The equivalence of sampling and searching

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

| Citation     | Aaronson, Scott. “The Equivalence of Sampling and Searching.” Computer Science – Theory and Applications. Ed. Alexander Kulikov & Nikolay Vereshchagin. Vol. 6651. Lecture Notes in Computer Science: Springer Berlin Heidelberg, 2011. 1-14. Web. 9 Aug. 2012. |
|--------------|---------------------------------------------------------------------------------------------------------------|
| As Published | http://dx.doi.org/10.1007/978-3-642-20712-9_1                                                                |
| Publisher    | Springer Berlin/Heidelberg                                                                                     |
| Version      | Author's final manuscript                                                                                      |
| Accessed     | Wed Jan 23 05:37:18 EST 2019                                                                                   |
| Citable Link | http://hdl.handle.net/1721.1/72065                                                                             |
| Terms of Use | Creative Commons Attribution-Noncommercial-Share Alike 3.0                                                    |
| Detailed Terms | http://creativecommons.org/licenses/by-nc-sa/3.0/                                                              |
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 $D_x$ 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 $C$ is any "reasonable" complexity class, then $R_S$ is in the search version of $C$ if and only if $S$ is in the sampling version. What makes this nontrivial is that the same $R_S$ works for every $C$.

As an application, we prove the surprising result that $\text{SampP} = \text{SampBQP}$ if and only if $\text{FBPP} = \text{FBQP}$. 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.

*MIT. Email: aaronson@csail.mit.edu. This material is based upon work supported by the National Science Foundation under Grant No. 0844626. Also supported by a DARPA YFA grant, a TIBCO Chair, and the Sloan Foundation.
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 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 \( \text{BPP} = \text{BQP} \)?

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 \( L_{\text{YES}} \) and \( L_{\text{NO}} \) 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

\[
\text{Given a quantum circuit } C, \text{ estimate the probability } p(C) \text{ that } C \text{ accepts}
\]

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 \( \varepsilon \geq 1/\text{poly} \,(n) \), 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 \( \text{PromiseBPP} \) and \( \text{PromiseBQP} \) 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 \( \text{PromiseBPP} = \text{PromiseBQP} \)?

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” $A_x \subseteq \{0,1\}^{\text{poly}(n)}$. Perhaps the 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 $1/\delta$.

Then a third version of the “ECT question” is:

Problem 3 Does FBPP = 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 $D_x$ over 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 SampBQP be the classes of sampling problems that are solvable by probabilistic and quantum computers respectively, to within $\varepsilon$ error in total variation distance, in time polynomial in $n$ and $1/\varepsilon$.

Then a fourth version of our question is:

Problem 4 Does SampP = 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:

$\text{SampP} = \text{SampBQP} \implies \text{FBPP} = \text{FBQP}$
$\implies \text{PromiseBPP} = \text{PromiseBQP}$
$\implies \text{BPP} = \text{BQP}$.

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

1Search 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 “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.

2The 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.

3Note that we write Samp instead of “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.
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 \( \text{BPP} = \text{BQP} \), yet \( \text{PromiseBPP} \neq \text{PromiseBQP} \)? Not only is this a formal
possibility, but it does not even seem absurd, when we consider that

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

(2) \( \text{PromiseBQP} \) 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 \( \text{PromiseBPP} = \text{PromiseBQP} \)
implies \( \text{SampP} = \text{SampBQP} \). 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 pre-
dicts? 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 \( \text{SampC} \) (the sampling version of \( \mathcal{C} \)) if and only if \( R \) is in \( \text{FC} \) (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 \( R_S \in \text{FBPP}^\mathcal{O} \).

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

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

As one application, we show that the “obvious” implication \( \text{SampP} = \text{SampBQP} \Rightarrow \text{FBPP} = \text{FBQP} \) can be reversed:

**Theorem 6** \( \text{FBPP} = \text{FBQP} \) if and only if \( \text{SampP} = \text{SampBQP} \). In other words, classical comput-
ers can efficiently solve every \( \text{FBQP} \) 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 $\text{SampBQP}$), but is not solvable efficiently by a classical computer, unless the permanent of a Gaussian random matrix can be approximated in $\text{BPP^{NP}}$. More formally, consider the following problem, called $|\text{GPE}|^2$ (the GPE stands for Gaussian Permanent Estimation):

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

$$|y - |\text{Per}(X)|^2| \leq \epsilon \cdot n!,$$

with probability at least $1 - \delta$ over $X \sim \mathcal{N}(0, 1)^{n \times n}$, in $\text{poly}(n, 1/\epsilon, 1/\delta)$ time.

The main result of [1] is the following:

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

The central conjecture made in [1] is that $|\text{GPE}|^2$ is $\#P$-complete. If this is the case, then $\text{SampP} = \text{SampBQP}$ would imply $P^{\#P} = \text{BPP}^{NP}$, which in turn would imply $\text{PH} = \text{BPP}^{NP}$ 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** $\text{FBPP} = \text{FBQP}$ implies $|\text{GPE}|^2 \in \text{FBPP}^{NP}$. So in particular, assuming $|\text{GPE}|^2$ is $\#P$-complete and $\text{PH}$ is infinite, it follows that $\text{FBPP} \neq \text{FBQP}$.

Indeed, assuming $|\text{GPE}|^2$ is $\#P$-complete, we cannot even have $\text{FBQP} \subseteq \text{FBPP}^{PH}$, unless $P^{\#P} = \text{PH}$ and 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 $\{0, 1\}^{\text{poly}(n)}$, we want to define a set $A_x \subseteq \{0, 1\}^{\text{poly}(n)}$, 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 $O$ that, on input $x$, returns the lexicographically first element of $A_x$. Such an oracle $O$ 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. In other words, given a sampling problem $S = \{\mathcal{D}_x\}_x$, we do not show that $S \in \text{SampP}^O$, where $O$ is any oracle that solves the corresponding search problem $R_S$. Instead, we use the fact that $O$ is computed by

---

Footnote: This was previously done for different reasons in a cryptographic context—see for example Barak’s beautiful PhD thesis [2].
a Turing machine. We then define $R_S$ in such a way that $O$ must return, not just any element in the support of $D_x$, but an element with \textit{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 $k = \text{poly} (n)$, such that the tuple $(y_1, \ldots, y_k)$ 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 \textit{arbitrary} distribution $D$, not necessarily uniform. One way of doing this would be to reduce to the uniform case, by dividing the support of $D$ into $\text{poly} (n)$ “buckets,” such that $D$ 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 between Kolmogorov complexity and Shannon information. In particular, we will use the notion of a \textit{universal randomness test} from algorithmic information theory [6, 4]. Let $\mathcal{U}$ be the “universal prior,” in which each string $y \in \{0,1\}^\omega$ occurs with probability proportional to $2^{-K(y)}$, where $K(y)$ is the prefix-free Kolmogorov complexity of $y$. Then given any computable distribution $D$ and fixed string $y$, the universal randomness test provides a way to decide whether $y$ was “plausibly drawn from $D$,” by considering the ratio $\Pr_D[y] / \Pr_U[y]$. The main technical fact we need to prove is simply that such a test can be applied in our \textit{complexity-theoretic} context, where we care (for example) that the number of samples from $D$ 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 \textit{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 \textit{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 $\text{SampP}$ and $\text{SampBQP}$ of sampling problems that are efficiently solvable by classical and quantum computers respectively.

\textbf{Definition 10 (Sampling Problems, SampP, and SampBQP)} A \textit{sampling problem} $S$ is a collection of probability distributions $(D_x)_{x \in \{0,1\}^n}$, one for each input string $x \in \{0,1\}^n$, where $D_x$ is a distribution over $\{0,1\}^{p(n)}$, for some fixed polynomial $p$. Then $\text{SampP}$ is the class of sampling problems $S = (D_x)_{x \in \{0,1\}^n}$, for which there exists a probabilistic polynomial-time algorithm $B$
that, given \((x, 0^{1/\varepsilon})\) as input, samples from a probability distribution \(C_x\) such that \(\|C_x - D_x\| \leq \varepsilon\). \[\text{SampBQP}\] 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 \(\text{FBPP}\) and \(\text{FBQP}\) 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 \(\text{FBPP}\) 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 \(\Pr[y \in A_x] \geq 1 - \varepsilon\), where the probability is over \(B\)'s internal randomness. \(\text{FBQP}\) 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 \(K(y|x)\), as the minimum length of a program \(x\) such that \(U((x,z)) = y\).

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 almost immediately from Shannon’s noiseless channel coding theorem.

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

\[
K(x) \leq \log_2 \frac{1}{p_x} + K(D) + O(1),
\]

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

The next lemma follows from a counting argument.

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

\[
\Pr_{x \sim D}[K(x) \geq \log_2 \frac{1}{p_x} - c] \geq 1 - \frac{b}{2^c}.
\]

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 \( A = \{p_x\}_x \) and \( B = \{q_x\}_x \) be two probability distributions over \([N]\). Then recall that the variation distance between \( A \) and \( B \) is defined as

\[
\|A - B\| := \frac{1}{2} \sum_{x=1}^{N} |p_x - q_x|,
\]

while the KL-divergence is

\[
D_{KL}(A||B) := \sum_{x=1}^{N} p_x \log_2 \frac{p_x}{q_x}.
\]

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

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

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 \( R \) be a distribution over \([N]^k\), with marginal distribution \( R_i \) on the \( i \)th coordinate. Let \( D \) be a distribution over \([N]^k\). Then

\[
\sum_{i=1}^{k} D_{KL}(R_i||D) \leq D_{KL}(R||D^k)
\]

3 Main Result

Let \( S = \{D_x\}_x \) 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 \( D := D_x \) be the corresponding probability distribution over \( \{0,1\}^m \). Let \( p_y := \Pr[D][y] \) be the probability of \( y \). We now define the search problem \( R \). Let \( N := m/\delta^2 \), 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

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

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 \( C = C_x \) be any distribution over \( \{0,1\}^m \) such that \( \|C - D\| \leq \varepsilon \). Then

\[
\Pr_{Y \sim C^N}[Y \notin A_{x,\delta}] \leq \varepsilon N + \frac{b}{2^\beta}.
\]
Proof. We have
\[
\Pr_{Y \sim \mathcal{C}^N} [Y \notin A_{x,\delta}] \leq \Pr_{Y \sim \mathcal{D}^N} [Y \notin A_{x,\delta}] + \|\mathcal{C}^N - \mathcal{D}^N\|
\]
\[
\leq \Pr_{Y \sim \mathcal{D}^N} [Y \notin A_{x,\delta}] + \varepsilon N.
\]
So it suffices to consider a $Y$ drawn from $\mathcal{D}^N$. By Lemma 14,
\[
\Pr_{Y \sim \mathcal{D}^N} \left[ K(Y \mid x, \delta) \geq \log_2 \frac{1}{p_{y_1} \cdots p_{y_N}} - \beta \right] \geq 1 - \frac{b}{2^\beta}
\]
Therefore
\[
\Pr_{Y \sim \mathcal{D}^N} [Y \notin A_{x,\delta}] \leq \frac{b}{2^\beta},
\]
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. Suppose that
\[
\Pr[B(x, 0^{1/\delta}) \in A_{x,\delta}] \geq 1 - \delta,
\]
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 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
\[
\|\mathcal{C} - \mathcal{D}\| \leq \delta + Q_B \sqrt{\beta \frac{\beta}{N}}.
\]
Proof. Let $\mathcal{R}'$ be a distribution that is identical to $\mathcal{R}$, except that we condition on $B(x, 0^{1/\delta}) \in A_{x,\delta}$. Then by hypothesis, $\|\mathcal{R} - \mathcal{R}'\| \leq \delta$. Now let $\mathcal{C}'_i$ be the marginal distribution of $\mathcal{R}'$ on the $i^{th}$ coordinate, and let
\[
\mathcal{C}' = \frac{1}{N} \sum_{i=1}^N \mathcal{C}'_i
\]
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,
\[
\|\mathcal{C} - \mathcal{D}\| \leq \|\mathcal{C} - \mathcal{C}'\| + \|\mathcal{C}' - \mathcal{D}\|
\]
\[
\leq \delta + \|\mathcal{C}' - \mathcal{D}\|,
\]
and it suffices to upper-bound $\|\mathcal{C}' - \mathcal{D}\|$.

Let $q_Y := \Pr_{\mathcal{R}'} [Y]$. Then by Lemma 13,
\[
K(Y \mid x, \delta) \leq \log_2 \frac{1}{q_Y} + K(\mathcal{R}') + O(1)
\]
9
for all $Y \in (\{0, 1\}^m)^N$. Also, since $Y \in A_{x, \delta}$, by assumption we have

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

Combining,

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

This implies the following upper bound on the KL-divergence:

$$D_{KL}(\mathcal{R}' \mid \mathcal{D}) = \sum_{Y \in ((0,1)^m)^N} q_Y \log_2 \frac{q_Y}{p_{y_1} \cdots p_{y_N}} \leq \max_Y \log_2 \frac{q_Y}{p_{y_1} \cdots p_{y_N}} \leq K(\mathcal{R}') + O(1) + \beta.$$  

So by Proposition 16,

$$\sum_{i=1}^{N} D_{KL}(\mathcal{R}'_i \mid \mathcal{D}) \leq D_{KL}(\mathcal{R}' \mid \mathcal{D}^N) \leq K(\mathcal{R}') + O(1) + \beta,$$

and by Proposition 15,

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

So by Cauchy-Schwarz,

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

Hence

$$\|\mathcal{C}' - \mathcal{D}\| \leq \sqrt{\frac{2\beta + 2K(\mathcal{R}') + O(1)}{N}},$$

and

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

for some constant $Q_B$ depending on $B$. ■

By combining Lemmas 17 and 18, we can now prove Theorem 5: that for any sampling problem $S = (D_x)_{x \in \{0, 1\}^n}$ (where $D_x$ is a distribution over $m = m(n)$-bit strings), there exists a search problem $R_S = (A_x)_{x \in \{0, 1\}^n}$ that is “equivalent” to $S$ in the following two senses.
(i) Let $O$ be any oracle that, given $\langle x, 0^{1/\varepsilon}, r \rangle$ as input, outputs a sample from a distribution $C_x$ such that $\|C_x - D_x\| \leq \varepsilon$, as we vary the random string $r$. Then $R_S \in \text{FBPP}^O$.

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

**Proof of Theorem 5 (Sampling/Searching Equivalence Theorem).** For part (i), given an input $\langle x, 0^{1/\delta} \rangle$, suppose we want to output an $N$-tuple $Y = \langle y_1, \ldots, y_N \rangle \in \{\{0, 1\}^m\}^N$ 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 $\varepsilon := \frac{\delta}{2N} = \frac{\delta^2.1}{2m}$.
2. Call $O$ on inputs $\langle x, 0^{1/\varepsilon}, r_1 \rangle, \ldots, \langle x, 0^{1/\varepsilon}, r_N \rangle$, 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 $\text{poly} (n, 1/\delta)$ time. Furthermore, by Lemma 17, its failure probability is at most

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

For part (ii), given an input $\langle x, 0^{1/\delta} \rangle$, suppose we want to sample from a distribution $C_x$ such that $\|C_x - D_x\| \leq \varepsilon$. 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 $\langle x, 0^{1/\delta} \rangle$, 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 $C_x$.

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

$$\|C_x - D_x\| \leq \delta + Q_B \sqrt{\frac{\beta}{N}} \leq \frac{\varepsilon}{2} + Q_B \sqrt{\frac{\varepsilon^2.1}{m} (2 + \log \frac{1/\varepsilon}{m})},$$

for some constant $Q_B$ depending only on $B$. So in particular, there exists a constant $C_B$ such that $\|C_x - D_x\| \leq \varepsilon$ for all $m \geq C_B$. For $m < C_B$, we can simply hardwire a description of $D_x$ 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 $S \in \text{SampP}$ if and only if $R_S \in \text{FBPP}$, and likewise $S \in \text{SampBQP}$ if and only if $R_S \in \text{FBQP}$, and so on for any model of computation that is “below recursive” (i.e., simulable by a Turing machine) and has the extremely simple closure properties used in the proof.
3.1 Implication for Quantum Computing

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

**Proof of Theorem 6.** First, suppose $\text{SampP} = \text{SampBQP}$. Then consider a search problem $R = (A_x)_x$ in $\text{FBQP}$. 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 $D_{x,\delta}$ be the probability distribution over $y$’s output by $Q$ on input $\langle x, 0^{1/\delta} \rangle$. Then to solve $R$ in $\text{FBPP}$, clearly it suffices to sample approximately from $D_{x,\delta}$ in classical polynomial time. But we can do this by the assumption that $\text{SampP} = \text{SampBQP}$.\footnote{As mentioned in Section 1, the same argument shows that $\text{SampP} = \text{SampBQP}$ (or equivalently, $\text{FBPP} = \text{FBQP}$) implies $\text{BPP} = \text{BQP}$. However, the converse is far from clear: we have no idea whether $\text{BPP} = \text{BQP}$ implies $\text{SampP} = \text{SampBQP}$.}

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

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

Theorem 6 is easily seen to relativize: for all oracles $A$, we have $\text{SampP}^A = \text{SampBQP}^A$ if and only if $\text{FBPP}^A = \text{FBQP}^A$. (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.)

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 $S \not\in \text{SampP}^L$, or else there exists an oracle $O$ solving $S$ such that $L \not\in \text{BPP}^O$? What if we require the oracle $O$ 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 $S = (D_x)_{x \in \{0,1\}^*}$ 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 $C_X$ such that $\|C_X - D_x\| \leq \varepsilon$. Also, if $x$ is an $n$-bit string, define the length of $X = \langle x, 0^{1/\varepsilon} \rangle$ to be $\ell(X) := n + 1/\varepsilon$.\footnote{As mentioned in Section 1, the same argument shows that $\text{SampP} = \text{SampBQP}$ (or equivalently, $\text{FBPP} = \text{FBQP}$) implies $\text{BPP} = \text{BQP}$. However, the converse is far from clear: we have no idea whether $\text{BPP} = \text{BQP}$ implies $\text{SampP} = \text{SampBQP}$.}
We now define a language $L_S \subseteq \{0,1\}^\ast$. For all $n$, let $M_{t(n)}$ be the lexicographically first $M_t$ that succeeds on all inputs $X$ such that $\ell(X) \leq n$. 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 $n^{t(n)}$ steps when run on a blank tape.

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

**Proof.** First suppose $S \in \text{SampP}$. 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 $L_S \in \text{P}$.

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

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 $S \in \text{SampP} \iff L_S \in \text{P}$:

“Take $L_S$ to be the empty language if $S \in \text{SampP}$, or an EXP-complete language if $S \notin \text{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 $\text{SampP} = \text{SampBQP}$ if and only if $\text{FBPP} = \text{FBQP}$. If we attempted to prove similar consequences from Proposition 19, then we would end up with a different language $L_S$, depending on whether our starting assumption was $S \in \text{SampP}$, $S \in \text{SampBQP}$, 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 $O$ solving $S$ such that $R \notin \text{FBPP}^O$, or there exists an oracle $O$ solving $R$ such that $S \notin \text{SampP}^O$? 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 \( S_R = \{D_x\}_x \) that is “almost equivalent” to \( R \), in the following senses.

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

(ii) If \( B \) is any probabilistic Turing machine solving \( R \), then there exists a constant \( \eta_B > 0 \) such that a \( \text{Samp}^B \) machine can sample from a probability distribution \( C_x \) with \( \|C_x - D_x\| \leq 1 - \eta_B \).

Proof. Let \( U_x \) 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 \( D_x \) be the distribution obtained by drawing \( y \sim U_x \) and then conditioning on the event \( y \in A_x \). (Note that \( D_x \) is well-defined, since \( U_x \) assigns nonzero probability to every \( y \).)

For (i), notice that \( D_x \) has support only on \( A_x \). So if we can sample a distribution \( C_x \) such that \( \|C_x - D_x\| \leq \varepsilon \), then certainly we can output an element of \( A_x \) with probability at least \( 1 - \varepsilon \).

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

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

\[
q_y \leq g_B \cdot 2^{-K(y|x)}
\]

for all \( y \in A_x \). On the other hand, let \( p_y := \Pr_{D_x}[y] \) and \( u_y := \Pr_{U_x}[y] \). Then there exists a constant \( \alpha \geq 1 \) such that \( p_y = \alpha u_y \) if \( y \in A_x \) and \( p_y = 0 \) otherwise. So

\[
p_y \geq u_y \geq c \cdot 2^{-K(y|x)}
\]

for all \( y \in A_x \). Hence \( p_y \geq \frac{\alpha}{g_B} q_y \), so

\[
\|C' - D_x\| = \sum_{y \in A_x : p_y < q_y} |p_y - q_y| \\
\leq 1 - \frac{c}{g_B}.
\]

Therefore

\[
\|C_{x,\delta} - D_x\| \leq \|C_{x,\delta} - C'\| + \|C' - D_x\| \\
\leq 1 - \frac{c}{g_B} + \delta,
\]

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 \( \|C_x - D_x\| \leq 1 - \eta_B \) replaced by \( \|C_x - D_x\| \leq \varepsilon \) (in other words, with \( S_R \in \text{Samp}^B \)).

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 $O$ is any oracle solving $S$, then $R_S \in \text{FBPP}^O$.

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

(iii) There exists a $\text{SPACE}(f(n) + n^{O(1)})$ 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 $K_{\text{SPACE}(T)}(y)$ 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 $K(Y | x, \delta)$ by the space-bounded complexity $K_{\text{SPACE}(T)}(Y | x, \delta)$. 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 $D = \{p_y\}$ be any distribution over strings that is samplable in $\text{BPTIME}(f(n))$, and let $y$ be any element in the support of $D$. Then there exists a constant $C_D$, depending on $D$, such that

$$K_{\text{SPACE}(T)}(y) \leq \log_2 \frac{1}{p_y} + C_D.$$ 

The proof is simply to notice that, in $\text{SPACE}(f(n) + n^{O(1)})$, we can compute the probability $p_y$ of every $y$ in the support of $D$, 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 $\text{BPTIME}(f(n))$ 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 $\text{SPACE}(f(n) + n^{O(1)})$ by $\text{CH}(f(n) + n^{O(1)})$ (where $\text{CH}$ denotes the counting hierarchy), but can we go further? It seems unlikely that one could check in $\text{NP}$ (or $\text{NTIME}(f(n) + n^{O(1)})$) 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, Dana Moshkovitz for pointing me to Proposition 16 from [7], and the anonymous reviewers for their comments.

## References

[1] S. Aaronson and A. Arkhipov. The computational complexity of linear optics. In *Proc. ACM STOC*, 2011. To appear. 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.

[7] A. Rao. Parallel repetition in projection games and a concentration bound. In Proc. ACM *STOC*, pages 1–10, 2008. ECCC TR08-013.

[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.