The Equivalence of Sampling and Searching

Scott Aaronson[*]

Abstract

In a *sampling problem*, we are given an input $x \in \{0,1\}^n$, and asked to sample approximately from a probability distribution $\mathcal{D}_x$ over poly (n)-bit strings. In a *search problem*, we are given an input $x \in \{0,1\}^n$, and asked to find a member of a nonempty set A_x with high probability. (An example is finding a Nash equilibrium.) In this paper, we use tools from Kolmogorov complexity to show that sampling and search problems are "essentially equivalent." More precisely, for any sampling problem S, there exists a search problem R_S such that, if $\mathcal{C}$ is any "reasonable" complexity class, then R_S is in the search version of $\mathcal{C}$ if and only if S is in the sampling version. What makes this nontrivial is that the same R_S works for every $\mathcal{C}$.

As an application, we prove the surprising result that $\mathsf{SampP} = \mathsf{SampBQP}$ *if and only if* $\mathsf{FBPP} = \mathsf{FBQP}$. In other words, classical computers can efficiently sample the output distribution of every quantum circuit, if and only if they can efficiently solve every search problem that quantum computers can solve.

1 Introduction

The *Extended Church-Turing Thesis (ECT)* says that all computational problems that are feasibly solvable in the physical world are feasibly solvable by a probabilistic Turing machine. By now, there have been almost two decades of discussion about this thesis, and the challenge that quantum computing poses to it. This paper is about a related question that has attracted surprisingly little interest: namely, what exactly should we understand the ECT to *state*? When we say "all computational problems," do we mean decision problems? promise problems? search problems? sampling problems? possibly other types of problems? Could the ECT hold for some of these types of problems but fail for others?

Our main result is an *equivalence* between sampling and search problems: the ECT holds for one type of problem if and only if it holds for the other. As a motivating example, we will prove the surprising fact that, if classical computers can efficiently solve any search problem that quantum computers can solve, then they can *also* approximately sample the output distribution of any quantum circuit. The proof makes essential use of Kolmogorov complexity. The technical tools that we will use are standard ones in the algorithmic information theory literature; our contribution is simply to apply those tools to obtain a useful equivalence principle in complexity theory that seems not to have been known before.

While the *motivation* for our equivalence theorem came from quantum computing, we wish to stress that the theorem itself is much more general, and has nothing to do with quantum computing in particular. Throughout this paper, we will use the *names* of quantum complexity classes—such as BQP (Bounded-Error Quantum Polynomial-Time), the class of languages efficiently decidable

[*]MIT. Email: aaronson@csail.mit.edu. This material is based upon work supported by the National Science Foundation under Grant No. 0844626, a TIBCO Chair, and an Alan T. Waterman award.

by a quantum algorithm—but only as "black boxes." No familiarity with quantum computing is needed.

The rest of the paper is organized as follows. Section 1.1 contains a general discussion of the relationships among decision problems, promise problems, search problems, and sampling problems; it can be safely skipped by readers already familiar with this material. Section 1.2 states our main result, as well as its implications for quantum computing in general and linear-optics experiments in particular. Section 1.3 explains how Kolmogorov complexity is used to prove the main result, and situates the result in the context of earlier work on Kolmogorov complexity. Next, in Section 2, we review some needed definitions and results from complexity theory (in Section 2.1), algorithmic information theory (in Section 2.2), and "standard" information theory (in Section 2.3). We prove the main result in Section 3, and the example application to quantum computing in Section 3.1. Finally, in Section 4, we present several extensions and generalizations of the main result, which address various shortcomings of it. Section 4 also discusses numerous open problems.

1.1 Background

Theoretical computer science has traditionally focused on *language decision problems*, where given a language $L \subseteq \{0,1\}^*$, the goal is to decide whether $x \in L$ for any input x. From this perspective, asking whether quantum computing contradicts the ECT is tantamount to asking:

Problem 1 *Does* $\mathsf{BPP} = \mathsf{BQP}$*?*

However, one can also consider *promise problems*, where the goal is to accept all inputs in a set $L_{\text{YES}} \subseteq \{0,1\}^*$ and reject all inputs in another set $L_{\text{NO}} \subseteq \{0,1\}^*$. Here L_{YES} and L_{NO} are disjoint, but their union is not necessarily all strings, and an algorithm can do whatever it likes on inputs not in $L_{\text{YES}} \cup L_{\text{NO}}$. Goldreich [5] has made a strong case that promise problems are at least as fundamental as language decision problems, if not more so. To give one relevant example, the task

Given a quantum circuit C, estimate the probability $p(C)$ that C accepts

is easy to formulate as a promise problem, but has no known formulation as a language decision problem. The reason is the usual "knife-edge" issue: given any probability $p^* \in [0,1]$ and error bound $\varepsilon \geq 1/\operatorname{poly}(n)$, we can ask a simulation algorithm to accept all quantum circuits C such that $p(C) \geq p^* + \varepsilon$, and to reject all circuits C such that $p(C) \leq p^* - \varepsilon$. But we cannot reasonably ask an algorithm to decide whether $p(C) = p^* + 2^{-n}$ or $p(C) = p^* - 2^{-n}$: if $p(C)$ is too close to p^*, then the algorithm's behavior is unknown.

Let $\mathsf{PromiseBPP}$ and $\mathsf{PromiseBQP}$ be the classes of promise problems solvable by probabilistic and quantum computers respectively, in polynomial time and with bounded probability of error. Then a second way to ask whether quantum mechanics contradicts the ECT is to ask:

Problem 2 *Does* $\mathsf{PromiseBPP} = \mathsf{PromiseBQP}$*?*

Now, if one accepts replacing languages by promise problems, then there seems little reason not to go further. One can also consider *search problems*, where given an input $x \in \{0,1\}^n$, the goal is to output any element of some nonempty "solution set" $A_x \subseteq \{0,1\}^{\operatorname{poly}(n)}$.[1] Perhaps the

[1]Search problems are also called "relational problems," for the historical reason that one can define such a problem using a binary relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, with $(x,y) \in R$ if and only if $y \in A_x$. Another name often used is

most famous example of a search problem is *finding a Nash equilibrium*, which Daskalakis et al. [3] showed to be complete for the class PPAD. By Nash's Theorem, every game has at least one Nash equilibrium, but the problem of finding one has no known formulation as either a language decision problem or a promise problem.

Let FBPP and FBQP be the classes of search problems solvable by probabilistic and quantum computers respectively, with success probability $1-\delta$, in time polynomial in n and $1/\delta$.[2] Then a third version of the "ECT question" is:

Problem 3 *Does* $\mathsf{FBPP} = \mathsf{FBQP}$*?*

There is yet another important type of problem in theoretical computer science. These are *sampling problems*, where given an input $x \in \{0,1\}^n$, the goal is to sample (exactly or, more often, approximately) from some probability distribution $\mathcal{D}_x$ over $\operatorname{poly}(n)$-bit strings. Well-known examples of sampling problems include sampling a random point in a high-dimensional convex body and sampling a random matching in a bipartite graph.

Let SampP and $\mathsf{SampBQP}$ be the classes of sampling problems that are solvable by probabilistic and quantum computers respectively, to within ε error in total variation distance, in time polynomial in n and $1/\varepsilon$.[3] Then a fourth version of our question is:

Problem 4 *Does* $\mathsf{SampP} = \mathsf{SampBQP}$*?*

Not surprisingly, *all* of the above questions are open. But we can ask an obvious meta-question:

> *What is the relationship among Problems 1-4? If the ECT fails in one sense, must it fail in the other senses as well?*

In one direction, there are some easy implications:

$$\begin{aligned} \mathsf{SampP} = \mathsf{SampBQP} &\Longrightarrow \mathsf{FBPP} = \mathsf{FBQP} && (1)\\ &\Longrightarrow \mathsf{PromiseBPP} = \mathsf{PromiseBQP} && (2)\\ &\Longrightarrow \mathsf{BPP} = \mathsf{BQP}. && (3) \end{aligned}$$

For the first implication, if every quantumly samplable distribution were also classically samplable, then given a quantum algorithm Q solving a search problem R, we could approximate Q's output distribution using a classical computer, and thereby solve R classically as well. For the second and third implications, every promise problem is also a search problem (with solution set $A_x \subseteq \{0,1\}$), and every language decision problem is also a promise problem (with the empty promise).

So the interesting part concerns the possible implications in the "other" direction. For example, could it be the case that $\mathsf{BPP} = \mathsf{BQP}$, yet $\mathsf{PromiseBPP} \neq \mathsf{PromiseBQP}$? Not only is this a formal possibility, but it does not even seem absurd, when we consider that

"function problems." But that is inaccurate, since the desired output is *not* a function of the input, except in the special case $|A_x| = 1$. We find "search problems" to be the clearest name, and will use it throughout. The one important point to remember is that a search problem need *not* be an NP search problem: that is, solutions need not be efficiently verifiable.

[2]The F in FBPP and FBQP stands for "function problem." Here we are following the standard naming convention, even though the term "function problem" is misleading for the reason pointed out earlier.

[3]Note that we write SampP instead of "$\mathsf{SampBPP}$" because there is no chance of confusion here. Unlike with decision, promise, and relation problems, with sampling problems it does not even make sense to talk about deterministic algorithms.

(1) the existing candidates for languages in $\mathsf{BQP} \setminus \mathsf{BPP}$ (for example, decision versions of the factoring and discrete log problems [8]) are all extremely "special" in nature, but

(2) $\mathsf{PromiseBQP}$ contains the "general" problem of estimating the acceptance probability of an arbitrary quantum circuit.

A second example of a difficult and unsolved meta-question is whether $\mathsf{PromiseBPP} = \mathsf{PromiseBQP}$ implies $\mathsf{SampP} = \mathsf{SampBQP}$. Translated into "physics language," the question is this: suppose we had an efficient classical algorithm to estimate the *expectation value* of any observable in quantum mechanics. Would that imply an efficient classical algorithm to *simulate any quantum experiment*, in the sense of sampling from a probability distribution close to the one quantum mechanics predicts? The difficulty is that, if we consider a quantum system of n particles, then a measurement could in general have c^n possible outcomes, each with probability on the order of c^{-n}. So, even supposing we could estimate any *given* probability to within $\pm\varepsilon$, in time polynomial in n and $1/\varepsilon$, that would seem to be of little help for the sampling task.

1.2 Our Results

This paper shows that *two* of the four types of problem discussed above—namely, sampling problems and search problems—are "equivalent" in a very non-obvious sense. Specifically, given any sampling problem S, we will construct a search problem $R = R_S$ such that, if $\mathcal{C}$ is any "reasonable" model of computation, then S is in $\mathsf{Samp}\mathcal{C}$ (the sampling version of $\mathcal{C}$) if and only if R is in $\mathsf{F}\mathcal{C}$ (the search version of $\mathcal{C}$). Here is a more formal statement of the result:

Theorem 5 (Sampling/Searching Equivalence Theorem) *Let S be any sampling problem. Then there exists a search problem R_S such that*

(i) If $\mathcal{O}$ is any oracle for S, then $R_S \in \mathsf{FBPP}^{\mathcal{O}}$.

(ii) If B is any probabilistic Turing machine solving R_S, then $S \in \mathsf{SampP}^{B}$.

(Importantly, the same search problem R_S works for all $\mathcal{O}$ and B.)

As one application, we show that the "obvious" implication $\mathsf{SampP} = \mathsf{SampBQP} \Longrightarrow \mathsf{FBPP} = \mathsf{FBQP}$ can be reversed:

Theorem 6 $\mathsf{FBPP} = \mathsf{FBQP}$ *if and only if* $\mathsf{SampP} = \mathsf{SampBQP}$. *In other words, classical computers can efficiently solve every* FBQP *search problem, if and only if they can approximately sample the output distribution of every quantum circuit.*

As a second application (which was actually the original motivation for this work), we extend a recent result of Aaronson and Arkhipov [1]. These authors gave a sampling problem that is solvable using a simple linear-optics experiment (so in particular, in $\mathsf{SampBQP}$), but is *not* solvable efficiently by a classical computer, unless the permanent of a Gaussian random matrix can be approximated in $\mathsf{BPP}^{\mathsf{NP}}$. More formally, consider the following problem, called $|\mathrm{GPE}|^2$ (the GPE stands for Gaussian Permanent Estimation):

Problem 7 ($|\mathbf{GPE}|^2$) *Given a matrix $X \in \mathbb{C}^{n\times n}$ of independent $\mathcal{N}(0,1)$ Gaussians, output a real number y such that*

$$\left| y - |\mathrm{Per}(X)|^2 \right| \leq \varepsilon \cdot n!, \tag{4}$$

with probability at least $1-\delta$ over $X \sim \mathcal{N}(0,1)_{\mathbb{C}}^{n\times n}$, in $\mathrm{poly}(n, 1/\varepsilon, 1/\delta)$ *time.*

The main result of [1] is the following:

Theorem 8 (Aaronson and Arkhipov [1]) $\mathsf{SampP} = \mathsf{SampBQP}$ *implies* $|\mathrm{GPE}|^2 \in \mathsf{FBPP}^{\mathsf{NP}}$.

The central conjecture made in [1] is that $|\mathrm{GPE}|^2$ is $\#\mathsf{P}$-complete. If this is the case, then $\mathsf{SampP} = \mathsf{SampBQP}$ would imply $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}$, which in turn would imply $\mathsf{PH} = \mathsf{BPP}^{\mathsf{NP}}$ by Toda's Theorem [9]. Or to put it differently: *we could rule out a polynomial-time classical algorithm to sample the output distribution of a quantum computer, under the sole assumption that the polynomial hierarchy is infinite.*

Now, by using Theorem 6 from this paper, we can deduce, in a completely "automatic" way, that the counterpart of Theorem 8 must hold with *search* problems in place of sampling problems:

Corollary 9 $\mathsf{FBPP} = \mathsf{FBQP}$ *implies* $|\mathrm{GPE}|^2 \in \mathsf{FBPP}^{\mathsf{NP}}$. *So in particular, assuming* $|\mathrm{GPE}|^2$ *is* $\#\mathsf{P}$*-complete and* PH *is infinite, it follows that* $\mathsf{FBPP} \neq \mathsf{FBQP}$.

Indeed, assuming $|\mathrm{GPE}|^2$ is $\#\mathsf{P}$-complete, even the containment $\mathsf{FBQP} \subseteq \mathsf{FBPP}^{\mathsf{PH}}$ would imply $\mathsf{P}^{\#\mathsf{P}} = \mathsf{PH}$ and hence (by Toda's Theorem) that the polynomial hierarchy collapses.

1.3 Proof Overview

Let us explain the basic difficulty we need to overcome to prove Theorem 5. Given a probability distribution $\mathcal{D}_x$ over $\{0,1\}^{\mathrm{poly}(n)}$, we want to define a set $A_x \subseteq \{0,1\}^{\mathrm{poly}(n)}$, such that the ability to *find* an element of A_x is equivalent to the ability to *sample* from $\mathcal{D}_x$. At first glance, such a general reduction seems impossible. For let $R = \{A_x\}_x$ be the search problem in which the goal is to find an element of A_x given x. Then consider an oracle $\mathcal{O}$ that, on input x, returns the lexicographically first element of A_x. Such an oracle $\mathcal{O}$ certainly solves R, but it seems useless if our goal is to *sample* uniformly from the set A_x (or indeed, from any other interesting distribution related to A_x).

Our solution will require going outside the black-box reduction paradigm.[4] In other words, given a sampling problem $S = \{\mathcal{D}_x\}_x$, we do *not* show that $S \in \mathsf{SampP}^{\mathcal{O}}$, where $\mathcal{O}$ is any oracle that solves the corresponding search problem R_S. Instead, we use the fact that $\mathcal{O}$ is computed by a Turing machine. We then define R_S in such a way that $\mathcal{O}$ must return, not just any element in the support of $\mathcal{D}_x$, but an element with *near-maximal Kolmogorov complexity.*

The idea here is simple: if a Turing machine B is probabilistic, then it can certainly output a string y with high Kolmogorov complexity, by just generating y at random. But the converse also holds: if B outputs a string y with high Kolmogorov complexity, then y *must* have been generated randomly. For otherwise, the code of B would constitute a succinct description of y.

Given any set $A \subseteq \{0,1\}^n$, it is not hard to use the above "Kolmogorov trick" to force a probabilistic Turing machine B to sample almost-uniformly from A. We simply ask B to produce k samples $y_1, \ldots, y_k \in A$, for some $k = \mathrm{poly}(n)$, such that the tuple $\langle y_1, \ldots, y_k \rangle$ has Kolmogorov complexity close to $k \log_2 |A|$. Then we output y_i for a uniformly random $i \in [k]$.

However, one can also generalize the idea, to force B to sample from an *arbitrary* distribution $\mathcal{D}$, not necessarily uniform. One way of doing this would be to reduce to the uniform case, by dividing the support of $\mathcal{D}$ into $\mathrm{poly}(n)$ "buckets," such that $\mathcal{D}$ is nearly-uniform within each bucket, and then asking B to output Kolmogorov-random elements in each bucket. In this paper, however, we will follow a more direct approach, which exploits the beautiful known connection

[4] This was previously done for different reasons in a cryptographic context—see for example Barak's beautiful PhD thesis [2].

between Kolmogorov complexity and Shannon information. In particular, we will use the notion of a *universal randomness test* from algorithmic information theory [6, 4]. Let $\mathcal{U}$ be the "universal prior," in which each string $y \in \{0,1\}^*$ occurs with probability proportional to $2^{-K(y)}$, where $K(y)$ is the prefix-free Kolmogorov complexity of y. Then given any computable distribution $\mathcal{D}$ and fixed string y, the universal randomness test provides a way to decide whether y was "plausibly drawn from $\mathcal{D}$," by considering the ratio $\Pr_{\mathcal{D}}[y] / \Pr_{\mathcal{U}}[y]$. The main technical fact we need to prove is simply that such a test can be applied in our *complexity-theoretic* context, where we care (for example) that the number of samples from $\mathcal{D}$ scales polynomially with the inverses of the relevant error parameters.

From one perspective, our result represents a surprising use of Kolmogorov complexity in the seemingly "distant" realm of polynomial-time reductions. Let us stress that we are *not* using Kolmogorov complexity as just a technical convenience, or as shorthand for a counting argument. Rather, Kolmogorov complexity seems essential even to define a search problem R_S with the properties we need. From another perspective, however, our use of Kolmogorov complexity is close in spirit to the reasons why Kolmogorov complexity was defined and studied in the first place! The whole point, after all, is to be able to talk about the "randomness of an individual object," without reference to any distribution from which the object was drawn. And that is exactly what we need, if we want to achieve the "paradoxical" goal of sampling from a distribution, using an oracle that is guaranteed only to output a *fixed* string y with specified properties.

2 Preliminaries

2.1 Sampling and Search Problems

We first formally define sampling problems, as well as the complexity classes SampP and SampBQP of sampling problems that are efficiently solvable by classical and quantum computers respectively.

Definition 10 (Sampling Problems, SampP, and SampBQP) *A sampling problem S is a collection of probability distributions $(\mathcal{D}_x)_{x\in\{0,1\}^*}$, one for each input string $x \in \{0,1\}^n$, where $\mathcal{D}_x$ is a distribution over $\{0,1\}^{p(n)}$, for some fixed polynomial p. Then* SampP *is the class of sampling problems $S = (\mathcal{D}_x)_{x\in\{0,1\}^*}$ for which there exists a probabilistic polynomial-time algorithm B that, given $\langle x, 0^{1/\varepsilon}\rangle$ as input, samples from a probability distribution $\mathcal{C}_x$ such that $\|\mathcal{C}_x - \mathcal{D}_x\| \leq \varepsilon$.* SampBQP *is defined the same way, except that B is a quantum algorithm rather than a classical one.*

Let us also define search problems, as well as the complexity classes FBPP and FBQP of search problems that are efficiently solvable by classical and quantum computers respectively.

Definition 11 (Search Problems, FBPP, and FBQP) *A search problem R is a collection of nonempty sets $(A_x)_{x\in\{0,1\}^*}$, one for each input string $x \in \{0,1\}^n$, where $A_x \subseteq \{0,1\}^{p(n)}$ for some fixed polynomial p. Then* FBPP *is the class of search problems $R = (A_x)_{x\in\{0,1\}^*}$ for which there exists a probabilistic polynomial-time algorithm B that, given an input $x \in \{0,1\}^n$ together with $0^{1/\varepsilon}$, produces an output y such that*

$$\Pr[y \in A_x] \geq 1 - \varepsilon, \tag{5}$$

where the probability is over B's internal randomness. FBQP *is defined the same way, except that B is a quantum algorithm rather than a classical one.*

2.2 Algorithmic Information Theory

We now review some basic definitions and results from the theory of Kolmogorov complexity. Recall that a set of strings $P \subset \{0,1\}^*$ is called *prefix-free* if no $x \in P$ is a prefix of any other $y \in P$.

Definition 12 (Kolmogorov complexity) *Fix a universal Turing machine U, such that the set of valid programs of U is prefix-free. Then $K(y)$, or the prefix-free Kolmogorov complexity of y, is the minimum length of a program x such that $U(x) = y$. We can also define the conditional Kolmogorov complexity $K(y|z)$, as the minimum length of a program x such that $U(\langle x, z\rangle) = y$.*

We are going to need two basic lemmas that relate Kolmogorov complexity to standard information theory, and that can be found in the book of Li and Vitányi [6] for example. The first lemma follows from Shannon's noiseless channel coding theorem, together with the Kraft-Chaitin lemma.

Lemma 13 *Let $\mathcal{D} = \{p_x\}$ be any computable distribution over strings, and let x be any element in the support of $\mathcal{D}$. Then*

$$K(x) \leq \log_2 \frac{1}{p_x} + K(\mathcal{D}) + O(1), \tag{6}$$

where $K(\mathcal{D})$ represents the length of the shortest program to sample from $\mathcal{D}$. The same holds if we replace $K(x)$ and $K(\mathcal{D})$ by $K(x|y)$ and $K(\mathcal{D}|y)$ respectively, for any fixed y.

The next lemma follows from a counting argument.

Lemma 14 ([6]) *Let $\mathcal{D} = \{p_x\}$ be any distribution over strings (not necessarily computable). Then there exists a universal constant b such that*

$$\Pr_{x \sim \mathcal{D}}\left[K(x) \geq \log_2 \frac{1}{p_x} - c\right] \geq 1 - \frac{b}{2^c}. \tag{7}$$

The same holds if we replace $K(x)$ by $K(x|y)$ for any fixed y.

2.3 Information Theory

This section reviews some basic definitions and facts from information theory. Let $\mathcal{A} = \{p_x\}_x$ and $\mathcal{B} = \{q_x\}_x$ be two probability distributions over $[N]$. Then recall that the *variation distance* between $\mathcal{A}$ and $\mathcal{B}$ is defined as

$$\|\mathcal{A} - \mathcal{B}\| := \frac{1}{2}\sum_{x=1}^{N} |p_x - q_x|, \tag{8}$$

while the *KL-divergence* is

$$D_{KL}(\mathcal{A}||\mathcal{B}) := \sum_{x=1}^{N} p_x \log_2 \frac{p_x}{q_x}. \tag{9}$$

The variation distance and the KL-divergence are related as follows:

Proposition 15 (Pinsker's Inequality) $\|\mathcal{A} - \mathcal{B}\| \leq \sqrt{2D_{KL}(\mathcal{A}||\mathcal{B})}$.

We will also need a fact about KL-divergence that has been useful in the study of parallel repetition, and that can be found (for example) in a paper by Rao [7].

Proposition 16 ([7]) *Let $\mathcal{R}$ be a distribution over $[N]^k$, with marginal distribution $\mathcal{R}_i$ on the i^{th} coordinate. Let $\mathcal{D}$ be a distribution over $[N]$. Then*

$$\sum_{i=1}^{k} D_{KL}\left(\mathcal{R}_i || \mathcal{D}\right) \leq D_{KL}\left(\mathcal{R} || \mathcal{D}^k\right). \tag{10}$$

3 Main Result

Let $S = \{\mathcal{D}_x\}_x$ be a sampling problem. Then our goal is to construct a search problem $R = R_S = \{A_x\}_x$ that is "equivalent" to S. Given an input of the form $\langle x, 0^{1/\delta}\rangle$, the goal in the search problem will be to produce an output Y such that $Y \in A_{x,\delta}$, with success probability at least $1-\delta$. The running time should be $\text{poly}(n, 1/\delta)$.

Fix an input $x \in \{0,1\}^n$, and let $\mathcal{D} := \mathcal{D}_x$ be the corresponding probability distribution over $\{0,1\}^m$. Let $p_y := \Pr_{\mathcal{D}}[y]$ be the probability of y. We now define the search problem R. Let $N := m/\delta^{2.1}$, and let $Y = \langle y_1, \ldots, y_N\rangle$ be an N-tuple of m-bit strings. Then we set $Y \in A_{x,\delta}$ if and only if

$$\log_2 \frac{1}{p_{y_1} \cdots p_{y_N}} \leq K(Y \mid x, \delta) + \beta, \tag{11}$$

where $\beta := 1 + \log_2 1/\delta$.

The first thing we need to show is that any algorithm that solves the sampling problem S also solves the search problem R with high probability.

Lemma 17 *Let $\mathcal{C} = \mathcal{C}_x$ be any distribution over $\{0,1\}^m$ such that $\|\mathcal{C} - \mathcal{D}\| \leq \varepsilon$. Then*

$$\Pr_{Y \sim \mathcal{C}^N}[Y \notin A_{x,\delta}] \leq \varepsilon N + \frac{b}{2^\beta}. \tag{12}$$

Proof. We have

$$\begin{aligned} \Pr_{Y \sim \mathcal{C}^N}[Y \notin A_{x,\delta}] &\leq \Pr_{Y \sim \mathcal{D}^N}[Y \notin A_{x,\delta}] + \left\|\mathcal{C}^N - \mathcal{D}^N\right\| & (13) \\ &\leq \Pr_{Y \sim \mathcal{D}^N}[Y \notin A_{x,\delta}] + \varepsilon N. & (14) \end{aligned}$$

So it suffices to consider a Y drawn from $\mathcal{D}^N$. By Lemma 14,

$$\Pr_{Y \sim \mathcal{D}^N}\left[K(Y \mid x, \delta) \geq \log_2 \frac{1}{p_{y_1} \cdots p_{y_N}} - \beta\right] \geq 1 - \frac{b}{2^\beta}. \tag{15}$$

Therefore

$$\Pr_{Y \sim \mathcal{D}^N}[Y \notin A_{x,\delta}] \leq \frac{b}{2^\beta}, \tag{16}$$

and we are done. ■

The second thing we need to show is that any algorithm that solves the search problem R also samples from a distribution that is close to $\mathcal{D}$ in variation distance.

Lemma 18 *Let B be a probabilistic Turing machine, which given input $\langle x, 0^{1/\delta}\rangle$ outputs an N-tuple $Y = \langle y_1, \ldots, y_N\rangle$ of m-bit strings. For some $\delta \leq 1/2$, suppose that*

$$\Pr\left[B\left(x, 0^{1/\delta}\right) \in A_{x,\delta}\right] \geq 1 - \delta, \tag{17}$$

where the probability is over B's internal randomness. Let $\mathcal{R} = \mathcal{R}_x$ be the distribution over outputs of $B(x)$, and let $\mathcal{C} = \mathcal{C}_x$ be the distribution over $\{0,1\}^m$ that is obtained by from $\mathcal{R}$ by choosing one of the y_i's uniformly at random. Then there exists a constant Q_B, depending on B, such that

$$\|\mathcal{C} - \mathcal{D}\| \leq \delta + Q_B\sqrt{\frac{\beta}{N}}. \tag{18}$$

Proof. Let $\mathcal{R}'$ be a distribution that is identical to $\mathcal{R}$, except that we condition on $B\left(x, 0^{1/\delta}\right) \in A_{x,\delta}$. Then by hypothesis, $\|\mathcal{R} - \mathcal{R}'\| \leq \delta$. Now let $\mathcal{R}'_i$ be the marginal distribution of $\mathcal{R}'$ on the i^{th} coordinate, and let

$$\mathcal{C}' = \frac{1}{N}\sum_{i=1}^{N} \mathcal{R}'_i \tag{19}$$

be the distribution over $\{0,1\}^m$ that is obtained from $\mathcal{R}'$ by choosing one of the y_i's uniformly at random. Then clearly $\|\mathcal{C} - \mathcal{C}'\| \leq \delta$ as well. So by the triangle inequality,

$$\|\mathcal{C} - \mathcal{D}\| \leq \left\|\mathcal{C} - \mathcal{C}'\right\| + \left\|\mathcal{C}' - \mathcal{D}\right\| \leq \delta + \left\|\mathcal{C}' - \mathcal{D}\right\|, \tag{20}$$

and it suffices to upper-bound $\|\mathcal{C}' - \mathcal{D}\|$.

Let $q_Y := \Pr_{\mathcal{R}}[Y]$. Then by Lemma 13,

$$K(Y \mid x, \delta) \leq \log_2 \frac{1}{q_Y} + K(\mathcal{R}) + O(1) \tag{21}$$

for all $Y \in (\{0,1\}^m)^N$. Also, for all $Y \in A_{x,\delta}$, by assumption we have

$$\log_2 \frac{1}{p_{y_1} \cdots p_{y_N}} \leq K(Y \mid x, \delta) + \beta. \tag{22}$$

Combining, for $Y \in A_{x,\delta}$ we have

$$\log_2 \frac{1}{p_{y_1} \cdots p_{y_N}} \leq \log_2 \frac{1}{q_Y} + K(\mathcal{R}) + O(1) + \beta. \tag{23}$$

This implies the following upper bound on the KL-divergence between $\mathcal{R}'$ and $\mathcal{D}^N$:

$$\begin{aligned} D_{KL}\left(\mathcal{R}' || \mathcal{D}^N\right) &\leq \frac{1}{1-\delta} \sum_{Y \in (\{0,1\}^m)^N \cap A_{x,\delta}} q_Y \log_2 \frac{q_Y}{p_{y_1} \cdots p_{y_N}} & (24)\\ &\leq \frac{1}{1-\delta} \max_{Y \in A_{x,\delta}} \log_2 \frac{q_Y}{p_{y_1} \cdots p_{y_N}} & (25)\\ &\leq \frac{1}{1-\delta}\left(K(\mathcal{R}) + O(1) + \beta\right). & (26) \end{aligned}$$

So by Proposition 16,

$$\sum_{i=1}^{N} D_{KL}\left(\mathcal{R}'_i || \mathcal{D}\right) \leq D_{KL}\left(\mathcal{R}' || \mathcal{D}^N\right) \leq \frac{1}{1-\delta}\left(K(\mathcal{R}) + O(1) + \beta\right), \tag{27}$$

and by Proposition 15,

$$\frac{1}{2}\sum_{i=1}^{N} \left\|\mathcal{R}'_i - \mathcal{D}\right\|^2 \leq \frac{1}{1-\delta}\left(K(\mathcal{R}) + O(1) + \beta\right). \tag{28}$$

So by Cauchy-Schwarz,

$$\sum_{i=1}^{N} \left\| \mathcal{R}'_i - \mathcal{D} \right\| \leq \sqrt{\frac{2N\left(\beta + K\left(\mathcal{R}\right) + O\left(1\right)\right)}{1-\delta}}. \tag{29}$$

Hence

$$\left\| \mathcal{C}' - \mathcal{D} \right\| \leq \sqrt{\frac{2\beta + 2K\left(\mathcal{R}\right) + O\left(1\right)}{\left(1-\delta\right)N}} \leq Q_B \sqrt{\frac{\beta}{N}} \tag{30}$$

for some constant Q_B depending on B, where we used the assumption $\delta \leq 1/2$. So

$$\left\| \mathcal{C} - \mathcal{D} \right\| \leq \left\| \mathcal{C} - \mathcal{C}' \right\| + \left\| \mathcal{C}' - \mathcal{D} \right\| \leq \delta + Q_B \sqrt{\frac{\beta}{N}}. \tag{31}$$

■

By combining Lemmas 17 and 18, we can now prove Theorem 5: that for any sampling problem $S = \left(\mathcal{D}_x\right)_{x\in\{0,1\}^*}$ (where $\mathcal{D}_x$ is a distribution over $m = m\left(n\right)$-bit strings), there exists a search problem $R_S = \left(A_x\right)_{x\in\{0,1\}^*}$ that is "equivalent" to S in the following two senses.

(i) Let $\mathcal{O}$ be any oracle that, given $\left\langle x, 0^{1/\varepsilon}, r \right\rangle$ as input, outputs a sample from a distribution $\mathcal{C}_x$ such that $\left\| \mathcal{C}_x - \mathcal{D}_x \right\| \leq \varepsilon$, as we vary the random string r. Then $R_S \in \mathsf{FBPP}^{\mathcal{O}}$.

(ii) Let B be any probabilistic Turing machine that, given $\left\langle x, 0^{1/\delta} \right\rangle$ as input, outputs a $Y \in \left(\{0,1\}^m\right)^N$ such that $Y \in A_{x,\delta}$ with probability at least $1-\delta$. Then $S \in \mathsf{SampP}^B$.

Proof of Theorem 5 (Sampling/Searching Equivalence Theorem). For part (i), given an input $\left\langle x, 0^{1/\delta} \right\rangle$, suppose we want to output an N-tuple $Y = \langle y_1, \ldots, y_N \rangle \in \left(\{0,1\}^m\right)^N$ such that $Y \in A_{x,\delta}$, with success probability at least $1-\delta$. Recall that $N = m/\delta^{2.1}$. Then the algorithm is this:

(1) Set $\varepsilon := \frac{\delta}{2N} = \frac{\delta^{3.1}}{2m}$.

(2) Call $\mathcal{O}$ on inputs $\left\langle x, 0^{1/\varepsilon}, r_1 \right\rangle, \ldots, \left\langle x, 0^{1/\varepsilon}, r_N \right\rangle$, where $r_1, \ldots, r_N$ are independent random strings, and output the result as $Y = \langle y_1, \ldots, y_N \rangle$.

Clearly this algorithm runs in $\operatorname{poly}\left(n, 1/\delta\right)$ time. Furthermore, by Lemma 17, its failure probability is at most

$$\varepsilon N + \frac{b}{2^{\beta}} \leq \delta. \tag{32}$$

For part (ii), given an input $\left\langle x, 0^{1/\varepsilon} \right\rangle$, suppose we want to sample from a distribution $\mathcal{C}_x$ such that $\left\| \mathcal{C}_x - \mathcal{D}_x \right\| \leq \varepsilon$. Then the algorithm is this:

(1) Set $\delta := \varepsilon/2$, so that $N = m/\delta^{2.1} = \Theta\left(m/\varepsilon^{2.1}\right)$.

(2) Call B on input $\left\langle x, 0^{1/\delta} \right\rangle$, and let $Y = \langle y_1, \ldots, y_N \rangle$ be B's output.

(3) Choose $i \in [N]$ uniformly at random, and output y_i as the sample from $\mathcal{C}_x$.

Clearly this algorithm runs in poly $(n, 1/\varepsilon)$ time. Furthermore, by Lemma 18 we have

$$\begin{aligned}\|\mathcal{C}_x - \mathcal{D}_x\| &\leq \delta + Q_B \sqrt{\frac{\beta}{N}} && (33)\\ &\leq \frac{\varepsilon}{2} + Q_B \sqrt{\frac{\varepsilon^{2.1}(2 + \log 1/\varepsilon)}{m}}, && (34)\end{aligned}$$

for some constant Q_B depending only on B. So in particular, there exists a constant C_B such that $\|\mathcal{C}_x - \mathcal{D}_x\| \leq \varepsilon$ for all $m \geq C_B$. For $m < C_B$, we can simply hardwire a description of $\mathcal{D}_x$ for every x into the algorithm (note that the algorithm can depend on B; we do not need a single algorithm that works for all B's simultaneously). ■

In particular, Theorem 5 means that $S \in \mathsf{SampP}$ if and only if $R_S \in \mathsf{FBPP}$, and likewise $S \in \mathsf{SampBQP}$ if and only if $R_S \in \mathsf{FBQP}$, and so on for any model of computation that is "below recursive" (i.e., simulable by a Turing machine), and that is able to implement the extremely simple algorithms in the proof of Theorem 5. (In particular, the model must be able to compute functions like $\varepsilon = \delta^{3.1}/2m$ and $N = m/\delta^{2.1}$, and it must be able to sample a polynomial number of independent random strings, and also choose one string uniformly at random from a polynomially long list.)

3.1 Implication for Quantum Computing

We now apply Theorem 5 to prove Theorem 6, that $\mathsf{SampP} = \mathsf{SampBQP}$ if and only if $\mathsf{FBPP} = \mathsf{FBQP}$.

Proof of Theorem 6. First, suppose $\mathsf{SampP} = \mathsf{SampBQP}$. Then consider a search problem $R = (A_x)_x$ in FBQP. By assumption, there exists a polynomial-time quantum algorithm Q that, given $\langle x, 0^{1/\delta} \rangle$ as input, outputs a y such that $y \in A_x$ with probability at least $1 - \delta$. Let $\mathcal{D}_{x,\delta}$ be the probability distribution over y's output by Q on input $\langle x, 0^{1/\delta} \rangle$. Then to solve R in FBPP, clearly it suffices to sample approximately from $\mathcal{D}_{x,\delta}$ in classical polynomial time. But we can do this by the assumption that $\mathsf{SampP} = \mathsf{SampBQP}$.[5]

Second, suppose $\mathsf{FBPP} = \mathsf{FBQP}$. Then consider a sampling problem S in $\mathsf{SampBQP}$. By Theorem 5, we can define a search counterpart R_S of S, such that

$$\begin{aligned}S \in \mathsf{SampBQP} &\Longrightarrow R_S \in \mathsf{FBQP} && (35)\\ &\Longrightarrow R_S \in \mathsf{FBPP} && (36)\\ &\Longrightarrow S \in \mathsf{SampP}. && (37)\end{aligned}$$

Hence $\mathsf{SampP} = \mathsf{SampBQP}$. ■

Theorem 6 is easily seen to relativize: for all oracles A, we have $\mathsf{SampP}^A = \mathsf{SampBQP}^A$ if and only if $\mathsf{FBPP}^A = \mathsf{FBQP}^A$. (Of course, when proving a relativized version of Theorem 5, we have to be careful to define the search problem R_S using Kolmogorov complexity for Turing machines with A-oracles.)

[5] As mentioned in Section 1, the same argument shows that $\mathsf{SampP} = \mathsf{SampBQP}$ (or equivalently, $\mathsf{FBPP} = \mathsf{FBQP}$) implies $\mathsf{BPP} = \mathsf{BQP}$. However, the converse is far from clear: we have no idea whether $\mathsf{BPP} = \mathsf{BQP}$ implies $\mathsf{SampP} = \mathsf{SampBQP}$.

4 Extensions and Open Problems

4.1 Equivalence of Sampling and *Decision* Problems?

Perhaps the most interesting question we leave open is whether any nontrivial equivalence holds between sampling (or search) problems on the one hand, and *decision* or *promise* problems on the other. In Theorem 5, it was certainly essential to consider large numbers of outputs; we would have no idea how to prove an analogous result with a promise problem P_S or language L_S instead of the search problem R_S.

One way to approach this question is as follows: does there exist a sampling problem S that is provably *not* equivalent to any decision problem, in the sense that for every language $L \subseteq \{0,1\}^*$, either $S \notin \mathsf{SampP}^L$, or else there exists an oracle $\mathcal{O}$ solving S such that $L \notin \mathsf{BPP}^{\mathcal{O}}$? What if we require the oracle $\mathcal{O}$ to be computable? As far as we know, these questions are open.

One might object that, given any sampling problem S, it is easy to define a language L_S that is "equivalent" to S, by using the following simple enumeration trick. Let $M_1, M_2, \ldots$ be an enumeration of probabilistic Turing machines with polynomial-time alarm clocks. Given a sampling problem $S = (\mathcal{D}_x)_{x \in \{0,1\}^*}$ and an input $X = \langle x, 0^{1/\varepsilon} \rangle$, say that M_t *succeeds* on X if $M_t(X)$ samples from a distribution $\mathcal{C}_X$ such that $\|\mathcal{C}_X - \mathcal{D}_x\| \leq \varepsilon$. Also, if x is an n-bit string, define the *length* of $X = \langle x, 0^{1/\varepsilon} \rangle$ to be $\ell(X) := n + 1/\varepsilon$.

We now define a language $L_S \subseteq \{0,1\}^*$. For all n, let $f(n)$ be the least t such that M_t succeeds on all inputs X satisfying $\ell(X) \leq n$. Then for all $y \in \{0,1\}^n$, we set $y \in L_S$ if and only if the Turing machine encoded by y halts in at most $n^{f(n)}$ steps when run on a blank tape.

Proposition 19 *$S \in \mathsf{SampP}$ if and only if $L_S \in \mathsf{P}$.*

Proof. First suppose $S \in \mathsf{SampP}$. Then there exists a polynomial-time Turing machine that succeeds on every input $X = \langle x, 0^{1/\varepsilon} \rangle$. Let M_t be the lexicographically first such machine. Then it is not hard to see that L_S consists of a finite prefix, followed by the n^t-time bounded halting problem. Hence $L_S \in \mathsf{P}$.

Next suppose $S \notin \mathsf{SampP}$. Then *no* machine M_t succeeds on every input X, so $f(n)$ grows without bound as a function of n. By standard diagonalization arguments, the $n^{f(n)}$-time bounded halting problem is not in P for any f that grows without bound, regardless of whether f is time-constructible.[6] Therefore $L_S \notin \mathsf{P}$. ■

Admittedly, Proposition 19 feels like cheating—but *why* exactly is it cheating? Notice that we *did* give a procedure to decide whether $y \in L_S$ for any input y. This fact makes Proposition 19 at least *somewhat* more interesting than the "tautological" way to ensure $S \in \mathsf{SampP} \Longleftrightarrow L_S \in \mathsf{P}$:

> *"Take L_S to be the empty language if $S \in \mathsf{SampP}$, or an EXP-complete language if $S \notin \mathsf{SampP}$!"*

In our view, the real problem with Proposition 19 is that it uses enumeration of Turing machines to avoid the need to *reduce* the sampling problem S to the language L_S or vice versa. Of course, Theorem 5 did not quite reduce S to the search problem R_S either. However, Theorem 5 came "close enough" to giving a reduction that we were able to use it to derive interesting consequences for complexity theory, such as $\mathsf{SampP} = \mathsf{SampBQP}$ if and only if $\mathsf{FBPP} = \mathsf{FBQP}$. If we attempted to prove similar consequences from Proposition 19, then we would end up with a *different* language

[6] Note, also, that it is irrelevant whether there exists a polynomial p such that $M_{f(n)}$ halts in at most $p(n)$ steps for all n. The *index* $f(n)$ determines how long we need to simulate y for, not the running time of $M_{f(n)}$.

L_S, depending on whether our starting assumption was $S \in \mathsf{SampP}$, $S \in \mathsf{SampBQP}$, or some other assumption. By contrast, Theorem 5 constructed a *single* search problem R_S that is equivalent to S in the classical model, the quantum model, and every other "reasonable" computational model.

4.2 Was Kolmogorov Complexity Necessary?

Could we have proved Theorem 5 *without* using Kolmogorov complexity or anything like it, and without making a computability assumption on the oracle for R_S? One way to formalize this question is to ask the analogue of our question from Section 4.1, but this time for sampling versus *search* problems. In other words, does there exist a sampling problem S such that, for every search problem R, either there exists an oracle $\mathcal{O}$ solving S such that $R \notin \mathsf{FBPP}^{\mathcal{O}}$, or there exists an oracle $\mathcal{O}$ solving R such that $S \notin \mathsf{SampP}^{\mathcal{O}}$? Notice that, if R is the search problem from Theorem 5, then the latter oracle (if it exists) must be uncomputable. Thus, we are essentially asking whether the computability assumption in Theorem 5 was necessary.

4.3 From Search Problems to Sampling Problems

Theorem 5 showed how to take any sampling problem S, and define a search problem $R = R_S$ that is equivalent to S. Can one go the other direction? That is, given a search problem R, can one define a sampling problem $S = S_R$ that is equivalent to R? The following theorem is the best we were able to find in this direction.

Theorem 20 *Let $R = (A_x)_x$ be any search problem. Then there exists a sampling problem $S_R = \{\mathcal{D}_x\}_x$ that is "almost equivalent" to R, in the following senses.*

(i) If $\mathcal{O}$ is any oracle solving S_R, then $R \in \mathsf{FBPP}^{\mathcal{O}}$.

(ii) If B is any probabilistic Turing machine solving R, then there exists a constant $\eta_B > 0$ such that a SampP^B machine can sample from a probability distribution $\mathcal{C}_x$ with $\|\mathcal{C}_x - \mathcal{D}_x\| \le 1-\eta_B$.

Proof. Let $\mathcal{U}_x$ be the universal prior, in which every string y occurs with probability at least $c \cdot 2^{-K(y|x)}$, for some constant $c > 0$. Then to define the sampling problem S_R, we let $\mathcal{D}_x$ be the distribution obtained by drawing $y \sim \mathcal{U}_x$ and then conditioning on the event $y \in A_x$. (Note that $\mathcal{D}_x$ is well-defined, since $\mathcal{U}_x$ assigns nonzero probability to every y.)

For (i), notice that $\mathcal{D}_x$ has support only on A_x. So if we can sample a distribution $\mathcal{C}_x$ such that $\|\mathcal{C}_x - \mathcal{D}_x\| \le \varepsilon$, then certainly we can output an element of A_x with probability at least $1-\varepsilon$.

For (ii), let $\mathcal{C}_{x,\delta}$ be the distribution over values of $B\left(x, 0^{1/\delta}, r\right)$ induced by varying the random string r. Then we claim that $\|\mathcal{C}_{x,\delta} - \mathcal{D}_x\| \le 1 - \Omega(1)$, so long as $\delta \le \Delta_B$ for some constant Δ_B depending on B. To see this, first let $\mathcal{C}'$ be the distribution obtained by drawing $y \sim \mathcal{C}_{x,\delta}$ and then conditioning on the event $y \in A_x$. Then since $\Pr_{y \sim \mathcal{C}_{x,\delta}}[y \in A_x] \ge 1 - \delta$, we have $\|\mathcal{C}' - \mathcal{C}_{x,\delta}\| \le \delta$.

Now let $q_y := \Pr_{\mathcal{C}'}[y]$. Then by Lemma 13, there exists a constant g_B depending on B such that

$$q_y \le g_B \cdot 2^{-K(y|x)} \tag{38}$$

for all $y \in A_x$. On the other hand, let $p_y := \Pr_{\mathcal{D}_x}[y]$ and $u_y := \Pr_{\mathcal{U}_x}[y]$. Then there exists a constant $\alpha \ge 1$ such that $p_y = \alpha u_y$ if $y \in A_x$ and $p_y = 0$ otherwise. So

$$p_y \ge u_y \ge c \cdot 2^{-K(y|x)} \tag{39}$$

for all $y \in A_x$. Hence $p_y \geq \frac{c}{g_B} q_y$, so

$$\left\| \mathcal{C}' - \mathcal{D}_x \right\| = \sum_{y \in A_x \,:\, p_y < q_y} |p_y - q_y| \leq 1 - \frac{c}{g_B}. \tag{40}$$

Therefore

$$\left\| \mathcal{C}_{x,\delta} - \mathcal{D}_x \right\| \leq \left\| \mathcal{C}_{x,\delta} - \mathcal{C}' \right\| + \left\| \mathcal{C}' - \mathcal{D}_x \right\| \leq 1 - \frac{c}{g_B} + \delta, \tag{41}$$

which is $1 - \Omega_B(1)$ provided $\delta \leq \frac{c}{2g_B}$. ■

We see it as an interesting problem whether Theorem 20 still holds with the condition $\|\mathcal{C}_x - \mathcal{D}_x\| \leq 1 - \eta_B$ replaced by $\|\mathcal{C}_x - \mathcal{D}_x\| \leq \varepsilon$ (in other words, with $S_R \in \mathsf{SampP}^B$).

4.4 Making the Search Problem Checkable

One obvious disadvantage of Theorem 5 is that the search problem $R = (A_x)_x$ is defined using Kolmogorov complexity, which is uncomputable. In particular, there is no algorithm to decide whether $y \in A_x$. However, it is not hard to fix this problem, by replacing the Kolmogorov complexity with the *time-bounded* or *space-bounded* Kolmogorov complexities in our definition of R. The price is that we then also have to assume a complexity bound on the Turing machine B in the statement of Theorem 5. In more detail:

Theorem 21 *Let S be any sampling problem, and let f be a time-constructible function. Then there exists a search problem $R_S = (A_x)_x$ such that*

(i) If $\mathcal{O}$ is any oracle solving S, then $R_S \in \mathsf{FBPP}^{\mathcal{O}}$.

(ii) If B is any $\mathsf{BPTIME}(f(n))$ Turing machine solving R_S, then $S \in \mathsf{SampP}^B$.

(iii) There exists a $\mathsf{SPACE}\left(f(n) + n^{O(1)}\right)$ algorithm to decide whether $y \in A_x$, given x and y.

Proof Sketch. The proof is almost the same as the proof of Theorem 5. Let $T := f(n) + n^{O(1)}$, and given a string y, let $K_{\mathrm{SPACE}(T)}(y)$ be the T-space bounded Kolmogorov complexity of y. Then the only real difference is that, when defining the search problem R_S, we replace the conditional Kolmogorov complexity $K(Y \mid x, \delta)$ by the space-bounded complexity $K_{\mathrm{SPACE}(T)}(Y \mid x, \delta)$. This ensures that property (iii) holds.

Certainly property (i) still holds, since it only used the fact that there are few tuples $Y \in (\{0,1\}^m)^N$ with small Kolmogorov complexity, and that is still true for space-bounded Kolmogorov complexity.

For property (ii), it suffices to observe that Lemma 13 has the following "effective" version. Let $\mathcal{D} = \{p_y\}$ be any distribution over strings that is samplable in $\mathsf{BPTIME}(f(n))$, and let y be any element in the support of $\mathcal{D}$. Then there exists a constant $C_{\mathcal{D}}$, depending on $\mathcal{D}$, such that

$$K_{\mathrm{SPACE}(T)}(y) \leq \log_2 \frac{1}{p_y} + C_{\mathcal{D}}. \tag{42}$$

The proof is simply to notice that, in $\mathsf{SPACE}\left(f(n) + n^{O(1)}\right)$, we can compute the probability p_y of *every* y in the support of $\mathcal{D}$, and can therefore recover any particular string y from its Shannon-Fano code. This means that the analogue of Lemma 18 goes through, as long as B is a $\mathsf{BPTIME}(f(n))$ machine. ■

In Theorem 21, how far can we decrease the computational complexity of R_S? It is not hard to replace the upper bound of $\mathsf{SPACE}\left(f(n)+n^{O(1)}\right)$ by $\mathsf{CH}\left(f(n)+n^{O(1)}\right)$ (where CH denotes the counting hierarchy), but can we go further? It seems unlikely that one could check in NP (or $\mathsf{NTIME}\left(f(n)+n^{O(1)}\right)$) whether $y \in A_x$, for a search problem $R_S = \{A_x\}_x$ equivalent to S, but can we give formal evidence against this possibility?

5 Acknowledgments

I thank Alex Arkhipov for helpful discussions that motivated this work, and Dana Moshkovitz for pointing me to Proposition 16 from [7]. I also thank Nikolay Vereshchagin for catching a bug in the proof of Lemma 18, and the anonymous reviewers for several helpful comments.

References

[1] S. Aaronson and A. Arkhipov. The computational complexity of linear optics. *Theory of Computing*, 9(4):143–252, 2013. Conference version in Proceedings of ACM STOC'2011. ECCC TR10-170, arXiv:1011.3245.

[2] B. Barak. *Non-Black-Box Techniques in Cryptography.* PhD thesis, Weizmann Institute of Science, 2003. www.wisdom.weizmann.ac.il/˜oded/PS/boaz-phd.ps.

[3] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. *Commun. ACM*, 52(2):89–97, 2009. Earlier version in Proceedings of STOC'2006.

[4] P. Gács. Lecture notes on descriptional complexity and randomness. www.cs.bu.edu/˜gacs/papers/ait-notes.pdf, 2010.

[5] O. Goldreich. On promise problems: a survey. In *Essays in Memory of Shimon Even*, pages 254–290. 2006. ECCC TR05-018.

[6] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications (3rd ed.).* Springer, 2008. First edition published in 1993.

[7] A. Rao. Parallel repetition in projection games and a concentration bound. *SIAM J. Comput.*, 40(6):1871–1891, 2011. ECCC TR08-013. Earlier version in STOC'2008.

[8] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. Earlier version in IEEE FOCS 1994. quant-ph/9508027.

[9] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.