mathrm{poly}(|x|)$. Suppose that for each $x \in \{0,1\}^*$ and for each $b \in \mathbb{N}$ there is an integer Λ_b and a random variable $R_{x,b}$ whose image contains $(T_x)^b$, and such that $R_{x,b}$ is computable by a circuit of size $\mathrm{poly}(|x|, b)$ and has a Λ_b-encoding of length $s'(x,b)$ decodable by a circuit of size $\mathrm{poly}(|x|, b)$. Then there are constants c_1 and c_2 so that, for every constant $\alpha > 0$, every $t \in \mathbb{N}$, every sufficiently large x, and every $\lceil t^\alpha \rceil$-suitable $y \in (T_x)^t$,*

$$\mathrm{KT}(y) \le t^{1-\alpha} \cdot s'(x, \lceil t^\alpha \rceil) + t^{\alpha \cdot c_1} \cdot |x|^{c_2}.$$

Here, we say that $y \in (T_x)^t$ is b-suitable if each block of y (of length bn) is Λ_b-heavy.

With the Encoding and Blocking Lemmas in hand, we can now show how to give upper and lower bounds on the KT complexity of concatenated samples from a distribution. The following lemma gives the upper bound.

▶ **Lemma 15.** *Suppose X is a distribution sampled by a circuit $C_x : \{0,1\}^m \to \{0,1\}^n$ of size polynomial in $|x|$. For every polynomial $w = w(|x|)$ with $|x| \le w$, there exist constants c_0, c_2, and α_0 such that for every sufficiently large polynomial t and for all large x, if y is the concatenation of t samples from X, then*

$$\mathrm{KT}(y) \le tH(X) + wm(t^{1-\alpha_0/2}) + t^{1-\alpha_0}|x|^{c_0+c_2}$$

Proof. Pick c_0 so that $|x|^{c_0} > m + wm + |x|$, and observe that for all large x we have $|x|^{c_0} > H(X) + wm + O(\log(|x|))$. Let $t = t(|x|)$ be any polynomial. Let $b \in \mathbb{N}$ with $b < t$, and let $\Lambda_b = bH(X) + wm\sqrt{b}$. Then, by the Encoding Lemma $X^b = \otimes^b X$ has a Λ_b-encoding of length $\Lambda_b + \log(\Lambda_b) + O(1)$ that is decodable by circuits of size $\mathrm{poly}(b|x|)$. Let $r = \lceil t/b \rceil$. Recall that $y = y_1 \ldots y_t$ where each y_i is a string of length n sampled according to the distribution X. Equivalently, we can consider y to be equal to $z_1 \ldots z_r$ where each z_i is a string of length bn sampled according to X^b; the strings z_i are the blocks of y. By the Flattening Lemma, the probability that any given z_b is not Λ_b-heavy is at most 2^{-w^2+1}. Thus, by the union bound, the probability that y is not b-suitable (i.e., the probability that there is at least one block that is not Λ_b-heavy) is at most $r \cdot 2^{-w^2+1} < t \cdot 2^{-w^2}$. Since $w \ge |x|$ and t is polynomial in $|x|$, it follows that for all large x, with probability at least $(1 - 1/2^{2|x|})$, each of the r blocks is Λ_b-heavy and hence, by the Encoding Lemma, each block has an encoding of length $s'(n,b) = \Lambda_b + \log(\Lambda_b) + O(1)$. Thus, by the Blocking Lemma, for certain constants c_1 and c_2 (which do not depend on t), for any constant $\alpha > 0$ (for all large enough y),

$$\begin{aligned}
\mathrm{KT}(y) &\le t^{1-\alpha} \cdot s'(x, \lceil t^\alpha \rceil) + t^{\alpha \cdot c_1} \cdot |x|^{c_2} \\
&= t^{1-\alpha} \cdot (\Lambda_{\lceil t^\alpha \rceil} + \log(\Lambda_{\lceil t^\alpha \rceil}) + O(1)) + t^{\alpha \cdot c_1} \cdot |x|^{c_2} \\
&= t^{1-\alpha} \cdot (\lceil t^\alpha \rceil H(X) + wm\sqrt{\lceil t^\alpha \rceil} + \log(\Lambda_{\lceil t^\alpha \rceil}) + O(1)) + t^{\alpha \cdot c_1} \cdot |x|^{c_2} \\
&\le t^{1-\alpha} \cdot (t^\alpha H(X) + |x|^{c_0} + wm\sqrt{t^\alpha}) + t^{\alpha \cdot c_1} \cdot |x|^{c_2}
\end{aligned}$$

Recall that the inequality above holds for *all* $\alpha > 0$. If we now pick $\alpha_0 \leq 1/(1+c_1)$, we obtain the claimed inequality

$$\mathrm{KT}(y) \leq tH(x) + wmt^{1-\alpha_0/2} + t^{1-\alpha_0}(|x|^{c_0+c_2}).$$

◀

We now turn to a lower bound on $\mathsf{KT}(y)$.

▶ **Lemma 16.** *Let* $\mathrm{poly}(|x|)$ *denote some fixed polynomial in* $|x|$*, and let* α_0 *be such that* $0 < \alpha_0 < 1/2$*. For all large* x*, if* X *is a distribution sampled by a circuit* $C_x : \{0,1\}^m \to \{0,1\}^n$ *of polynomial size, then it holds that for every* w *and every* $t > w^4$*, if* y *is sampled from* X^t*, then with probability at least* $1 - 2^{-w^2}$*,*

$$\mathrm{KT}(y) \geq tH(X) - wm\sqrt{t} - t^{1-\alpha_0}\mathrm{poly}(|x|)$$

Proof. Consider the distribution $X^t = \otimes^t X$ and sample y from it. Recall that $H(X^t) = tH(x)$. By the Flattening Lemma, X^t is $\sqrt{t} \cdot m$-flat. Therefore, the probability that y is $wm\sqrt{t}$-typical is at least $1 - 2^{-w^2+1}$. We would like to bound the probability that $\mathrm{KT}(y) < tH(X) - wm\sqrt{t} - t^{1-\alpha_0} \cdot \mathrm{poly}(|x|)$. To bound this probability, note that $\Pr[\mathrm{KT}(y) < k]$ is equal to

$$\begin{aligned}
&\Pr[\mathrm{KT}(y) < k \wedge y \text{ is typical}] + \Pr[\mathrm{KT}(y) < k \wedge y \text{ is atypical}] \\
&\leq \Pr[\mathrm{KT}(y) < k \wedge y \text{ is typical}] + \Pr[y \text{ is atypical}]
\end{aligned}$$

where we are interested in $k = tH(x) - wm\sqrt{t} - t^{1-\alpha_0} \cdot \mathrm{poly}(|x|)$ and "y is typical" means "y is $wm\sqrt{t}$-typical." We have already observed above that the second term is bounded by 2^{-w^2+1}. For the first term, we have

$$\begin{aligned}
\Pr[\mathrm{KT}(y) < k \wedge y \text{ is typical}] &= \sum_{\{y:\mathrm{KT}(y)<k \wedge y \text{ is typical}\}} \Pr(y) \\
&\leq \sum_{\{y:\mathrm{KT}(y)<k \wedge y \text{ is typical}\}} 2^{wm\sqrt{t}} \cdot 2^{-H(X^t)} \\
&\leq 2^k \cdot 2^{wm\sqrt{t}} \cdot 2^{-H(X^t)} \\
&= 2^{tH(x) - wm\sqrt{t} - t^{1-\alpha_0} \cdot \mathrm{poly}(|x|)} \cdot 2^{wm\sqrt{t}} \cdot 2^{-tH(X)} \\
&= 2^{-t^{1-\alpha_0} \cdot \mathrm{poly}(|x|)}
\end{aligned}$$

where the first inequality follows from the definition of typicality, and the second inequality follows since there are only $\sum_{i=0}^{k-1} 2^i < 2^k$ descriptions of strings with complexity less than k.

Summarizing, we conclude that the probability that $\mathrm{KT}(y) < tH(x) - wm\sqrt{t} - t^{1-\alpha_0} \cdot \mathrm{poly}(|x|)$ is at most

$$2^{-t^{1-\alpha_0} \cdot \mathrm{poly}(|x|)} + 2^{-w^2+1}.$$

To show that the above probability is less than $1/2^{w^2}$ is equivalent to showing that

$$2^{-t^{1-\alpha_0} \cdot \mathrm{poly}(|x|)} < 2^{-w^2+1}.$$

Thus we must show that $w^2 - 1 < t^{1-\alpha_0} \cdot \text{poly}(|x|)$. This holds, since

$$\begin{aligned} w^2 - 1 &< w^2 \\ &< (t^{1/4})^2 \\ &= \sqrt{t} \\ &\leq t^{1-\alpha_0} \\ &\leq t^{1-\alpha_0} \cdot \text{poly}(|x|). \end{aligned}$$

◀

3.3 Reducing co-NISZK to MKTP

▶ **Theorem 17.** MKTP *is hard for* co-NISZK *under* P/poly *many-one reductions.*

Proof. We prove the claim by reduction from the NISZK-complete problem EA. Let $x = (C_x, k)$ be an arbitrary instance of Promise-EA, where $C_x : \{0,1\}^m \to \{0,1\}^n$ is a circuit that represents distribution X. Let $w = 2|x|$, and let α_0, c_0, and c_2 be the constants from Lemma 15. Let $\lambda = wmt^{1-\alpha_0/2}$. Pick the polynomial t so that $t(|x|) > 2(\lambda + t^{1-\alpha_0}|x|^{c_0+c_2})$ and $w^4 < t$ (and note that all large polynomials have this property). Construct y as t samples from X. Let $\theta = tk + \lambda + t^{1-\alpha_0}|x|^{c_0+c_2}$. We claim that, with probability at least $1 - \frac{1}{2^{2|x|}}$, if $(X,k) \in \mathsf{EA}_{YES}$, then $(y,\theta) \in \mathsf{MKTP}_{NO}$ and if $(X,k) \in \mathsf{EA}_{NO}$, then $(y,\theta) \in \mathsf{MKTP}_{YES}$.

If $(X,k) \in \mathsf{EA}_{NO}$, then $H(X) < k$. Then by Lemma 15, we have that, with high probability,

$$\begin{aligned} \mathrm{KT}(y) &\leq tH(X) + \lambda + t^{1-\alpha_0}|x|^{c_0+c_2} \\ &< tk + \lambda + t^{1-\alpha_0}|x|^{c_0+c_2} \\ &= \theta \end{aligned}$$

thus $\mathrm{KT}(y) \leq \theta$, and thus $(y,\theta) \in \mathsf{MKTP}_{YES}$.

If $(X,k) \in \mathsf{EA}_{YES}$, then $H(X) > k+1$. Then by Lemma 16, with probability at least $1 - 2^{-w^2} > 1 - 2^{2|x|}$, we have that

$$\begin{aligned} \mathrm{KT}(y) &\geq tH(X) - wm\sqrt{t} - t^{1-\alpha_0}|x|^{c_0+c_2}, \\ &> tH(X) - \lambda - t^{1-\alpha_0}|x|^{c_0+c_2} && \text{(since } \alpha_0 < 1/2) \\ &> t(k+1) - \lambda - t^{1-\alpha_0}|x|^{c_0+c_2} \\ &> tk + \lambda + t^{1-\alpha_0}|x|^{c_0+c_2} && \text{(since } t > 2(\lambda + t^{1-\alpha_0}|x|^{c_0+c_2})) \\ &= \theta \end{aligned}$$

thus $\mathrm{KT}(y) > \theta$, and thus $(y,\theta) \in \mathsf{MKTP}_{NO}$.

We have shown that there is a polynomial-time-computable function f, such that, if $x \in \mathsf{EA}_{YES}$, then with high probability (for random r) $f(x,r) = (y,\theta)$ is in MKTP_{NO}, and if $x \in \mathsf{EA}_{NO}$, then with high probability $f(x,r) = (y,\theta)$ is in MKTP_{YES}. By a standard counting argument (similar to the proof that BPP ⊆ P/poly), since the probability of success for either bound is greater than $(1 - 1/2^{2n})$, we can fix a sequence of random bits to hardwire in to this reduction and obtain a family of circuits computing a $\leq_m^{\mathsf{P/poly}}$ reduction from any problem in NISZK to $\overline{\mathsf{MKTP}}$. ◀

▶ **Corollary 18.** MKTP *is hard for* NISZK *under* BPP *reductions that make at most one query along any path.*

Proof. This follows from the proof of Theorem 17. Namely, on input $x = (C_x, k)$, construct the string y consisting of t random samples from C_x and query the oracle on (y, θ). On Yes-instances, y will have KT complexity greater than θ (with high probability), and on No-instances, y will have KT complexity less than θ (with high probability). ◀

▶ **Corollary 19.** MKTP *is hard for* SZK *under non-adaptive* BPP*-Turing reductions.*

Proof. Recall from [15] that SZK reduces to Promise-EA via non-adaptive (deterministic) reductions. The result is now immediate, from Corollary 18. ◀

4 A Complete Problem for $\mathsf{NISZK_L}$

Having established a hardness result for MKTP under $\leq_m^{\mathsf{P/poly}}$ reductions, we now establish an analogous hardness result under the much more restrictive $\leq_m^{\mathsf{proj}}$ reductions. For this, we first need to present a complete problem for $\mathsf{NISZK_L}$.

Recall that the NISZK-complete problem EA deals with the question of approximating the entropy of a distribution represented by a circuit. In order to talk about $\mathsf{NISZK_L}$, we shall need to consider probability distributions that are represented using restricted class of circuits. In particular, we shall focus on a problem that we denote $\mathsf{EA_{NC^0}}$.

▶ **Definition 20** (Promise-$\mathsf{EA_{NC^0}}$)**.** *Promise-*$\mathsf{EA_{NC^0}}$ *is the promise problem obtained from Promise-*EA*, by considering only instances* (C, k) *such that* C *is a circuit of fan-in two gates, where no output gate depends on more than* four *input gates.*

It is not surprising that $\mathsf{EA_{NC^0}}$ is complete for $\mathsf{NISZK_L}$. The completeness proof that we present owes much to the proof presented by Dvir et al. [13] (showing that an $\mathsf{NC^0}$-variant of the SZK-complete problem EntropyDifference is complete for $\mathsf{SZK_L}$) and to the proof presented by Goldreich et al. [15] showing that EA is complete for NISZK. We will need to make use of various detailed aspects of the constructions presented in this prior work, and thus we will present the full details here.

First, we show membership in $\mathsf{NISZK_L}$.

4.1 Membership in $\mathsf{NISZK_L}$

▶ **Theorem 21.** *Promise-*$\mathsf{EA_{NC^0}} \in \mathsf{NISZK_L}$

Proof. In order to show membership, we must show the existence of a non-interactive proof system where the verifier and simulator are both in logspace. To do this, we adapt the protocol that is used in [15] to show that EA is in NISZK. Their protocol works by first transforming an instance (C, k) of EA, of length s, (where C represents a distribution X) into a representation of a distribution Z on ℓ bits. The transformation consists of four steps:

1. Take poly(s) samples from X and concatenate them. Call this distribution X' and let $C_{X'}$ be the circuit representing X'.
2. Hash the output of X' with a hash function h chosen at random from a 2-universal family of hash functions. (The parameters of the hash function depend on the value k of the EA instance.) Let this distribution be Y, represented by C_Y.
3. Take poly(s) copies of Y and concatenate their output. Call this distribution Y', represented by $C_{Y'}$.
4. Hash a sample of Y' with a hash function h' chosen at random from a 2-universal family of hash functions. Let this distribution be Z, represented by C_Z.

Section 2 and Appendix C of [15] give a careful proof of the fact that, with Z defined as above from the EA instance (C, k), a NISZK protocol for EA is given by:

1. With reference string σ, the prover selects a string r uniformly at random from the set $\{r' : Z(r') = \sigma\}$.
2. The verifier accepts if $C_Z(r) = \sigma$ and rejects otherwise.

They also show that the following simulator satisfies the required zero-knowledge properties:

1. Select an input r to Z uniformly at random and let $\sigma = C_Z(r)$.
2. return (σ, r).

It suffices for us to show that, if (C, k) is an instance of $\mathsf{EA}_{\mathsf{NC}^0}$, then the tasks of the verifier and the simulator in the protocol above can be implemented in logspace.

First, we observe that, given (C, k), a branching program P_Z realizing the distribution Z can be constructed in logspace. Indeed, it is trivial to construct a branching program P_X that realizes X (since each output bit of the NC^0 circuit Z has an easy-to-compute branching program of constant size). Then a branching program $P_{X'}$ realizing X' consists of several copies of P_X concatenated together (where each copy uses independent random input bits). The hash functions h considered in [15] are represented by Boolean matrices M_h, where computing $h(w)$ is simply multiplying M_h by the vector w. Since Boolean matrix multiplication is easy to compute in $\mathsf{NC}^1 \subseteq \mathsf{L}$, and since the composition of two logspace-computable functions is also logspace-computable, it is easy to build a branching program P_Y representing the distribution Y (That is, given a branching program for computing $M_h \cdot w$, for any node v that queries a bit of w, replace the pair of edges leaving v by a branching program that computes that bit of w (as a sample from X').) In the same way, branching programs for Y' and Z are easy to construct, given P_Y.

Hence a logspace verifier, with access to r, σ, and an $\mathsf{EA}_{\mathsf{NC}^0}$ instance (C, k), can construct the branching program P_Z and compute $P_Z(r)$ and check if the output is equal to σ. It is equally easy to see that the simulator can be implemented in logspace. This establishes membership in $\mathsf{NISZK}_\mathsf{L}$. ◀

The following corollary is a direct analog to [15, Proposition 1].

▶ **Corollary 22.** *If* Π *is any promise problem that is* $\leq^{\mathsf{L}}_{\mathrm{m}}$ *reducible to* $\mathsf{EA}_{\mathsf{NC}^0}$*, then* $\Pi \in \mathsf{NISZK}_\mathsf{L}$.

We close this section by presenting an example of a well-studied natural problem in $\mathsf{NISZK}_\mathsf{L}$. (A graph is said to be *rigid* if it has no nontrivial automorphism.)

▶ **Corollary 23.** *The Non-Isomorphism Problem for Rigid Graphs lies in* $\mathsf{NISZK}_\mathsf{L}$

Proof. First note that the proof of Theorem 21 carries over to show that a problem that we may call $\mathsf{EA}_{\mathsf{BP}}$ (defined just as $\mathsf{EA}_{\mathsf{NC}^0}$ but where the distribution is represented as a branching program instead of as an NC^0 circuit) also lies in $\mathsf{NISZK}_\mathsf{L}$. Now observe that the reduction given in Section 3.1 of [6] shows how to take as input two rigid graphs on n vertices (G_0, G_1) and build a branching program that takes as input a bitstring w of length t and t permutations $\pi_1, \ldots, \pi_t$ and output the sequence of t permuted graphs $\pi_i(G_{w_i})$. It is observed in [6] that this distribution has entropy $t(1+\log n!)$ if the graphs are non-isomorphic, and has entropy at most $t \log n!$ otherwise. ◀

4.2 Hardness for $\mathsf{NISZK_L}$

In order to re-use the tools developed in [15], we will follow the structure of the proof given there, showing that EA is hard for NISZK. Namely, we introduce the problem SDU (Statistical Distance from Uniform) and its NC^0 variant, and prove hardness for $\mathsf{SDU}_{\mathsf{NC}^0}$.

▶ **Definition 24** (SDU and $\mathsf{SDU}_{\mathsf{NC}^0}$). *Consider Boolean circuits $C_X : \{0,1\}^m \to \{0,1\}^n$ representing distributions X. The promise problem*

$$\mathsf{SDU} = (\mathsf{SDU}_{YES}, \mathsf{SDU}_{NO})$$

is given by

$$\mathsf{SDU}_{YES} \stackrel{def}{=} \{C_X : \Delta(X, U_n) < 1/n\}$$
$$\mathsf{SDU}_{NO} \stackrel{def}{=} \{C_X : \Delta(X, U_n) > 1 - 1/n\}$$

where $\Delta(X,Y) = \Sigma_\alpha |\Pr[X = \alpha] - \Pr[Y = \alpha]|/2$.

$\mathsf{SDU}_{\mathsf{NC}^0}$ *is the analogous problem, where the distributions X are represented by* NC^0 *circuits where no output bit depends on more than* four *input bits.*

It is shown in [15, Lemma 4.1] that C_X is in SDU if and only if $(C_X, n-3)$ is in EA. This yields the following corollary:

▶ **Corollary 25.** $\mathsf{SDU}_{\mathsf{NC}^0} \le_{\mathrm{m}}^{\mathsf{proj}} \mathsf{EA}_{\mathsf{NC}^0}$.

Proof. This is trivial if we assume an encoding of $\mathsf{SDU}_{\mathsf{NC}^0}$ instances, such that the NC^0 circuits $C_X : \{0,1\}^m \mapsto \{0,1\}^n$ are encoded by strings of length given by the standard pairing function $\frac{m^2+3m+2mn+n+n^2}{2}$, so that the length of an instance of $\mathsf{SDU}_{\mathsf{NC}^0}$ completely determines n. (An NC^0 circuit with m inputs and n outputs has a description of size $O(n \log m)$, and thus it is easy to devise a padded encoding of this much larger length.) Thus, in the projection circuit computing the reduction $C_X \mapsto (C_X, n-3)$, the output bits encoding $n-3$ are hardwired to constants, and the input bits encoding C_X are copied directly to the output. ◀

▶ **Theorem 26.** *Promise-*$\mathsf{EA}_{\mathsf{NC}^0}$ *and Promise-*$\mathsf{SDU}_{\mathsf{NC}^0}$ *are hard for* $\mathsf{NISZK_L}$ *under* $\le_{\mathrm{m}}^{\mathsf{proj}}$ *reductions.*

Proof. By Corollary 25, it suffices to show hardness for $\mathsf{SDU}_{\mathsf{NC}^0}$. In order to establish hardness, we need to develop the machinery of *perfect randomized encodings*, which were developed by Applebaum et al. [10] and then were applied in the setting of NISZK by Dvir et al. [13].

4.2.1 Perfect Randomized Encodings

▶ **Definition 27.** *Let $f : \{0,1\}^n \to \{0,1\}^\ell$ be a function. We say that $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ is a perfect randomized encoding of f with blowup b if it is:*

- ***Input independent:*** *for every $x, x' \in \{0,1\}^n$ such that $f(x) = f(x')$, the random variables $\hat{f}(x, U_m)$ and $\hat{f}(x', U_m)$ are identically distributed.*
- ***Output Disjoint:*** *for every $x, x' \in \{0,1\}^n$ such that $f(x) \neq f(x')$, $Supp(\hat{f}(x, U_m)) \cap Supp(\hat{f}(x', U_m)) = \emptyset$.*
- ***Uniform:*** *for every $x \in \{0,1\}^n$ the random variable $\hat{f}(x, U_m)$ is uniform over $Supp(\hat{f}(x, U_m))$.*

- ***Balanced:*** *for every* $x, x' \in \{0,1\}^n$ $|Supp(\hat{f}(x, U_m))| = |Supp(\hat{f}(x', U_m))| = b$

The following property of perfect randomized encodings is established in [13].

▶ **Lemma 28** (entropy)**.** *Let* $f : \{0,1\}^n \to \{0,1\}^\ell$ *be a function and let* $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ *be a perfect randomized encoding of* f *with blowup* b*. Then* $H(\hat{f}(U_n, U_m)) = H(f(U_n)) + \log b$

The following two properties are given in Applebaum et al. [10].

▶ **Lemma 29** (concatenation)**.** *For* $i = 1, \ldots, \ell$ *let* $f_i : \{0,1\}^n \to \{0,1\}$ *be the Boolean function computing the* $i-th$ *bit of* $f : \{0,1\}^n \to \{0,1\}^\ell$*. If* $\hat{f}_i : \{0,1\}^n \times \{0,1\}^{m_i} \to \{0,1\}^{s_i}$ *is a perfect randomized encoding of* f_i*, then the function* $\hat{f} : \{0,1\}^n \times \{0,1\}^{m_1 + \ldots, m_\ell} \to \{0,1\}^{s_1 + \ldots + s_\ell}$ *defined by* $\hat{f}(x, (r_1, \ldots, r_\ell)) \stackrel{def}{=} (\hat{f}_1(x, r_1), \ldots, \hat{f}_\ell(x, r_\ell))$ *is a perfect randomized encoding of* f*.*

▶ **Lemma 30** (composition)**.** *Let* $g(x, r_g)$ *be a perfect randomized encoding of* $f(x)$ *and let* $h((x, r_g), r_h)$ *be a perfect randomized encoding of* $g(x, r_g)$ *(viewed as a single argument function). Then, the function* $\hat{f}((x, r_g), r_h) \stackrel{def}{=} h((x, r_g), r_h)$ *is a perfect randomized encoding of* f*.*

The following claim is not formally stated in [10] but can be found in their discussion of perfect randomized encodings in section 4.1 of that paper.

▷ Claim 31. Let $f : \{0,1\}^n \to \{0,1\}^\ell$ be a function. If $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ is a perfect randomized encoding of f, then $\hat{f}$ has blowup 2^m.

The following is apparent from Lemma 29, Lemma 30, and Claim 31.

▷ Claim 32. The blowup of a perfect randomized encoding $\hat{f}$ created by composing or concatenating perfect randomized encodings $\hat{f}_1, \ldots, \hat{f}_\ell$ is $\prod_{i=1}^{\ell} b_i$.

4.2.2 Constructing an NC^0 perfect randomized encoding

The first step in showing completeness of $\mathsf{EA}_{\mathsf{NC}^0}$ is to use the following construction of perfect randomized encodings of functions computed by branching programs, from [10].

▶ **Definition 33.** *Let* Q *be a branching program of size* ℓ *computing a Boolean function* $f : \{0,1\}^n \to \{0,1\}$*. Fix some topological ordering of the vertices of* Q *where the source vertex is labelled 1 and the terminal vertex is labelled* ℓ*. Let* $A(x)$ *be the* $\ell \times \ell$ *adjacency matrix of* G_x *where entry* (i,j) *is a degree-1 polynomial* $q_{i,j} \in \{x_k, (1 - x_k), 1, 0\}$*, such that the transition from node* i *to node* j *queries variable* x_k *and proceeds if* $q_{i,j}(x_k) = 1$*. Define* $L(x)$ *as the submatrix of* $A(x) - I$ *obtained by deleting the first column and last row.*

$$\begin{pmatrix} * & * & * & * & * \\ -1 & * & * & * & * \\ 0 & -1 & * & * & * \\ 0 & 0 & -1 & * & * \\ 0 & 0 & 0 & -1 & * \end{pmatrix}$$

Let $r^{(1)}$*, and* $r^{(2)}$ *be vectors over GF(2) of length* $\binom{\ell-1}{2}$ *and* $\ell - 2$ *respectively. Let* $R_1(r^{(1)})$ *be an* $\ell \times \ell$ *matrix with 1's on the main diagonal, 0's below it and the elements of* $r^{(1)}$ *in the remaining* $\binom{\ell-1}{2}$ *entries above the main diagonal. Let* $R_2(r^{(2)})$ *be an* $\ell \times \ell$ *matrix with 1's on the main diagonal, 0's below it, and the elements of* $r^{(2)}$ *in the last column.*

$$\begin{pmatrix} 1 & r_1^{(1)} & r_2^{(1)} & \cdot & r_{\ell-1}^{(1)} \\ 0 & 1 & \cdot & \cdot & \cdot \\ 0 & 0 & 1 & \cdot & \cdot \\ 0 & 0 & 0 & 1 & r^{(1)}_{\binom{\ell-1}{2}} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & r_1^{(2)} \\ 0 & 1 & 0 & 0 & \cdot \\ 0 & 0 & 1 & 0 & \cdot \\ 0 & 0 & 0 & 1 & r_{\ell-2}^{(2)} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The following lemma appears as [10, Lemma 4.15].

▶ **Lemma 34.** *Let Q be a branching program of size ℓ computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$. Let the function $\hat{f}(x,(r^{(1)},r^{(2)}))$ produce as output the $\binom{\ell}{2}$ entries on or above the main diagonal of the matrix*

$$R_1(r^{(1)})L(x)R_2(r^{(2)}).$$

Then $\hat{f}$ is a perfect randomized encoding of f.

▶ **Lemma 35.** *There is a function computable in AC^0 (in fact, it can be a projection) that takes as input a branching program Q of size ℓ computing a function $f : \{0,1\}^n \to \{0,1\}$, and produces as output a list $(q_1, \ldots, q_{\binom{\ell}{2}})$ of degree-three polynomials over GF(2), where $q_i(x,(r^{(1)},r^{(2)}))$ produces the i-th output bit of $\hat{f}(x,(r^{(1)},r^{(2)}))$. The blowup of the encoding $\hat{f}$ is $2^{\binom{\ell}{2}-1}$.*

Proof. Claim 31 establishes the claim regarding blowup. Constructing the three matrices $L(x), R_1$ and R_2 can clearly be done in AC^0. Their product cannot be computed in AC^0 (since this involves computing PARITY), but it is easy to compute an *encoding* of the expression for entry (i,m) of the product, which is given by the degree-three polynomial $\sum_{j,k} R_{1\,(i,k)} L_{(k,j)} R_{2\,(j,m)}$. To see that this can be a projection, note that the entries of the matrices R_1 and R_2 are entirely determined by the *size* ℓ (and thus they depend only on the length of the encoding of the branching program). The entries of $L(x)$ are essentially the entries of the adjacency matrix encoding the branching program Q, and thus they can be copied directly via a projection. Then, given the encodings of the matrices, the encodings of the terms of each polynomial q_i are simply copied from the encodings of the matrices, and thus this can be done via a projection also. ◀

Note that each polynomial q_i in the statement of the preceding lemma is most conveniently expressed as a sum of terms. We now show how to replace each q_i with NC^0 circuitry, using the following lemma from [10, Lemma 4.17].

▶ **Lemma 36.** *Let $f(x) = T_1(x) + \ldots + T_k(x)$ where $f, T_1, \ldots, T_k : GF(2)^n \to GF(2)$, and summation is over GF(2), and each term T_i has degree at most 3. Let the local encoding $\hat{f} : GF(2)^{n+(2k-1)} \to GF(2)^{2k}$ be such that $\hat{f}(x,(r_1,\ldots,r_k,r'_1,\ldots,r'_{k-1}))$ is equal to*

$$(T_1(x) - r_1, T_2(x) - r_2, \ldots, T_k(x) - r_k,$$
$$r_1 - r'_1, r'_1 + r_2 - r'_2, \ldots, r'_{k-2} + r_{k-1} - r'_{k-1}, r'_{k-1} + r_k)$$

Then $\hat{f}$ is a perfect randomized encoding of f where each bit of the output depends on at most 4 bits of $(x,(r_1,\ldots,r_k,r'_1,\ldots,r'_{k-1}))$.

▶ **Lemma 37.** *There is a function computable in AC^0 (in fact, it can be a projection) that takes as input a branching program Q of size ℓ computing a function $f : \{0,1\}^n \to \{0,1\}$, and produces as output a list p_i of NC^0 circuits, where p_i computes the i-th bit of a function $\hat{f}$ that is a perfect randomized encoding of f that has blowup $2^{(\binom{\ell}{2}-1)(2(\ell-1)^2-1)}$. Each p_i*

depends on at most four input bits from (x, r) (where r is the sequence of random bits in the randomized encoding).

Proof. This follows immediately by applying the construction of Lemma 36 to the degree-three polynomials for each entry in the product matrix given by AC^0-computable function given by Lemma 35. Each of those polynomials has $(\ell-1)^2$ terms, and it is apparent from Lemma 36 that each such entry gives rise to $2(\ell-1)^2-1$ new random bits in the randomized encoding. The assertion regarding blowup now follows from Claim 31. The assertions regarding the bits upon which each p_i depends follows from inspection. The construction of Lemma 36 can clearly be accomplished via a projection, and composing that projection with the projection from Lemma 35 again yields a projection. ◀

4.2.3 $\mathsf{SDU}_{\mathsf{NC}^0}$ is Complete for $\mathsf{NISZK_L}$

We now have all of the tools required to complete the proof of Theorem 26.

Let $\prod$ be an arbitrary promise problem in $\mathsf{NISZK_L}$ with proof system (P, V) and simulator S and let x be an instance of $\prod$. Let $M_x(s)$ denote a routine that simulates $S(x)$ with randomness s to obtain a transcript (σ, p); if $V(x, \sigma, p)$ accepts, then $M_x(s)$ outputs σ, otherwise it outputs $0^{|\sigma|}$. (We can assume without loss of generality that $|\sigma| = |x|^k$.) It is shown in [15, Lemma 4.2] that the map $x \mapsto M_x$ is a reduction of Π to SDU:

▷ Claim 38. If $x \in \prod_{YES}$, then $\Delta(M_x, U_{|x|^k}) < 1/|x|^k$, and $x \in \prod_{NO}$ implies $\Delta(M_x, U_{|x|^k})) > 1 - 1/|x|^k$.

The proof of the preceding claim in [15, Lemma 4.2] actually shows a stronger result. It shows that, if the statistical difference between the output distribution of the simulator and the distribution of true transcripts is at most $1/e(n)$, then the statistical difference of M_x and the uniform distribution is at most $1/e(n) + 2^{-n}$ on inputs of length n. Thus, using Definition 1 (which is equivalent to the definition of NISZK given in [15]), the simulator produces a distribution that differs from the uniform distribution by only $1/n^{\omega(1)}$. Thus we have the following claim:

▷ Claim 39. Let $c \in \mathbb{N}$. Then for all large x, If $x \in \prod_{YES}$, then $\Delta(M_x, U_{|x|^k}) < 1/|x|^{kc}$, and $x \in \prod_{NO}$ implies $\Delta(M_x, U_{|x|^k})) > 1 - 1/|x|^{kc}$.

Furthermore, it is also shown in [15, Lemma 3.1] that EA has a NISZK protocol in which the completeness error, soundness error, and simulator deviation are all at most 2^{-m} on inputs of length m. Furthermore, that proof carries over to show that $\mathsf{EA_{BP}} \in \mathsf{NISZK_L}$ with these same parameters. Thus we obtain the following fact, which we will use later in Section 6.

▷ Claim 40. Let $c \in \mathbb{N}$. Then for all large x, If x is a Yes-instance of $\mathsf{EA_{BP}}$, then $\Delta(M_x, U_{|x|^k}) < 1/2^{|x|-1}$, and if x is a No-instance of $\mathsf{EA_{BP}}$, then $\Delta(M_x, U_{|x|^k})) > 1-1/2^{|x|-1}$.

Since S runs in logspace, each bit of $M_x(s)$ can be simulated with a branching program Q_x. Furthermore, it is straightforward to see that there is an AC^0-computable function that takes x as input and produces an encoding of Q_x as output, and it can even be seen that this function can be a *projection*. (To see this, note that in the standard construction of a Turing machine from a logspace-bounded Turing machine S (with input (x, s)) each node of the branching program has a name that encodes a configuration of the machine, a time step, and the position of the input head. This branching program can be constructed in AC^0, given only the *length* of x. In order to construct Q_x, it suffices merely to hardwire in the transitions from any node that is "scanning" some bit position x_i. That is, if the transition out of node v goes to node v_0 if $x_i = 0$ and to node v_1 if $x_i = 1$, then in the adjacency matrix for Q_x, entry $(v, v_1) = x_i$ and entry (v, v_0) is $\neg x_i$. This is clearly a projection.)

Now apply the projection of Lemma 37 to (each output bit of) the branching program Q_x of size ℓ, to obtain an NC^0 circuit C_x computing a perfect randomized encoding with blowup $b = 2^{|x|^k(\binom{\ell}{2}-1)(2(\ell-1)^2-1)}$. ($C_x$ has $\log b + |x|^k$ output bits.)

Now consider $|H(C_x) - H(U_{\log b+|x|^k})|$. By Lemma 28 this is equal to $|H(Q_x) + \log b - H(U_{\log b+|x|^k})|$. Since $H(Q_x) = H(M_x)$ and $H(U_{\log b+|x|^k}) = \log b + H(U_{|x|^k})$, we have that $|H(C_x) - H(U_{\log b+|x|^k})| = |H(M_x) - H(U_{|x^k|})|$. The proof of Theorem 26 is now complete, by appeal to Claim 39. ◀

5 Hardness of MKTP under Projections

▶ **Theorem 41.** MKTP *is hard for* co-NISZK$_\mathsf{L}$ *under nonuniform* $\leq_\mathrm{m}^{\mathsf{AC}^0}$ *reductions.*

Proof. We build on the proof of Theorem 17, and present a reduction from the NISZK$_\mathsf{L}$-complete problem $\mathsf{EA}_{\mathsf{NC}^0}$. Let $x = (C_x, k)$ be an arbitrary instance of Promise-$\mathsf{EA}_{\mathsf{NC}^0}$, where $C_x : \{0,1\}^m \to \{0,1\}^n$ is an NC^0 circuit that represents distribution X. Let $|x| < w < \sqrt[4]{t}$, and let α_0, c_0, and c_2 be the constants from Lemma 15. Let $\lambda = wmt^{1-\alpha_0/2}$ and construct y as t samples from X. Let $\theta = tk + \lambda + t^{1-\alpha_0}|x|^{c_0+c_2}$.

As in the proof of Theorem 17, we have that, with probability at least $1 - \frac{1}{2^{2|x|}}$, if (X, k) is a Yes-instance of $\mathsf{EA}_{\mathsf{NC}^0}$, then $(y, \theta) \in \mathsf{MKTP}_{NO}$ and if (X, k) is a No-instance of $\mathsf{EA}_{\mathsf{NC}^0}$, then $(y, \theta) \in \mathsf{MKTP}_{YES}$.

Thus we have shown that there is a uniform AC^0-computable function f, such that, if $x \in \mathsf{EA}_{YES}$, then with high probability (for random r) $f(x, r) = (y, \theta)$ is in MKTP_{NO}, and if $x \in \mathsf{EA}_{NO}$, then with high probability $f(x, r) = (y, \theta)$ is in MKTP_{YES}. (Namely, the AC^0 function takes $x = (C_x, k)$ and r as input, computes θ from k and $|x|$, and computes y by feeding t segments of r into the NC^0 circuit C_x.)

As in the proof of Theorem 17, we can fix a sequence of random bits to hardwire in to this reduction and obtain a (nonuniform) $\leq_\mathrm{m}^{\mathsf{AC}^0}$ reduction from $\mathsf{EA}_{\mathsf{NC}^0}$ to $\overline{\mathsf{MKTP}}$.

◀

An immediate corollary (making use of the "Gap Theorem" of [1]) is that MKTP is hard for co-NISZK$_\mathsf{L}$ under $\leq_\mathrm{m}^{\mathsf{NC}^0}$ reductions. Below, we improve this, showing hardness under projections.

▶ **Corollary 42.** MKTP *is hard for* co-NISZK$_\mathsf{L}$ *under nonuniform* $\leq_\mathrm{m}^{\mathsf{NC}^0}$ *reductions.*

Proof. Corollary 22, combined with the NISZK$_\mathsf{L}$-completeness of $\mathsf{EA}_{\mathsf{NC}^0}$, implies that co-NISZK$_\mathsf{L}$ is closed under $\leq_\mathrm{m}^{\mathsf{L}}$ reductions. It is shown in the "Gap Theorem" of [1] that, for any class $\mathcal{C}$ closed under $\leq_\mathrm{m}^{\mathsf{L}}$ reductions, any set that is hard for $\mathcal{C}$ under $\leq_\mathrm{m}^{\mathsf{AC}^0}$ reductions is also hard under $\leq_\mathrm{m}^{\mathsf{NC}^0}$ reductions. Thus from Theorem 41, we have that MKTP is hard for co-NISZK$_\mathsf{L}$ under $\leq_\mathrm{m}^{\mathsf{NC}^0}$ reductions. ◀

▶ **Corollary 43.** MKTP *is hard for* co-NISZK$_\mathsf{L}$ *under nonuniform* $\leq_\mathrm{m}^{\mathsf{proj}}$ *reductions.*

Proof. We now need to claim that the AC^0-computable reduction of Theorem 41 can be replaced by a projection. Note that, since $\mathsf{SDU}_{\mathsf{NC}^0}$ is complete for NISZK$_\mathsf{L}$ under projections, and since the reduction from $\mathsf{SDU}_{\mathsf{NC}^0}$ to $\mathsf{EA}_{\mathsf{NC}^0}$ given in Corollary 25 always uses the same entropy bound $n - 3$, we have that it suffices to consider $\mathsf{EA}_{\mathsf{NC}^0}$ instances $x = (C_x, k)$ where the bound k depends only on the *length* of x. Thus the bound θ produced by our AC^0 reduction also depends only on the length of x, and hence can be hardwired in.

We now need only consider the string y. The $\leq_{\mathrm{m}}^{\mathsf{AC}^0}$ reduction presented in the proof of Theorem 41 takes as input C_x and r and produces the bits of y by feeding bits of r into C_x. Let us recall where the NC^0 circuitry producing the i-th bit of y comes from.

Lemma 35 shows how to take an arbitrary branching program and encode the problem of whether the program accepts as a question about the entropy of a distribution represented as a matrix of degree-three polynomials. Each term in this matrix is of the form $\sum_{j,k} R_{1\,(i,k)} L_{(k,j)} R_{2\,(j,m)}$, where the matrices R_1 and R_2 are the same for all inputs of of the same length. Thus we need only concern ourselves with the matrix L.

In Section 4.2.3, it is observed that, given an instance x of a promise problem in $\mathsf{NISZK_L}$, the branching program Q_x that is used, in order to build the matrix L, can be constructed from x by means of a projection. The "input" to this branching program Q_x is a sequence of random bits (part of the random sequence r that is hardwired in, in order to create the nonuniform AC^0 reduction in the proof of Theorem 41). Thus, the only entries of the matrix L that depend on x are entries of the form (u, v) where u and v are configurations of a logspace machine, where the machine is scanning x_i in configuration u, and it is possible to move to configuration v. Lemma 37 then shows how to construct NC^0 circuitry for each term in the degree-three polynomial constructed from Q_x in the proof of Lemma 35. The important thing to notice here is that each output bit in the NC^0 circuit depends on at most one term of one of the degree-three polynomials, and hence it depends on at most one entry of the matrix L – which means that it depends on at most one bit of the string x.

Thus, consider any bit y_i of the string y produced by the nonuniform AC^0 reduction from Theorem 41. Either y_i does not depend on any bit of x, or it depends on exactly one bit x_j of x. In the latter case, either $y_i = x_j$ or $y_i = \neg x_j$. This defines the projection, as required. ◀

6 An Application: Average-Case Complexity

The efficient reductions that we have presented have some immediate applications regarding worst-case to average-case reductions. First, we recall the definition of errorless heuristics:

▶ **Definition 44.** *Let A be any language. An* errorless heuristic *for A is an algorithm (or oracle) H such that, for every x, $H(x) \in \{$*YES, NO*, ?$\}$, and*

- $C_n(x) =$ YES *implies $x \in A$.*
- $C_n(x) =$ NO *implies $x \notin A$.*

▶ **Definition 45.** *A language A* has no average-case errorless heuristics in $\mathcal{C}$ *if, for every polynomial p, and every errorless heuristic $H \in \mathcal{C}$ for A, there exist infinitely many n such where $\Pr_{x \in U_n}[H(x) =?] > 1 - 1/p(n)$.*

In order to state our first theorem relating to average-case complexity, we need the following circuit-based definition:

▶ **Definition 46.** *Let $\mathcal{C}$ be any complexity class. (Usually, we will think of $\mathcal{C}$ being a class defined in terms of circuits, and the definition is thus phrased in terms of circuits, although it can be adapted for other complexity classes as well.) The class $\mathsf{OR} \circ \mathcal{C}$ is the class of problems that can be solved by a family of circuits whose output gate is an unbounded fan-in* OR *gate, connected to the outputs of circuits in the class $\mathcal{C}$.*

If problems in $\mathsf{NISZK_L}$ are hard in the worst case, then there are problems in NP that are hard on average:

▶ **Theorem 47.** *Let $\mathcal{C}$ be any complexity class that is closed under $\leq_{\mathrm{m}}^{\mathsf{proj}}$ reductions. If $\mathsf{NISZK_L} \not\subseteq \mathsf{OR} \circ \mathcal{C}$, then there is a set A in NP that has no average-case errorless heuristics in $\mathcal{C}$.*

Proof. Consider the reduction from $\overline{\mathsf{EA_{NC^0}}}$ to MKTP given in the proof of Corollary 43. This reduction takes as input a pair $(C, n-3)$ where C is an NC^0 circuit that produces output of length n. The reduction produces as output a string of length tn where $t = t(n)$ is a polynomial in n. The proof of Corollary 43 shows that, if $(C, n-3)$ is a No-instance (a low-entropy instance) of $\mathsf{EA_{NC^0}}$, then concatenating t samples from $C(r)$ (for independent uniformly random samples r) produces output that, with probability exponentially-close to 1, has KT-complexity less than $\theta < (n-2)t(n)$ for all large n. Let f be a function computed as follows: On input d of length m', compute the smallest n such that $m' < (n-2)t(n)$, and then simulate the universal Turing machine U on d for $t(n)^2$ steps, and produce as output the first $nt(n)$ bits of output that $U(d)$ produces in this amount of time. Let $A = \{y : \exists d\ f(d) = y\}$ be the range of f. Note that A contains all strings y of length $nt(n)$ such that $\mathsf{KT}(y) \leq (n-2)t(n)$. Clearly, $A \in \mathsf{NP}$. We will show that if A has an average-case errorless heuristic in $\mathcal{C}$, then $\mathsf{NISZK_L} \in \mathsf{OR} \circ \mathcal{C}$.[1]

If A has an average-case errorless heuristic in $\mathcal{C}$, then there is a family $\{C_m : m \in \mathbb{N}\}$ of $\mathcal{C}$ circuits (or other algorithms, if $\mathcal{C}$ is not a circuit family) with the property that, for all large n, for all strings x of length n, $C_n(x) \in \{\text{YES}, \text{NO}, ?\}$, where

- $C_n(x) = \text{YES}$ implies $x \in A$.
- $C_n(x) = \text{NO}$ implies $x \notin A$.
- $\Pr_x[C_n(x) = ?] < 1 - \frac{1}{p_1(n)}$

for some polynomial p_1.

Since there are three possible outputs, there must be two output bits, which we can call a and b. The encoding of YES, NO and ? below is chosen in order to simplify the statement of our results. If a different encoding is chosen, then the form of the circuits for $\mathsf{NISZK_L}$ might be slightly different.

a	b	
1	0	YES
0	1	NO
0	0	?
1	1	Illegal

Now consider the family $\{C'_m : m \in \mathbb{N}\}$, where C'_m is just like C_m but has only output bit b.

[1] In fact, A can be taken to be any set in NP that contains all strings of KT complexity below a certain threshold, while still containing only a small fraction of the strings of any length n.

For any $m = nt(n)$,

$$\begin{aligned}\Pr_x[C'_m(x) = 1] &= 1 - \Pr[C_m(x) = \text{YES}] - \Pr[C_m(x) = ?] \\ &\geq 1 - \frac{|A \cap \{0,1\}^m|}{2^m} - (1 - \frac{1}{p_1(m)}) \\ &\geq 1 - \frac{2^{(n-2)t(n)}}{2^{nt(n)}} - (1 - \frac{1}{p_1(m)}) \\ &= \frac{1}{p_1(nt(n))} - \frac{1}{2^{2t(n))}} \\ &> \frac{1}{p_2(n)}\end{aligned}$$

for some polynomial p_2.

We now present efficient circuits for promise problems in $\mathsf{NISZK_L}$.

Since the $\mathsf{NISZK_L}$-complete problem $\mathsf{EA_{NC^0}}$ is a special case of $\mathsf{EA_{BP}}$, we know that $\mathsf{EA_{BP}}$ is also complete for $\mathsf{NISZK_L}$ (say, under $\leq^{\mathsf{L}}_{\mathrm{m}}$ reductions). Thus it follows from Claim 40 that, for any problem $\prod \in \mathsf{NISZK_L}$, and for any instance $x \in \prod_{YES}$, the distribution M_x introduced in Section 4.2.3 can actually be assumed to have statistical difference at most $1/2^{|x|^\epsilon)}$ from the uniform distribution, for some $\epsilon > 0$. This in turn implies that the NC^0 circuit C_x (which is constructed in the paragraphs right after Claim 40) also has statistical difference at most $1/2^{|x|^\epsilon)}$ from the uniform distribution (again, if $x \in \prod_{YES}$). We highlight this fact, so that we can refer to it more easily later:

▷ Claim 48. If $x \in \prod_{YES}$, then the NC^0 circuit C_x has statistical difference at most $1/2^{|x|^\epsilon}$ from the uniform distribution.

Now consider the circuit family $\{D_{n_0} : n_0 \in \mathbb{N}\}$ that has the following form: The input is a string x of length n_0. Recall that the NC^0 circuit C_x from Section 4.2.3 takes "random" inputs r of length polynomial in $|x|$ and produces output of length n which is also polynomial in $|x|$. Let $\{E_n : n \in \mathbb{N}\}$ be a circuit family that takes (x, r) as input and computes $C_x(r)$. (The family E_n can in fact be chosen to be very efficient, but we do not need that; it will turn out later that E_n can be replaced by a single wire connected to a possibly-negated bit of x, or by a constant.) The "bottom layer" of D_{n_0} consists of $n_0^2 p_2^2(n)t(n)$ copies of E_n, using $n_0^2 p_2^2(n)t(n)$ independent random strings $r_1, \ldots, r_{n_0^2 p_2^2(n)t(n)}$, and producing a string of length $n_0^2 p_2^2(n)t(n)n$, which is then fed into $n_0^2 p_2^2(n)$ copies of $C'_{t(n)n}$. Finally, the output gate of each of the copies of $C'_{t(n)n}$ is fed into an OR gate, which is the output gate of D_{n_0}.

If $x \in \prod_{NO}$ then, as in the proof of Theorem 41, with probability (over the random inputs) exponentially close to 1, the string feeding into the inputs of each of the copies of C' has low KT complexity, and consequently (by the definition of C') each C' outputs 0, and thus D_{n_0} outputs 0.

If $x \in \prod_{YES}$ then, by Claim 48, the distribution represented by each copy of E_n (using random inputs r) has statistical difference from the uniform distribution bounded by 2^{-n^ϵ}. The strings that are fed into each copy of $C'_{nt(n)}$ are generated by $t(n)$ independent copies of E_n. By [30, Lemma 3.4], we can conclude that the distribution that is fed into each copy of $C'_{nt(n)}$ has statistical distance from the uniform distribution bounded by $\frac{t(n)}{2^{n^\epsilon}}$. We showed above that the probability that $C'_{nt(n)}$ accepts a uniformly-random string of length $nt(n)$ is greater than $\frac{1}{p_2(n)}$. It follows that the probability that $C'_{nt(n)}$ accepts the string produced by $t(n)$ independent copies of E_n is no less than $\frac{1}{p_2(n)} - \frac{t(n)}{2^{n^\epsilon}} > \frac{1}{np_2(n)}$. Thus the probability that *none* of the $n_0^2 p_2^2(n)$ independent copies of $C'_{nt(n)}$ accepts is at most $2^{-n_0^2}$.

A simple counting argument now shows that there is a sequence of probabilistic bits r that can be hardwired in to D_{n_0} so that, for all x of length n_0, $D_{n_0}(x,r) = 1$ if $x \in \prod_{YES}$ and $D_{n_0}(x,r) = 0$ if $x \in \prod_{NO}$. It still remains to simplify D_{n_0} so that it lies in $\mathsf{OR} \circ \mathcal{C}$.

As in the proof of Corollary 43, each bit that feeds into any of the copies of $C'_{nt(n)}$ depends on *at most one* bit of x; many of the bits may be set to constants after hardwiring in the choice of r. Thus we build the circuit family F_{n_0} that takes x as input, and projects the bits of x into the $n_0^2 p_2^2(n)$ copies of $C'_{nt(n)}$, to obtain a $\mathsf{OR} \circ \mathcal{C}$ circuit family for $\prod$. ◀

The following definition is implicit in the work of Bogdanov and Trevisan [12].

▶ **Definition 49.** *A* worst-case to errorless average-case reduction *from a promise problem $\prod$ to a language A is given by a polynomial p and* BPP *machine M, such that A is accepted by M^h for every oracle errorless heuristic H for A such that* $\Pr_{x \in U_n}[H(x) = ?] < 1 - 1/p(n)$.

▶ **Corollary 50.** *There is a problem $A \in$ NP such there is a non-adaptive worst-case to errorless average-case reduction from every problem in* SZK *to A.*

Proof. We mimic the proof of Theorem 47, and use the same set A. Consider the BPP reduction from the NISZK complete problem EA to MKTP given in Corollary 18. This reduction takes as input a pair (C, k) (where C is a circuit that produces output of length n) and makes a single query along each path, where the query is a string y of length tn where $t = t(n)$ is a polynomial in n. (Since SDU is complete for NISZK, we can assume that $k = n - 3$, as in the proof of Theorem 47.) Rather than using MKTP as an oracle, instead we will use an errorless heuristic H for A where the $\Pr_z[H(z) = ?] < 1 - 1/p(|z|)$, interpreting any answer where $H(y) =$ "No" as meaning "$\mathsf{KT}(y) > \theta$" and any answer where $H(y) \in \{?, \text{Yes}\}$ as meaning "$\mathsf{KT}(y) < \theta$". (We will actually replace each query to MKTP by a polynomial number of independent queries to the heuristic H, and if *any* of these queries returns $H(y) =$ "No", we will conclude that $(C, k) \in \mathsf{EA}_{YES}$, and otherwise conclude that $(C, k) \in \mathsf{EA}_{NO}$.)

As in the proof Theorem 47, if the distribution represented by C has low entropy, then with probability exponentially close to 1, the query y will have low KT complexity, and thus $H(y)$ will return a value in $\{?, \text{Yes}\}$ (and this probability will remain small even if a polynomial number of independent trials are made). And if C has high entropy, then (as in the proof of Theorem 47) we can assume that the distribution given by C is exponentially close to the uniform distribution, and thus the distribution on the queries y will have small statistical difference from the uniform distribution, and hence, with exponentially high probability, at least one of the queries will return the value No. Thus every problem in NISZK has an errorless non-adaptive worst-case to average-case reduction to A.

The proof is completed by recalling from [15] that SZK is non-adaptively (deterministically) polynomial-time reducible to NISZK. ◀

Remark: It is implicitly shown by Hirahara [17] that Corollary 50 holds under *adaptive* reductions. The significance of the improvement from adaptive and non-adaptive reductions lies in the fact that Bogdanov and Trevisan showed that the problems in NP for which there is a non-adaptive worst-case to errorless average-case reduction to a problem in NP lie in NP/poly ∩ coNP/poly [12, Remark (iii) in Section 4]. Thus SZK may be close to the largest class of problems for which non-adaptive worst-case to errorless average-case reductions to problems in NP exist.

The worst-case to average-case reductions of Definition 49, must work for *every* errorless heuristic that has a sufficiently small probability of producing "?" as output. If we consider a less-restrictive notion (allowing a different reduction for different errorless heuristics) then in some cases we can lower the complexity of the reduction from BPP to AC^0.

▶ **Definition 51.** *Let $\mathcal{D}$ be a complexity class, and let $\mathcal{R}$ be a class of reducibilities. We say that* errorless heuristics for language A are average-case hard for $\mathcal{D}$ under $\mathcal{R}$ reductions *if, for every polynomial p and every errorless heuristic H for A where $\Pr_{x \in U_{|x|}}[H(x) = ?] < 1 - 1/p(|x|)$, and for every language $B \in \mathcal{D}$, there is a reduction $r \in \mathcal{R}$ reducing B to H.*

▶ **Corollary 52.** *There is a language $A \in \mathsf{NP}$, such that errorless heuristics for A are average-case hard for $\mathsf{SZK_L}$ under non-adaptive AC^0-Turing reductions.*

Proof. The proof of Theorem 47 introduces a language $A \in \mathsf{NP}$ that is defined in terms of the parameters of the reduction from the $\mathsf{NISZK_L}$-complete promise problem $\mathsf{EA_{NC^0}}$. We show that errorless heuristics for this same A are average-case hard for $\mathsf{SZK_L}$ under non-adaptive AC^0-Turing reductions. By Proposition 3 and Theorem 26, every problem in $\mathsf{SZK_L}$ is non-adaptively AC^0-Turing-reducible to $\mathsf{EA_{NC^0}}$; let this AC^0-Turing reduction be computed by the family $\{D_n : n \in \mathbb{N}\}$. In the proof of Theorem 47, if we take the circuit family $\{C_m : m \in \mathbb{N}\}$ to consist of oracle gates for an errorless heuristic H for A, we obtain that every query that D_n makes to $\mathsf{EA_{NC^0}}$ can be replaced by an OR of queries consisting of oracle gates from $\{C_m : m \in \mathbb{N}\}$. This yields the desired non-adaptive AC^0-Turing reduction to the errorless heuristic for A. ◀

▶ **Corollary 53.** *Let $\mathcal{C}$ be any class that is closed under non-adaptive AC^0-Turing reductions. If $\mathsf{SZK_L} \not\subset \mathcal{C}$, then there is a problem in NP that has no average-case errorless heuristic in $\mathcal{C}$.*

Proof. If $\mathsf{SZK_L} \not\subset \mathcal{C}$, then by Proposition 3, $\mathsf{NISZK_L}$ is also not contained in $\mathcal{C}$. The result is now immediate from Theorem 47. ◀

Remark: Building on earlier work of Goldwasser et al. [16], average-case hardness results for some subclasses of P based on reductions computable by constant-depth threshold circuits were presented in [2]. (Although certain aspects of the reductions presented in [2, 16] are computable in AC^0, in order to obtain deterministic worst-case algorithms, $\mathsf{MAJORITY}$ gates are required in those constructions.) We are not aware of any prior work that provides average-case hardness results based on reductions computable in AC^0, particularly for classes that are believed to contain problems whose complexity is suitable for cryptographic applications.

7 Conclusion and Open Problems

By focusing on non-uniform versions of $\leq_m^{\mathsf{P}}$ reductions, we have shed additional light on how MKTP relates to subclasses of SZK. Some researchers are of the opinion that MCSP (and MKTP) are likely complete for NP under some type of reducibility, and some recent progress seems to support this [22]. For those who share this opinion, a plausible first step would be to show that MKTP is hard not only for $\mathsf{co\text{-}NISZK}$, but also for NISZK, under $\leq_m^{\mathsf{P/poly}}$ reductions. And of course, it will be very interesting to see if our hardness results for MKTP hold also for MCSP.

Acknowledgments

EA was supported in part by NSF Grant CCF-1909216. JG and CR were supported in part by the DIMACS 2020 REU program under NSF grants CCF-1852215 and CCF-1836666. We thank Dieter van Melkebeek and Andrew Morgan for helpful discussions.

References

1. Manindra Agrawal, Eric Allender, and Steven Rudich. Reductions in circuit complexity: An isomorphism theorem and a gap theorem. *Journal of Computer and System Sciences*, 57(2):127–143, 1998.
2. Eric Allender, V Arvind, Rahul Santhanam, and Fengming Wang. Uniform derandomization from pathetic lower bounds. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1971):3512–3535, 2012. doi:10.1098/rsta.2011.0318.
3. Eric Allender, Azucena Garvia Bosshard, and Amulya Musipatla. A note on hardness under projections for graph isomorphism and time-bounded Kolmogorov complexity. Technical Report TR20-158, Electronic Colloquium on Computational Complexity (ECCC), 2020.
4. Eric Allender, Harry Buhrman, Michal Koucký, Dieter Van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35(6):1467–1493, 2006. doi:10.1007/978-3-662-03927-4.
5. Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. *Information and Computation*, 256:2–8, 2017. Special issue for MFCS '14. doi:10.1016/j.ic.2017.04.004.
6. Eric Allender, Joshua A Grochow, Dieter Van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum circuit size, graph isomorphism, and related problems. *SIAM Journal on Computing*, 47(4):1339–1372, 2018. doi:10.1137/17M1157970.
7. Eric Allender and Shuichi Hirahara. New insights on the (non-) hardness of circuit minimization and related problems. *ACM Transactions on Computation Theory*, 11(4):1–27, 2019. doi:10.1145/3349616.
8. Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. *Computational Complexity*, 26(2):469–496, 2017. doi:10.1007/s00037-016-0124-0.
9. Eric Allender, Rahul Ilango, and Neekon Vafa. The non-hardness of approximating circuit size. *Theory of Computing Systems*, 2021. To appear. Special issue on CSR'19. doi:10.1007/s00224-020-10004-x.
10. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM Journal on Computing*, 36(4):845–888, 2006. doi:10.1137/S0097539705446950.
11. Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991. doi:10.1137/0220068.
12. Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006. doi:10.1137/S0097539705446974.
13. Zeev Dvir, Dan Gutfreund, Guy N Rothblum, and Salil P Vadhan. On approximating the entropy of polynomial mappings. In *Second Symposium on Innovations in Computer Science*, 2011.
14. Bin Fu. Hardness of sparse sets and minimal circuit size problem. In *Proc. Computing and Combinatorics - 26th International Conference (COCOON)*, volume 12273 of *Lecture Notes in Computer Science*, pages 484–495. Springer, 2020. doi:10.1007/978-3-030-58150-3_39.
15. Oded Goldreich, Amit Sahai, and Salil Vadhan. Can statistical zero knowledge be made non-interactive? or On the relationship of SZK and NISZK. In *Annual International Cryptology Conference*, pages 467–484. Springer, 1999. doi:10.1007/3-540-48405-1_30.
16. Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. A (de)constructive approach to program checking. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 143–152. ACM, 2008. doi:10.1145/1374376.1374399.

17 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 247–258. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00032.

18 Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In *35th Computational Complexity Conference (CCC)*, volume 169 of *LIPIcs*, pages 20:1–20:47. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.20.

19 Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *32nd Conference on Computational Complexity (CCC)*, volume 79 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.CCC.2017.7.

20 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *31st Conference on Computational Complexity (CCC)*, volume 50 of *LIPIcs*, pages 18:1–18:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.CCC.2016.18.

21 John M. Hitchcock and Aduri Pavan. On the NP-completeness of the minimum circuit size problem. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 45 of *LIPIcs*, pages 236–245. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.236.

22 Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In *35th Computational Complexity Conference (CCC)*, volume 169 of *LIPIcs*, pages 22:1–22:36. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.22.

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

24 Valentine Kabanets, Daniel M. Kane, and Zhenjian Lu. A polynomial restriction lemma with applications. In *Proceedings of the 49th Annual Symposium on Theory of Computing (STOC)*, pages 615–628. ACM, 2017. doi:10.1145/3055399.3055470.

25 Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984. doi:10.1016/S0019-9958(84)80060-1.

26 Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Proceedings of the 51st Annual Symposium on Theory of Computing (STOC)*, pages 1215–1225, 2019. doi:10.1145/3313276.3316396.

27 Cody Murray and Ryan Williams. On the (non) NP-hardness of computing circuit complexity. *Theory of Computing*, 13(4):1–22, 2017. doi:10.4086/toc.2017.v013a004.

28 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *34th Computational Complexity Conference (CCC)*, volume 137 of *LIPIcs*, pages 27:1–27:29. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.CCC.2019.27.

29 Michael Rudow. Discrete logarithm and minimum circuit size. *Information Processing Letters*, 128:1–4, 2017. doi:10.1016/j.ipl.2017.07.005.

30 Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003. doi:10.1145/636865.636868.

31 Michael Saks and Rahul Santhanam. Circuit lower bounds from np-hardness of MCSP under turing reductions. In *35th Computational Complexity Conference (CCC)*, volume 169 of *LIPIcs*, pages 26:1–26:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.26.

32 Heribert Vollmer. *Introduction to circuit complexity: a uniform approach.* Springer Science & Business Media, 1999. doi:10.1007/978-3-662-03927-4.

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