An Efficient Coding Theorem via Probabilistic Representations and Its Applications

Zhenjian Lu
University of Warwick, Coventry, UK

Igor C. Oliveira
University of Warwick, Coventry, UK

Abstract

A *probabilistic representation* of a string $x \in \{0,1\}^n$ is given by the code of a randomized algorithm that outputs x with high probability (Oliveira, ICALP 2019, [30]). We employ probabilistic representations to establish the first unconditional Coding Theorem in *time-bounded* Kolmogorov complexity. More precisely, we show that if a distribution ensemble $\mathcal{D}_m$ can be uniformly sampled in time $T(m)$ and generates a string $x \in \{0,1\}^*$ with probability at least δ, then x admits a time-bounded probabilistic representation of complexity $O(\log(1/\delta) + \log(T) + \log(m))$. Under mild assumptions, a representation of this form can be computed from x and the code of the sampler in time polynomial in $n = |x|$.

We derive consequences of this result relevant to the study of data compression, pseudodeterministic algorithms, time hierarchies for sampling distributions, and complexity lower bounds. In particular, we describe an *instance-based* search-to-decision reduction for Levin's Kt complexity (Levin, Information and Control 1984, [23]) and its probabilistic analogue rKt [30]. As a consequence, if a string x admits a succinct time-bounded representation, then a near-optimal representation can be generated from x with high probability in polynomial time. This partially addresses in a time-bounded setting a question from [23] on the efficiency of computing an optimal encoding of a string.

2012 ACM Subject Classification Theory of computation

Keywords and phrases computational complexity, randomized algorithms, Kolmogorov complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.94

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2021/041/` [28]

Acknowledgements We are grateful to Lijie Chen for a suggestion that allowed us to improve the list size in Corollary 3. We also thank Rahul Santhanam and Rahul Ilango for discussions on search-to-decision reductions, and James Maynard for an email exchange on number-theoretic techniques and the problem of generating primes. Igor would like to thank Michal Koucký for asking a question about the probability threshold in the definition of rKt during the Dagstuhl Seminar "Computational Complexity of Discrete Problems" (2019). This work received support from the Royal Society University Research Fellowship URF\R1\191059.

1 Introduction

Shannon's information theory provides a foundation for the study of data transmission and data compression. However, it inherently considers *probability distributions* and *random variables*, and for this reason it does not apply to an *individual* object. The theory of Kolmogorov complexity on the other hand captures the information or computational content of an individual string or message. Despite significant conceptual differences between the two theories, results obtained in one setting sometimes admit an analogue in the other (see e.g. the textbooks [12, 24, 34]).

A fundamental result connecting the distributional framework of Shannon and the information of an individual object x is the *Coding Theorem* in Kolmogorov complexity [22].[1] This theorem states that if a randomized machine A generates a string x with probability δ, then its Kolmogorov complexity $\mathsf{K}(x)$ is at most $\log(1/\delta) + O_A(1)$. The result is part of a deep and beautiful theory that we will not be able to survey here. For a complexity-theoretic perspective that is closer to our work, we refer to [21], where the coding theorem is referred to as one of the four pillars of Kolmogorov complexity.

An issue with this result and with Kolmogorov complexity more broadly is that many aspects of the theory are *nonconstructive*. For instance, computing or even estimating $\mathsf{K}(x)$ of an input string x is known to be undecidable. This limits the applicability of Kolmogorov complexity and of the aforementioned coding theorem in algorithms and complexity theory.

In order to import methods of Kolmogorov complexity from computability to complexity theory, a number of works have introduced *time-bounded* variants of Kolmogorov complexity (cf. [1, 2, 13, 3] for a survey of results). In other words, one considers the minimum description length of a string x with respect to machines that operate *under a time constraint.* Among many applications, this idea has led Sipser [35] to a proof that BPP is contained in the polynomial hierarchy, and Levin [23, Section 1.3] to further develop universal search and optimal search algorithms.

Naturally, many authors have investigated time-bounded variants of the main results in Kolmogorov complexity.[2] Unfortunately, under standard hardness assumptions some of its most powerful theorems do not survive in time-bounded settings. Two notable examples are the language compression theorem (see [8] and references therein) and the principle of symmetry of information (see [26, 27]).

Our focus in this work is on the coding theorem and its applications. Interestingly, there is no barrier to proving certain versions of the coding theorem in a time-bounded setting. For instance, Fortnow and Antunes [5] have implicitly established a form of this result, *under a strong computational assumption.* While their results are conditional and currently beyond reach without an assumption, they indicate that a useful time-bounded version of the coding theorem might be true.

Before proceeding with our discussion, we recall Levin's time-bounded Kolmogorov complexity. Let M be a deterministic machine and $|M|$ be its description length. For a string $x \in \{0,1\}^*$,

$$\mathsf{Kt}(x) \overset{\text{def}}{=} \min_{\text{TM } M,\, t \geq 1} \{|M| + \log t \mid M(\varepsilon) \text{ outputs } x \text{ in } t \text{ steps}\},$$

where $M(\varepsilon)$ denotes the computation of M over the empty string.[3] An important advantage of this definition over Kolmogorov complexity $\mathsf{K}(x)$ is that an encoding of minimal Kt complexity can be computed in exponential time via exhaustive search. Moreover, given an encoding w of x of Kt complexity at most k, we can recover x from w in time at most 2^k. Beyond its established applications in topics such as optimal search algorithms and pseudorandomness, Levin's Kt complexity plays an important role in a few other areas. Recent examples include hardness magnification (e.g. [31, 10]) and the investigation of non-disjoint promise problems [18].

[1] Some authors also refer to it as the Source Compression Theorem or Source Coding Theorem.

[2] Troy Lee's PhD thesis [21] contains a particularly nice exposition of the contrast between Kolmogorov complexity and its time-bounded variants, including pointers to relevant references.

[3] To obtain precise results about the encoding lengths, it is necessary to fix a universal machine and to consider short inputs for this machine that produce x, as done by Levin. Since our techniques are not sensitive to encoding choices and our results incur a constant factor overhead in the encoding length, this is not relevant for our discussion.

A *probabilistic version* of Levin's Kt complexity has been recently investigated in [30]. In this definition, we minimize over *randomized* machines that output x with probability at least $2/3$ after computing for at most t steps. In other words,

$$\mathsf{rKt}(x) \stackrel{\text{def}}{=} \min_{\text{RTM } M,\, t \geq 1} \{|M| + \log t \mid M(\varepsilon) \text{ outputs } x \text{ in } t \text{ steps with probability} \geq 2/3\}.$$

Consequently, an rKt description w of a string x provides a *probabilistic representation* of x, in the sense that x can be recovered with high probability from w. (Note that the description itself is a deterministic object.)

Under a mild derandomization assumption, $\mathsf{Kt}(x) = \Theta(\mathsf{rKt}(x))$ for every string x [30, Theorem 5]. If so, this would show that any object that can be succinctly described in a probabilistic way can also be succinctly described in a deterministic way, and that results about rKt can be transferred to Kt. However, we appear to be far from establishing this relation, and it is consistent with our knowledge that there is an infinite sequence $\{x_n\}_{n\geq 1}$ where each x_n is an n-bit string satisfying $\mathsf{rKt}(x) = O(\log n)$ and $\mathsf{Kt}(x) = \Omega(n)$.

We do know without assumptions that various natural objects such as certain n-bit prime numbers can have rKt complexity $n^{o(1)}$ [30, 32], while establishing even an upper bound of $o(n)$ on the Kt complexity of a sequence of n-bit primes would lead to a breakthrough in the deterministic generation of primes [37]. It seems therefore that probabilistic representations are useful to represent data, given our current knowledge of algorithms and complexity.

Another interesting aspect of rKt is that, despite its conjectured equivalence to Kt, we are able to settle basic questions about rKt that remain longstanding conjectures in the case of Kt. For instance, a natural computational problem is to approximate, given a string x, the value $\mathsf{rKt}(x)$. In other words, we would like to decide if a string has a short probabilistic representation. [30] proved *unconditionally* that this problem cannot be solved in probabilistic polynomial time. In contrast, showing that computing Kt cannot be done in deterministic polynomial time (i.e., proving $\mathsf{MKtP} \notin \mathsf{P}$) is an important problem in time-bounded Kolmogorov complexity (cf. [4]).

We are motivated by these intriguing recent results and by the possibility of studying algorithmic information theory from the vantage point of probabilistic descriptions. The following questions are particularly relevant in this context.

1. Is it possible to employ rKt to establish new results in time-bounded Kolmogorov complexity? Specifically, can we establish an *unconditional* version of the coding theorem discussed above?
2. We have natural examples where probabilistic representations might be helpful, but no positive algorithmic results about *finding* such descriptions. Is it possible to compute a probabilistic representation of a string in some non-trivial way?
3. Can we further develop the theory of probabilistic representations and show stronger results for natural objects, such as prime numbers?

1.1 Contributions

We make progress in the context of these questions, obtaining results that suggest new research directions connected to data compression, time-bounded Kolmogorov complexity, search-to-decision reductions, and related areas. In particular, our results highlight the usefulness of *probabilistic representations* in algorithms and complexity. We describe these contributions in detail next.

1.1.1 Main results

For an algorithm or machine A, recall that we use $|A|$ to denote its description length under a fixed encoding scheme. One of our main contributions in this work is to show that an important result in Kolmogorov complexity survives in the time-bounded setting.

▶ **Theorem 1** (Efficient Coding Theorem for rKt). *Suppose there is a randomized algorithm A for sampling strings such that $A(1^m)$ runs in time $T(m)$ and outputs a string $x \in \{0,1\}^*$ of length $n \leq T(m)$ with probability at least $\delta > 0$. Then*

$$\mathsf{rKt}(x) \;=\; O\big(\log(1/\delta) + \log(T(m)) + \log(m)\big),$$

where the constant behind the $O(\cdot)$ depends on $|A|$ and is independent of the remaining parameters. Moreover, given x, m, the code of A, and δ, it is possible to compute with probability ≥ 0.99 some rKt *encoding of x of at most this complexity in time* $\mathsf{poly}(|A|, \log(m), |x|, \log(1/\delta))$.

A few comments are in order. It is not hard to prove an effective coding theorem for a distribution whose *cumulative probability function* is computable in polynomial time (cf. [24]). Other works have established incomparable results under the assumption that deciding if a string is in the support of the distribution is easy (see e.g. [39]). Crucially, the theorem above only assumes that the distribution is *samplable*, which makes it more broadly applicable. To our knowledge, Theorem 1 is the first result that provides a general approach to constructing a probabilistic encoding of a string.

It follows from Shannon's Coding Theorem that the expected encoding length in Theorem 1 is essentially optimal up to a constant factor. An interesting feature of the result is that the algorithm computing the rKt encoding runs in polynomial time regardless of the time complexity of the sampler $A(1^m)$. Furthermore, producing the encoding of a string x only requires knowledge of a lower bound on its probability weight, in contrast to encoding algorithms such as Huffman coding where knowledge of the probabilities of all elements in the support of the distribution is necessary.

Using an argument of Levin (see [16] and [21, Section 5.3] or [28, Section A.3]), under the existence of one-way functions there is a polynomial-time samplable source $\mathcal{D}_n$ supported over $\{0,1\}^n$ such that every $x \in \mathsf{Support}(\mathcal{D}_n)$ has probability weight $\mathcal{D}_n(x) \geq 2^{-n^{\varepsilon}}$, but $\mathcal{D}_n$ does not admit a pair $(\mathsf{Enc}_n, \mathsf{Dec}_n)$ of efficient (probabilistic) encoding and decoding algorithms such that each $x \in \mathsf{Support}(\mathcal{D}_n)$ is assigned a description of length $\leq n - 3$. In contrast, Theorem 1 is able to sidestep this limitation by using an efficient encoding algorithm whose associated decoding procedure (provided by the rKt representation itself) is not necessarily efficient. There are applications where this trade-off might be acceptable, i.e., where it is crucial to achieve high compression rates and fast (or low energy) encoding, but for which decoding is allowed to take more resources.[4]

Similarly to the coding theorem in Kolmogorov complexity, it is possible to derive a variety of consequences from Theorem 1. We cover in more detail one of its most unexpected implications, deferring the discussion of a couple of other results to Section 1.1.2 below.

A *search-to-decision reduction* is an efficient procedure that allows one to find solutions to a problem from the mere ability to decide when a solution exists. These reductions are particularly important in algorithms and complexity. On the one hand, theories of computational complexity are often easier to develop in the context of *decision problems.*

[4] As a speculative scenario, one can imagine data transmission for deep space exploration.

On the other hand, in practice solving a *search problem* tends to be the relevant task. A search-to-decision reduction for a given problem shows that the complexities of its search and decision versions are similar.

It is well known that any NP-complete problem admits a search-to-decision reduction. However, there are problems of interest for which such reductions are still unknown. A notable example in the realm of time-bounded Kolmogorov complexity is whether MCSP, the Minimum Circuit Size Problem, admits a search-to-decision reduction (cf. [19] for a discussion and a recent result).

We establish the existence of the following search-to-decision reductions for rKt and Kt complexities.

▶ **Theorem 2** (Instance-based search-to-decision reductions). *The following results hold:*

(i) *There is a randomized polynomial-time algorithm that, when given an input string $x \in \{0,1\}^n$ and a value $k \geq \mathsf{rKt}(x)$, outputs with probability ≥ 0.99 a valid* rKt *representation of x of complexity $O(k)$.*

(ii) *Similarly, there is a randomized polynomial-time algorithm that, when given an input string $x \in \{0,1\}^n$ and a value $k \geq \mathsf{Kt}(x)$, outputs with probability ≥ 0.99 a valid* Kt *representation of x of complexity $O(k)$.*

We note that [4] established that any language in deterministic exponential time E can be non-uniformly reduced via polynomial-size circuits to the problem of deciding Kt complexity (or even of just approximating Kt on a large fraction of inputs). Since the problem of finding a minimal Kt representation of an input string can be encoded as a language in E, it follows from their work that there is a polynomial-size search-to-decision reduction for Kt. Observe that the reduction given by [4] finds a description of minimum length, while Theorem 2 only provides a description that is within a constant factor of the optimal description length.

There are now a number of search-to-decision reductions in the context of time-bounded Kolmogorov complexity with respect to a variety of string complexity measures (e.g. [9, 17, 19, 20, 25]). We are not aware, however, of a previous *instance-based* search-to-decision reduction in the sense of Theorem 2. In other words, Theorem 2 shows that it is possible to produce a near-optimal Kt representation of x from a decision oracle for Kt *that is only queried on x*.[5] In contrast, the aforementioned reductions require an oracle to the decision problem that is correct on all or at least on a large fraction of inputs. More broadly, search-to-decision reductions for problems in other domains such as circuit satisfiability and graph theory tend to query inputs that modify the original input x.

As a result of the property described above, we are able to derive consequences from Theorem 2 that do not follow from other reductions. Unsurprisingly, our approach employs significantly different techniques compared to the papers cited above. In particular, it departs from the PRG-based approach of [4] and of a few other subsequent works.

We elaborate next on some aspects of Theorem 2 that make it particularly interesting, at least to the authors. Note that Kt (similarly for rKt and probabilistic algorithms) is a *universal* complexity measure for data compression, in the following precise sense. If a string x can be encoded/decoded by *some* uniform compression scheme in time $\leq T$ and using a description of length $\leq \log T$, then its Kt complexity is at most $O(\log T)$. As a consequence, compressing a string x to $O(\mathsf{Kt}(x))$ bits is not far from optimal *in a strong sense.*

[5] A binary search is sufficient to compute $\mathsf{rKt}(x) \in \mathbb{N}$ from a decision oracle that checks if $\mathsf{rKt}(x) \leq \gamma$ for a given threshold γ.

Now suppose we have a string x of length n, and we estimate its Kt complexity to be at most, say, $O(\sqrt{n})$. Then finding a valid Kt representation of this complexity via *exhaustive search* would take time $2^{O(\sqrt{n})}$. Assuming the widely believed hardness of estimating the Kt complexity of a string (which is provably true for rKt by [30]), one is tempted to conjecture that nothing better can be done. Still, Theorem 2 tells us that there is an algorithm that can produce a valid Kt representation of x of complexity $O(\sqrt{n})$ in time just $\mathsf{poly}(n)$.[6] This result addresses to some extent in the time-bounded setting a question from [23, Page 5] on the efficiency of generating programs of length close to the optimal description length.

It is unclear to us whether there are *deterministic* search-to-decision reductions for Items (i) and (ii) of Theorem 2. Although this requires more investigation, it is possible that these reductions provide natural examples of randomized algorithms that cannot be replaced by deterministic ones with only a polynomial overhead.[7]

1.1.2 Further applications and open problems

Efficient construction of time-bounded programs and complexity lower bounds for estimating rKt. By executing the efficient search-to-decision reduction from Theorem 2 with all values of $k = O(n)$ that are powers of 2, the following corollary is immediate.

▶ **Corollary 3** (Effective short lists with short programs in short time). *Given an arbitrary string x of length n, it is possible to compute with high probability and in polynomial time a collection of at most $d = \log(n) + O(1)$ strings $w_1, \ldots, w_d$ such that at least one of these strings is a valid* rKt *encoding of x of complexity $O(\mathsf{rKt}(x))$.*

The same result can be obtained for Kt complexity. Corollary 26 should be contrasted with the results from [7, 6] in the context of (time-unbounded) Kolmogorov complexity. They achieve optimal compression rates, by mapping a string of Kolmogorov complexity k to a collection of strings that contains a valid representation of length $k + O(1)$. While we are not able to achieve this level of compression, our setting is more stringent, since we need to construct a short representation that can be decompressed under a time constraint. It would be interesting to explore connections between our techniques and those employed in Kolmogorov complexity to see if our parameters can be further improved.

In light of Corollary 3 and the complexity lower bound for estimating the rKt complexity of a string ([30]; see Theorem 14), it follows that the intractability of estimating rKt does not lie in the exhaustive search required to *find* a succinct representation. Instead, the hardness is a consequence of the intractability of *checking* if a given rKt representation is valid for a string x.

It is unlikely that circuit minimization of a given truth-table (MCSP) admits a search-to-decision reduction of the form given by Theorem 2. This would allow one to construct in time polynomial in the size of the input truth-table a collection of circuits that contains a circuit of near-optimal size for the truth-table. As opposed to rKt, checking if a circuit correctly encodes a string can be done in polynomial time, which implies that such a search-to-decision reduction provides a natural property in the sense of [33]. Consequently, under cryptographic assumptions, minimizing circuit size and minimizing Kt/rKt behave differently with respect to instance-based search-to-decision reductions.

[6] This does not contradict the intractability result from [30]. Indeed, it implies that *checking* if a given representation generates a particular string is the problem that requires super-polynomial time.

[7] Note that this is not inconsistent with $\mathsf{P} = \mathsf{BPP}$ because these are not decision problems.

Hardness of approximately sampling distributions. It is not hard to show that if Promise-BPP $\subseteq$ Promise-P then BPTIME$[\cdot]$ admits a time hierarchy theorem. This easily follows from the deterministic time hierarchy theorem for decision problems. As a consequence of our results, we observe that a similar derandomization hypothesis for *decision problems* implies a strong time hierarchy theorem for *sampling distributions*.

To our knowledge, a uniform time hierarchy theorem for sampling distributions was first established in [41]. While the results of his work are *unconditional*, [41] left open the problems of proving an *almost-everywhere* lower bound and of achieving a *larger statistical gap* [41, Section 5]. Our next result provides a conditional solution to these questions.

▶ **Theorem 4** (A strong time hierarchy theorem for sampling distributions). *Under the assumption that* Promise-BPE $\subseteq$ Promise-E*, there is a constant $\zeta > 0$ for which the following holds. Let $n^{1/\zeta} \leq T(n) \leq 2^n$ be any constructive time bound. There is an ensemble $\{\mathcal{D}_n\}_{n\geq 1}$ of distributions $\mathcal{D}_n$ such that:*

(i) *Each distribution $\mathcal{D}_n$ is supported over a single string $z_n \in \{0,1\}^n$.*
(ii) *There is a deterministic algorithm $A(1^n)$ that samples $\mathcal{D}_n$ and runs in time $O(T(n))$.*
(iii) *For each randomized algorithm $B(1^n)$ that runs in time $O(T(n)^\zeta)$ and for every large enough n, the statistical distance of $\mathcal{D}_n$ and $B(1^n)$ is at least $1 - 1/T(n)^\zeta$.*

We are not aware of a previous time hierarchy theorem for sampling distributions able to produce hard distributions of support size one, even under computational assumptions. (Interestingly, the unconditional results of [41] hold for support size $k \geq 2$.) Note that Theorem 4 exploits *uniformity* in a crucial way: each distribution $\mathcal{D}_n$ can be sampled by a (non-uniform) linear-size circuit C_n that ignores its random input and outputs the unique string z_n in the support of $\mathcal{D}_n$.

Computational patterns in prime numbers. We now step back and revisit one of the most basic questions associated with the power of probabilistic representations. The problem stated below is concerned with the computational (in)compressibility of n-bit prime numbers.

▶ **Problem 5** (Primes with short descriptions). *Is there an infinite sequence $\{p_n\}_{n\geq 1}$ of prime numbers $p_n \in [2^{n-1}, 2^n - 1]$ such that* rKt$(p_n) = o(n)$*?*

This would show that there are primes of *every* large length that admit effective short encodings, i.e., such primes possess computational patterns that translate into effective representations (e.g. Mersenne primes). A partial result appears in [32], where this is proved for *infinitely* many n. Consequently, some primes can have short descriptions. *Is this a rare phenomenon, or does it happen for primes of all lengths?* This is the question captured by Problem 5.

We note that obtaining a solution to Problem 5 is necessary before showing the existence of a deterministic algorithm that generates n-bit primes in time $2^{o(n)}$. We refer to [37] for more background on this problem.

Theorem 1 offers a path to solving this problem. In other words, it shows that it is enough to sample n-bit numbers in time $2^{o(n)}$ in a way that assigns enough weight to some (possibly unknown) prime. Given that many advances to our understanding of prime numbers employ *probabilistic ideas* (see e.g. [38, 29] and references therein), this perspective could be fruitful.

Our last result shows that the existence of a sampler of this form is in fact *equivalent* to a positive solution to Problem 5.

▶ **Theorem 6** (Equivalence between faster samplability and improved time-bounded descriptions). *The following statements are equivalent:*

(i) Sampling Algorithm. *For every $\varepsilon > 0$, there is a randomized algorithm $A(1^n)$ sampling strings in $\{0,1\}^*$ that runs in time $T(n) = O(2^{\varepsilon n})$ and for which the following holds. For every large n, there is an n-bit prime q_n such that* $\Pr[A(1^n)$ *outputs* $q_n] \geq 2^{-\varepsilon n}$.

(ii) Short Descriptions. *Let $\delta > 0$ be an arbitrary constant. For every large n, there is an n-bit prime p_n such that* $\mathsf{rKt}(p_n) \leq \delta n$.

We stress that the equivalence in Theorem 6 is not particular to prime numbers and to exponential time bounds. It can be seen as the analogue for rKt of Levin's fundamental insight that a sequence of objects (such as n-bit primes or solutions to search problems) can be deterministically generated in time $T(n)$ if and only if they have Kt complexity of order $\log(T(n))$.

Finally, complementing the applications and open problems mentioned above, it would be interesting to understand when a mathematical method employed to show the existence of certain combinatorial objects also implies the existence of objects of bounded rKt complexity. This is particularly interesting in settings where the desired objects are "rare" and the probability of producing them is small (e.g. Lovász local lemma and techniques from discrepancy theory). Extracting rKt upper bounds from existential proofs offers an alternate way of designing non-trivial algorithms for generating the corresponding objects, since it is sufficient to exhaustively search for objects of bounded description length.[8]

1.2 Techniques

In this section, we describe the main conceptual ideas behind Theorems 1 and 2. Since our goal is to establish a coding theorem in a time-bounded setting and our applications require an algorithm that produces a valid encoding in polynomial time, it is not clear if arguments employed in the context of (unbounded) Kolmogorov complexity can be adapted to our setting. For instance, it is not hard to construct an encoding given all strings in the support of the distribution and their corresponding probabilities, but we cannot assume in the time-bounded setting that this is available. Similarly, under additional assumptions on the distribution, such as the ability to compute its cumulative probability function, producing an encoding is easier. However, we are aiming for a result that applies to the larger class of samplable distributions.

Sketch of the proof of Theorem 1. We are given a randomized algorithm $A(1^m)$ that runs in time T and samples from a distribution $\mathcal{D}_m$. Fix a string $x \in \mathsf{Support}(\mathcal{D}_m)$, and assume that its probability weight $\mathcal{D}_m(x) \geq \delta$. Our goal is to (efficiently) produce a succinct probabilistic representation of x in the sense of rKt complexity. The first thing to notice is that there are at most $1/\delta$ strings $y \in \mathsf{Support}(\mathcal{D}_m)$ such that $\mathcal{D}_m(y) \geq \delta$. Let $S_\delta \overset{\text{def}}{=} \{y \in \{0,1\}^* \mid \mathcal{D}_m(y) \geq \delta\}$ be the set of such strings, which includes our target string x. We assume for simplicity of the exposition that $S_\delta \subseteq \{0,1\}^n$, where $n = |x|$ is the length of x.

Let's pretend for now that we know the set S_δ. Note that this is not really a realistic assumption, since we are aiming to output a valid rKt description of x in a number of steps that might not even allow us to sample a single string from $\mathcal{D}_m$. We will revisit this assumption later on, and argue that explicit knowledge of S_δ is not needed to produce a probabilistic representation of x.

[8] In some applications, one might need to consider *conditional* rKt complexity. The techniques employed in this work also extend in this direction.

Our current goal is to be able to identify x among the elements of S_δ, ideally with an advice string of length close to $O(\log|S_\delta|) = O(\log(1/\delta))$. A natural way to try to achieve this goal is by producing a "fingerprint" or "hash value" from x that uniquely specifies this string. In other words, we would like to have a function $h\colon S_\delta \to \{0,1\}^*$ such that $h(x) \neq h(y)$ for every $y \in S_\delta \backslash \{x\}$. Then we can identify x in S_δ using h and the value $z = h(x)$. Assuming that we know S_δ, the total description length of x would be upper bounded by roughly $|h| + |z|$, where $|h|$ is the description length of h and $|z|$ is the length of z.

This is a basic algorithmic problem, and one way to achieve this goal with a reasonable upper bound on the description length is as follows. Given an explicit polynomial-time computable error-correcting code $E\colon \{0,1\}^n \to \{0,1\}^{O(n)}$, let $T_\delta \overset{\text{def}}{=} E(S_\delta)$, i.e., each string $y' \in T_\delta$ is obtained by applying E to a string $y \in S_\delta$. Consequently, for every distinct pair $y', y'' \in T_\delta$, their relative hamming distance $d(y', y'') = \Omega(1)$. For this reason, it follows by a simple probabilistic analysis that if we *randomly project* about $O(\log(1/\delta))$ coordinates of the strings in T_δ, we are likely to produce a "fingerprint" that uniquely specifies each string. Thus to specify x in S_δ it is enough to compute $E(x)$ and to store $O(\log(1/\delta))$ random pairs (i, b_i), where $i \sim [O(n)]$ and $b_i \overset{\text{def}}{=} E(x)_i$, the i-th bit of the string $E(x)$. Following our notation from above, one can think of h as being given by the sequence of coordinates, and z by the bits obtained from the projection of $E(x)$.[9]

There are two potential issues with this approach:

(a) In order to store x's fingerprint information (h and z), we still need $O(|h| + |z|) = O(\log(1/\delta) \cdot \log(n) + \log(1/\delta))$ bits, instead of just $O(\log(1/\delta))$.

(b) We have assumed explicit knowledge of S_δ to recover x from h and z.

The first issue is of a quantitative nature, and it can be handled via standard techniques. By projecting coordinates of $E(x)$ according to a random walk on an explicit constant-degree expander graph on $O(n)$ vertices, the total description length can be reduced to $O(\log(n) + \log(1/\delta))$.

Regarding the more challenging issue (b), first notice that the argument we have described so far uses randomness only to produce h. In particular, it gives a *randomized* algorithm that with high probability produces a *deterministic* representation of x from h, $z = h(x)$, and S_δ. Therefore, we have not yet exploited the power of *probabilistic representations.*

A simple but crucial idea is that we can use the *code* of the sampler A and m to efficiently *compute a valid* rKt *representation* of x, without ever running $A(1^m)$. More precisely, the rKt *instructions* to output x include running $A(1^m)$ about $O((1/\delta) \cdot \log(1/\delta))$ times to collect (with high probability) a superset $W_\delta \supseteq S_\delta$. Let's assume for simplicity that $W_\delta = S_\delta$ (it is possible to take care of extra elements by a more delicate argument). Given the set W_δ, $n = |x|$, and an independently generated h obtained before we compute the rKt representation, h and the hash value $z = h(x)$ are likely to isolate x among the elements in W_δ. A careful implementation of these ideas allows us in randomized polynomial-time to generate an rKt representation of x whose description length is $O(\log(m) + \log(1/\delta) + \log(n))$ and whose logarithm of the running time is $O(\log(T(m)) + \log(1/\delta))$. Since generating an n-bit string x takes time $T \geq n$, the overall rKt complexity (description length + log of the running time) is $O(\log(1/\delta) + \log(T(m)) + \log(m))$. ◀

[9] Notice that an *explicit* error-correcting code together with a bounded number of random coordinates of the string $E(x)$ were used to minimize the description length of x's fingerprint. We can also generate a fingerprint by collecting the value of random XORs χ_S applied to x, but storing the relevant sets S would have been expensive.

Sketch of the proof of Theorem 2. First, we discuss a reduction for rKt. Given a string x and a parameter $k \geq \mathsf{rKt}(x)$, we need to output a valid rKt representation of x of complexity $O(k)$. This is not a lot of information, but we have a general tool at our disposal: the Coding Theorem for rKt (Theorem 1). In particular, if we could sample x in time $2^{O(k)}$ and with probability $2^{-\Omega(k)}$ using an *explicit* algorithm A, we would be done by the moreover part of this result. The only challenge is to uniformly construct an explicit sampler of this form.

Fortunately, there is a *universal* sampler U that works for all strings of rKt complexity at most k. In more detail, the sampler randomly selects the code of a randomized machine M of length at most k, simulates M with its internal randomness for at most 2^k steps, and outputs whatever is left on the output tape of M after this simulation. Using that $\mathsf{rKt}(x) \leq k$, which implies that some randomized machine of length at most k outputs x within 2^k steps with probability at least $2/3$, it is easy to see that x has probability weight at least $2^{-\Omega(k)}$ under U. Given that the code of U is explicit and we know x and k, the desired representation of x can be generated with high probability in polynomial time via Theorem 1.

We now consider a search-to-decision reduction for Kt. Recall that, under a derandomization assumption, $\mathsf{Kt}(x) = \Theta(\mathsf{rKt}(x))$ [30, Theorem 5]. Moreover, it is not hard to adapt the proof to give an efficient deterministic algorithm that converts a probabilistic representation in the sense of rKt into a deterministic one in the sense of Kt. For this reason, it is possible to show, under a plausible computational assumption, that there is an instance-based search-to-decision reduction for Kt.

The most interesting aspect of Item (ii) of Theorem 2 is that it is possible to *unconditionally* establish the result. This is obtained by a more careful investigation of the elements employed in the proofs of Theorem 1 and Theorem 2 Item (i), which reveals that the derandomization assumption is not really needed. We refer the reader to the main body of the paper for the details. ◀

Organization. The proof of Theorem 1 appears in Section 3, while the remaining results are established in Section 4. Due to lack of space, the proof of the strong time hierarchy theorem for sampling distributions (Theorem 4) appears in the full version of the paper [28].

2 Preliminaries

2.1 Basic notions

We use $|x|$ to denote the length of a binary string $x \in \{0,1\}^*$. We abuse notation and use $|M|$ denote the length of the binary encoding of a machine M with respect to a fixed universal machine. Although this will not be essential, we assume a prefix-free encoding of machines, and remark that our statements are robust with respect to encoding choices.

Probabilistic machines have an extra tape with random bits. We use $\boldsymbol{M_{\leq t}}$ to denote a random variable that represents the content of the output tape of M when it computes for t steps over the empty string ε (or its final content if the machine halts before that).

▶ **Definition 7** (rKt_δ Complexity). *For $\delta \in [0,1]$ and a string $x \in \{0,1\}^*$, we let*

$$\mathsf{rKt}_\delta(x) = \min_{M,t} \left\{ |M| + \lceil \log t \rceil \mid \Pr[\boldsymbol{M_{\leq t}} = x] \geq \delta \right\}.$$

The randomized time-bounded Kolmogorov complexity of x is given by $\mathsf{rKt}(x) \overset{\text{def}}{=} \mathsf{rKt}_{2/3}(x)$.

Levin's complexity $\mathsf{Kt}(x)$ can be defined similarly, either by taking $\delta = 1$ in the definition above or by restricting the minimization over M and t to deterministic machines. For more information about rKt, see [30].

We will assume from our encoding that for every string x of length n, $\mathsf{rKt}(x) \leq \gamma \cdot n$, where $\gamma \in \mathbb{N}$ is a universal constant. This is achieved by a trivial machine that simply stores x and prints it when given an empty string. Since printing a string x takes time at least $|x|$, we have $\mathsf{Kt}(x) \geq \mathsf{rKt}(x) \geq \log |x|$. We might implicitly use this fact to omit certain $\log n$ additive factors in our upper bounds.

These definitions and conventions are sufficient for the formalization of our results, given that the proof of Theorem 1 incurs a constant factor overhead in the resulting rKt upper bound. For the interested reader, we present a careful discussion of the parameters of Theorem 1 in the full version [28, Section A]. For more background in time-bounded Kolmogorov complexity, see [24].

For a discrete probability distribution $\mathcal{D}$, we use $\mathsf{Support}(\mathcal{D})$ to denote its support. For $x \in \mathsf{Support}(\mathcal{D})$, we let $\mathcal{D}(x)$ denote its probability weight under $\mathcal{D}$. We extend this definition to a set T in the natural way, i.e., $\mathcal{D}(T) = \sum_{x \in T} \mathcal{D}(x)$. If $\mathcal{D}$ and $\mathcal{D}'$ are distributions with support contained in a set S, their statistical distance $|\mathcal{D} - \mathcal{D}'|$ is defined as $\max_{T \subseteq [S]} |\mathcal{D}(T) - \mathcal{D}'(T)|$.

▶ **Definition 8** (Entropy). *The entropy $H(\mathcal{D})$ of a discrete probability distribution $\mathcal{D}$ is defined as*

$$H(\mathcal{D}) \;=\; \sum_{x \in \mathsf{Support}(\mathcal{D})} \mathcal{D}(x) \cdot \log \frac{1}{\mathcal{D}(x)},$$

where $\log$ *is the binary logarithm function.*

We will also require a standard application of expander graphs in order to minimize the amount of randomness in one of our constructions.

▶ **Definition 9** (Expander Graph). *An m-vertex undirected graph G is an (m, d, λ)-expander if G is d-regular and $\lambda(G) \leq \lambda$, where $\lambda(G)$ denotes the second largest eigenvalue (in absolute value) of the normalized adjacency matrix of G (i.e., the adjacency matrix of G divided by d).*

2.2 Technical tools

The version of the concentration bound appearing below can be found for instance in [40].

▶ **Theorem 10** (Chernoff Bound). *Let $X = \sum_{i=1}^{n} X_i$, where each X_i is an independent $0/1$-valued random variable. The following inequalities hold.*

(i) *For every $\gamma > 0$, if $\mu \geq \mathbf{E}[X]$ then $\mathbf{Pr}[X \geq (1+\gamma)\mu] \leq \left(\frac{e^{\gamma}}{(1+\gamma)^{(1+\gamma)}}\right)^{\mu}$.*

(ii) *For every $0 < \gamma \leq 1$, if $\mu \leq \mathbf{E}[X]$ then $\mathbf{Pr}[X \leq (1-\gamma)\mu] \leq \left(\frac{e^{-\gamma}}{(1-\gamma)^{(1-\gamma)}}\right)^{\mu}$.*

▶ **Theorem 11** (Explicit ECCs; see e.g. [36]). *There is a constant $C \in \mathbb{N}$ and a polynomial-time computable function $E_n \colon \{0,1\}^n \to \{0,1\}^{Cn}$ such that for each $n \geq 1$ and for any distinct strings $a, b \in \{0,1\}^n$, the relative hamming distance between $E_n(a)$ and $E_n(b)$ is at least $1/10$.*

We will rely on the following explicit construction of expander graphs.

▶ **Theorem 12** ((Strongly) Explicit Expander Graphs [14]). *There are constants $d \in \mathbb{N}$ and $0 < \lambda < 1$ for which the following holds. There is an (m, d, λ)-expander family $\{G_m\}$ of m-vertex graphs and a deterministic algorithm A that on inputs $m \in \mathbb{N}$, $v \in [m]$, and $i \in [d]$ outputs the i-th neighbor of v in G_m in time polynomial in $\log(m)$.*

We will also need the following well-known property of a random walk on an expander graph.

▶ **Theorem 13** (Expander Chernoff Bound [15]). *Let $G_m = (V, E)$ be an (m, d, λ)-expander, $f\colon V \to \{0,1\}$ be an arbitrary function, and $\mu \overset{\text{def}}{=} \mathbf{E}_{v\sim V}[f(v)]$. Let $v_1 \sim V$ be a uniformly chosen vertex and $v_1, \ldots, v_t$ be a random walk on G of length t. Then, for any $\alpha > 0$,*

$$\Pr_{v_1,\ldots,v_t}\left[\frac{1}{t}\sum_{i=1}^{t} f(v_i) < \mu - \alpha\right] \;\le\; e^{-(1-\lambda)\alpha^2 t/4}.$$

It is known that estimating the rKt complexity of an input string is intractable.

▶ **Theorem 14** (Hardness of estimating rKt [30]). *Let $\varepsilon \in (0,1)$ and $C \in \mathbb{N}$. There is no randomized algorithm A running in time $T(n) = O(n^{(\log n)^C})$ such that for every large enough n:*

- *For every $x \in \{0,1\}^n$ such that $\mathsf{rKt}(x) \le n^{\varepsilon}$, $\Pr_A[A(x) = 1] \ge 2/3$.*
- *For every $x \in \{0,1\}^n$ such that $\mathsf{rKt}(x) \ge n - 10$, $\Pr_A[A(x) = 0] \ge 2/3$.*

3 A Coding Theorem via Probabilistic Representations

3.1 An efficient String Isolation Lemma

The next lemma allows us to efficiently isolate a string x from a collection W of strings using a short advice string v whose length depends on the logarithm of the size of W. We follow a construction described in the proof of [11, Lemma 6.1].

▶ **Lemma 15** (String Isolation Lemma). *There is a deterministic algorithm M for which the following holds. For any set $W \subseteq \{0,1\}^n$ of size ℓ, there exists a string $u \in \{0,1\}^{O(\log(n\cdot\ell))}$ such that $M(1^n, u)$ runs in $\mathsf{poly}(n)$ time and outputs a Boolean circuit that computes a function $H\colon \{0,1\}^n \to \{0,1\}^{O(\log \ell)}$ with the following property:*

$H(w) \neq H(w')$ for every distinct pair $w, w' \in W$.

Moreover, the same guarantee is achieved by a random string u of the same length with probability at least 0.99.

Proof. Let $E_n\colon \{0,1\}^n \to \{0,1\}^{Cn}$ be the error-correcting code from Theorem 11. Moreover, let $t = c_1 \log \ell$, where c_1 is a large enough universal constant. Finally, let G_{Cn} be the expander graph from Theorem 12, where $m = Cn$.

Given a walk $\gamma = (v_1, \ldots, v_t)$ in G_{Cn}, we define the function $H_\gamma\colon \{0,1\}^n \to \{0,1\}^t$ as follows. On a string z, let $z' = E_n(z)$, and set $H_\gamma(z)_i = z'_{v_i}$, where $v_i \in [Cn]$ is given by γ and z'_j denotes the j-th bit of z'.

For distinct strings $z^1, z^2 \in \{0,1\}^n$, $E_n(z^1)$ and $E_n(z^2)$ have relative distance at least $1/10$. As a consequence, if $\boldsymbol{\gamma}$ is a length-t *random walk* on G_{Cn}, it follows from the Expander Chernoff Bound (Theorem 13) that

$$\Pr_{\boldsymbol{\gamma}}\left[H_{\boldsymbol{\gamma}}(z^1) = H_{\boldsymbol{\gamma}}(z^2)\right] \;\le\; \Pr_{\boldsymbol{\gamma}}\left[\frac{1}{t}\cdot\sum_{i=1}^{t} 1_{[H_{\boldsymbol{\gamma}}(z^1)_i \neq H_{\boldsymbol{\gamma}}(z^2)_i]} < \frac{1}{20}\right] \;\le\; e^{-\Omega(t)} \;<\; \frac{1}{100\ell^2},$$

where we have used that c_1 is a large enough constant in the definition of t. By a union bound over all distinct pairs of strings in W, there is some length-t random walk such that the corresponding function H satisfies the condition of the lemma.

Note that any length-t walk $\gamma = (v_1, \ldots, v_t)$ can be described by a string of length $\log(Cn) + O(t) = O(\log(n\cdot\ell))$, given that G_{Cn} is an m-vertex d-regular graph with $d = O(1)$ and $m = Cn$.

Finally, given 1^n and a description u of a length-t walk γ, it is possible to produce a circuit that computes as the function H_γ in time polynomial in n. This is because E_n, G_{Cn} and the walk encoded by u can be computed in time $\mathsf{poly}(n,t) = \mathsf{poly}(n, \log \ell) = \mathsf{poly}(n)$, where the last step uses that $\ell \leq 2^n$. This completes the proof of the lemma. ◀

The power of Lemma 15 comes from the fact that we don't need to know the set W, i.e., an upper bound on its size is sufficient. We remark that it is possible to achieve better parameters in Lemma 15 if we do not consider the efficiency of M. However, we need an efficient M for our applications in time-bounded Kolmogorov complexity. We also note that in our proofs we will only need that for a particular string $x \in W$ fixed in advance the value $H(x)$ is not contained in the image of H on $W \backslash \{x\}$.

3.2 An efficient Coding Theorem for rKt

We prove Theorem 1 in this section, restated below for convenience of the reader.

▶ **Theorem 16** (Efficient Coding Theorem for rKt)**.** *Suppose there is a randomized algorithm A for sampling strings such that $A(1^m)$ runs in time $T(m)$ and outputs a string $x \in \{0,1\}^*$ of length $n \leq T(m)$ with probability at least $\delta > 0$. Then*

$$\mathsf{rKt}(x) = O(\log(1/\delta) + \log(T(m)) + \log(m)),$$

where the constant behind the $O(\cdot)$ depends on $|A|$ and is independent of the remaining parameters. Moreover, given x, m, the code of A, and δ, it is possible to compute with probability ≥ 0.99 some rKt *encoding of x of at most this complexity in randomized time* $\mathsf{poly}(|A|, \log(m), |x|, \log(1/\delta))$.

Proof. Let x be a string generated with probability at least δ by a sampler $A(1^m)$. Since we only need a *lower bound* on the probability and our rKt upper bound is stated asymptotically, for representation purposes we assume without loss of generality that δ is of the form $2^{-\ell}$ for some $\ell \in \mathbb{N}$. We now show how to obtain an rKt description of x.

With the binary descriptions of m and $1/\delta$, we first run the sampler $A(1^m)$ for $t = 64 \cdot (1/\delta) \cdot \log(1/\delta)$ times to obtain a multi-set of strings S_0 of size t. Let V be the set

$$V \overset{\text{def}}{=} \{y \mid A(1^m) \text{ outputs } y \text{ with probability at least } \delta\}.$$

Note that $x \in V$ and that $|V| \leq 1/\delta$. Also, without loss of generality, we assume $\delta < 2/3$; otherwise the sampler $A(1^m)$ yields a desired rKt description of x.

▷ Claim 17. With probability at least $5/6$ (over S_0), every $y \in V$ appears in S_0 at least $\alpha = 32 \cdot \log(1/\delta)$ times.

Proof of Claim 17. Fix a $y \in V$. Note that the expected number of times that y appears in S_0 is at least

$$\mu \overset{\text{def}}{=} t \cdot \delta \;=\; 64 \cdot \log(1/\delta).$$

By the Chernoff bound (Item (*ii*) of Theorem 10), we have

$$\Pr_{S_0}[y \text{ appears in } S_0 \text{ less than } \alpha \text{ times}] \;\leq\; \left(\frac{e^{-1/2}}{(1/2)^{(1/2)}}\right)^{64 \cdot \log(1/\delta)} \;<\; \delta^6.$$

By a union bound over the strings in V, where $|V| \leq 1/\delta$, we have that the probability that there exists a $y \in V$ such that y appears in S_0 less than α times is less than $\delta^5 < 1/6$. ◁

▷ Claim 18. With probability at most $1/6$ *(*over S_0*)*, there exists a string z in S_0 such that z appears at least $\alpha = 32 \cdot \log(1/\delta)$ times in S_0 and that $A(1^m)$ outputs z with probability less than $\delta/32$.

Proof of Claim 18. Let's view S_0 as an ordered multi-set $(z_1, z_2, \ldots, z_t)$. For $z_i \in S_0$, we say "z_i is bad" if that $A(1^m)$ outputs z_i with probability less than $\delta/32$ and that it appears in S_0 at least α times. Let $\mathcal{E}$ be the event that there exists some $z_i \in S_0$ such that z_i is bad. Then we have

$$\mathbf{Pr}[\mathcal{E}] \;\le\; \sum_{i=1}^{t} \mathbf{Pr}[z_i \text{ is bad}]. \tag{1}$$

Fix an $i \in [t]$, we have

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{z_1,z_2,\ldots,z_t \sim A}[z_i \text{ is bad}] \;&=\; \sum_{\substack{z\,:\,A \text{ outputs } z\\ \text{w.p. less than } \delta/32}} \mathbf{Pr}[z_i = z \text{ AND } z_i \text{ appears in } S_0 \text{ at least } \alpha \text{ times}]\\
&\le\; \sum_{z \text{ as above}} \mathbf{Pr}[z_i \text{ appears in } S_0 \text{ at least } \alpha \text{ times} \mid z_i = z] \cdot \mathbf{Pr}[z_i = z]\\
&\le\; \max_{z \text{ as above}} \mathbf{Pr}[z_i \text{ appears in } S_0 \text{ at least } \alpha \text{ times} \mid z_i = z]\\
&\le\; \max_{z \text{ as above}} \mathbf{Pr}[z \text{ appears } S_0 \backslash \{z_i\} \text{ at least } \alpha - 1 \text{ times}]. \qquad (2)
\end{aligned}$$

Note that if for a string z, A outputs z with probability less than $\delta/32$, then the expected number of times that z appears in the multi-set $S_0 \backslash \{z_i\}$ is less than

$$\mu \overset{\text{def}}{=} (t-1) \cdot \delta/32 \;=\; 2 \cdot \log(1/\delta) - \delta/32,$$

and hence $\alpha - 1 > 11 \cdot \mu$. Then by the Chernoff bound (Item (i) of Theorem 10), we have

$$\textit{Equation } (2) \;\le\; \left(\frac{e^9}{10^{10}}\right)^{\log(1/\delta)} \;<\; \delta^{18}.$$

Therefore, we have

$$\textit{Equation } (1) \;\le\; t \cdot \delta^{18} \;<\; 64 \cdot (1/\delta)^2 \cdot \delta^{18} \;<\; 1/6,$$

as desired. ◁

Next, from S_0, we build a set S by removing every string in S_0 that appears less than $\alpha = 32 \cdot \log(1/\delta)$ times in S_0, and keeping only one copy for each of the remaining strings. Let W be the set

$$W \overset{\text{def}}{=} \{w \mid A(1^m) \text{ outputs } w \text{ with probability at least } \delta/32\}.$$

Note that $|W| \le 32/\delta$. From Claim 17 and Claim 18, we get that with probability at least $2/3$ over the choices of S_0, we obtain a "good" set S in the sense that $V \subseteq S$ (hence $x \in S$) and $S \subseteq W$.

Consider the algorithm in the String Isolation Lemma (Lemma 15), and let $n = |x| \le T(m)$. Let u be a string of length $O(\log(n \cdot |W|)) = O(\log(T(m)) + \log(1/\delta))$ such that the algorithm in Lemma 15 running on u produces a good hash function $H\colon \{0,1\}^n \to \{0,1\}^\tau$ for all length-n strings in W, where $\tau = O(\log(|W|)) = O(\log(1/\delta))$. That is, for every distinct

$w, w' \in W \cap \{0,1\}^n$, we have $H(w) \neq H(w')$. Then for every "good" set S, H maps the strings in S of length n into different buckets. Therefore, given u, n, and the hash value for x, $H(x)$, we can recover x from a good set S.

The whole algorithm runs in randomized time $\mathrm{poly}(m, T(m), 1/\delta)$ and requires an advice of length $O(\log(1/\delta) + \log(m) + \log(n))$, which gives the desired upper bound for $\mathsf{rKt}(x)$.

For the moreover part of the theorem, first observe that to output a description of x it is not necessary to run the sampling algorithm $A(1^m)$ nor to know an upper bound on its running time. We need instead the following information: the code of A, the input parameter m, the probability lower bound δ, the length $n = |x|$, the advice string u given by Lemma 15, and the hash value $H(x)$. Crucially, the proof of Lemma 15 shows that a *random* string u (encoding a random walk) satisfies the conditions of the lemma with probability ≥ 0.99. Furthermore, the same proof shows that, given u and x, we can compute $H(x)$ in time $\mathsf{poly}(n) = \mathsf{poly}(|x|)$. Therefore, by sampling a random string u of an appropriate length that depends on n and δ, it is possible to compute a correct description of x with probability at least 0.99. The necessary information can be computed from x, the code of A, n, and δ in time $\mathsf{poly}(|A|, \log(m), |x|, \log(1/\delta))$. ◀

4 Applications

4.1 Equivalence between samplability and succinct probabilistic descriptions

It will be useful in the proof of some results to introduce the following sampler.

Definition of $U(1^m)$.

1. Given 1^m, U samples an integer $\ell \sim [m]$ uniformly at random. It then samples a uniformly random string z of length ℓ, and an independent uniformly random string r of length 2^m.
2. U interprets z as the code of a randomized machine M_z, simulates M_z on the empty input string with randomness r for 2^m steps, and outputs the string y that is left on the output tape of M_z after this simulation.

This sampler satisfies the following properties.

▷ Claim 19. On every input 1^m and for every choice of its randomness, $U(1^m)$ runs in time at most $2^{O(m)}$.

Proof. This is immediate from the definition. ◁

▷ Claim 20. Suppose $x \in \{0,1\}^*$ is a string such that $\mathsf{rKt}(x) \leq k$. Then the probability of x under $U(1^k)$ is at least $(1/k) \cdot 2^{-k} \cdot (2/3)$.

Proof. Since $\mathsf{rKt}(x) \leq k$, there is a randomized machine M running in time at most t such that

$$\Pr[\boldsymbol{M}_{\leq t} = x] \;\geq\; 2/3 \quad \text{and} \quad |M| + \log t \;\leq\; k.$$

Let $\ell = |M| \leq k$, and note that $t \leq 2^k$. It is clear that x is output by $U(1^k)$ with probability at least $(1/\ell) \cdot 2^{-\ell} \cdot (2/3) \geq (1/k) \cdot 2^{-k} \cdot (2/3)$. ◁

We state next an immediate consequence of the coding theorem for rKt and the existence of universal time-bounded samplers. For concreteness, we focus on the generation of prime numbers in the context of Problem 5.

▶ **Theorem 21** (Equivalence between fast samplability and short time-bounded descriptions). *The following statements are equivalent:*

(i) Sampling Algorithm. *For every $\varepsilon > 0$, there is a randomized algorithm $A(1^n)$ sampling strings in $\{0,1\}^*$ that runs in time $T(n) = O(2^{\varepsilon n})$ and for which the following holds. For every large n, there is an n-bit prime q_n such that $\Pr[A(1^n)$ outputs $q_n] \geq 2^{-\varepsilon n}$.*

(ii) Short Descriptions. *Let $\delta > 0$ be an arbitrary constant. For every large n, there is an n-bit prime p_n such that $\mathsf{rKt}(p_n) \leq \delta n$.*

Proof. That samplability as in the first item leads to short descriptions follows immediately from Theorem 16 by taking $\varepsilon > 0$ small enough as a function of the given $\delta > 0$. For the other direction, for a given $\varepsilon > 0$, we consider the sampler $A(1^n) \overset{\text{def}}{=} U(1^m)$ with $m = \varepsilon' n$, where $\varepsilon' = \varepsilon / C$ for a large constant C. By Claim 19, $A(1^n)$ runs in time $O(2^{\varepsilon n})$. Under the assumption that (*ii*) holds, Claim 20 guarantees that for large n there is an n-bit prime q_n that is output by $A(1^n)$ with probability at least $2^{-\varepsilon n}$. ◀

This should be contrasted with the equivalence between the existence of primes of bounded Kt complexity and fast deterministic generation of primes. As mentioned in Section 1.1.2, the equivalence in Theorem 21 also holds in a broader sense.

4.2 Instance-based search-to-decision reduction for rKt and its consequences

In this section, we show that there is a *uniform* polynomial-time "approximate" search-to-decision reduction for rKt. More precisely, we show that given a *linear* approximation of the rKt complexity of the input string, it is possible to output a valid rKt representation that is optimal up to a *constant factor.*

We will need the following definition.

▶ **Definition 22.** *Let $\gamma \colon \mathbb{N} \to \mathbb{R}$. We say that a function $\mathcal{O} \colon \{0,1\}^* \to \mathbb{N}$ γ-approximates rKt complexity if for every string $x \in \{0,1\}^*$,*

$$\frac{\mathsf{rKt}(x)}{\gamma(|x|)} \;\leq\; \mathcal{O}(x) \;\leq\; \gamma(|x|) \cdot \mathsf{rKt}(x).$$

If this holds for a constant $\gamma \in \mathbb{N}$, we say that $\mathcal{O}$ linearly approximates rKt.

▶ **Theorem 23** (An instance-based search-to-decision reduction for rKt). *Let $\mathcal{O}$ be a function that linearly approximates rKt complexity. There is a randomized polynomial-time algorithm with access to $\mathcal{O}$ that, when given an input string x, outputs with probability ≥ 0.99 a valid rKt representation of x of complexity $O(\mathsf{rKt}(x))$. Furthermore, this algorithm makes a single query q to $\mathcal{O}$, where $q = x$.*

Proof. Given an input $x \in \{0,1\}^*$, let $\widetilde{k} \overset{\text{def}}{=} \mathcal{O}(x)$ be the estimate provided by the oracle, and recall that $\mathsf{rKt}(x)/C \leq \widetilde{k} \leq C \cdot \mathsf{rKt}(x)$ for a universal constant C. By Claim 20, we get that the universal sampler $U(1^{C \cdot \widetilde{k}})$ outputs x with probability at least $\delta \overset{\text{def}}{=} (1/(C \cdot \widetilde{k})) \cdot 2^{-C \cdot \widetilde{k}} \cdot (2/3)$. In addition, by Claim 19, the running time of this sampler is bounded by $2^{O(C \cdot \widetilde{k})}$. It then follows from the "moreover" part of Theorem 16 that it is possible to efficiently compute with probability at least 0.99 a valid rKt representation of x of complexity $O(C \cdot \widetilde{k}) = O(C^2 \cdot \mathsf{rKt}(x)) = O(\mathsf{rKt}(x))$. ◀

As explained in Section 1, an interesting aspect of this reduction is that it works on an *input by input* basis. In other words, if we are able to linearly approximate the rKt complexity of the input string, then we can compute a near-optimal representation for the same string.

We note that a reduction that works on an input by input basis is easy to come up with in the setting of (time-unbounded) Kolmogorov complexity: from an upper bound on $K(x)$, it is possible to compute a string that represents x of complexity at most $K(x)$ simply by running in parallel all machines of at most this description length. Theorem 23 can be interpreted as an analogue result in the more challenging setting of time-bounded Kolmogorov complexity.

Using ideas from the proof of the coding theorem for rKt, we can also show an *unconditional* instance-based search-to-decision reduction for Kt.

▶ **Theorem 24** (An instance-based search-to-decision reduction for Kt). *Let $\mathcal{O}$ be a function that linearly approximates* Kt *complexity. There is a randomized polynomial-time algorithm with access to $\mathcal{O}$ that, when given an input string x, outputs with probability ≥ 0.99 a valid* Kt *representation of x of complexity $O(\mathsf{Kt}(x))$. Furthermore, this algorithm makes a single query q to $\mathcal{O}$, where $q = x$.*

Proof. We are given an input $x \in \{0,1\}^*$ where $|x| = n$, and $\mathsf{Kt}(x)/C \leq \widetilde{k} \leq C \cdot \mathsf{Kt}(x)$ for a universal constant C, where $\widetilde{k}$ is obtained via a single query to $\mathcal{O}$ on x. Our encoding algorithm is as follows:

1. Pick $u \in \{0,1\}^d$ uniformly at random, where $d = O\left(\log\left(n \cdot 2 \cdot 2^{C \cdot \widetilde{k}}\right)\right) = O(\widetilde{k} + \log(n))$.
2. Invoke the deterministic algorithm $M(1^n, u)$ from Lemma 15 to obtain a hash function $H \colon \{0,1\}^n \to \{0,1\}^{O(\widetilde{k})}$, and compute $z = H(x)$.
3. Output $(n, \widetilde{k}, u, z)$.

It is easy to verify that the output of the above procedure has length

$$O(\log(\widetilde{k}) + \log(n)) = O(\mathsf{Kt}(x) + \log(n)) = O(\mathsf{Kt}(x)).$$

Next, we claim that with probability at least 0.99 over the choice of u, the tuple generated in Step 3 can be easily converted into a valid Kt representation of x. (Note that the randomness is only over the encoding algorithm that generates the Kt representation, which provides a deterministic description.) To decode x from this information, we first run $M(1^n, u)$ from Lemma 15 to recover the hash function H. We then enumerate each deterministic machine of description length at most $C \cdot \widetilde{k}$ and simulate it for at most $2^{C \cdot \widetilde{k}}$ steps. For each output y of these machines that is in $\{0,1\}^n$, we compute $H(y)$ and output the first y such that $H(y) = z$. Note that there are at most $\ell = 2 \cdot 2^{C \cdot \widetilde{k}}$ distinct machines of length at most $C \cdot \widetilde{k}$, and each machine produces at most one output string. By the fact that $\mathsf{Kt}(x) \leq C \cdot \widetilde{k}$, at least one of them will output x. Also, by Lemma 15, for at least a fraction of .99 of the u's, the hash function H obtained from u isolates every n-bit string in the set of outputs of these ℓ machines, and x is the only string such that $H(x) = z$. Therefore, for any such "good" u, the string x can be decoded correctly from n, $\widetilde{k}$, u, and z. It is easy to see that the running time of the decoding algorithm is $\mathrm{poly}(n) \cdot 2^{O(\widetilde{k})}$, so the Kt complexity of the resulting representation for x is $O(\mathsf{Kt}(x))$. ◀

Note that Theorem 23 (search-to-decision reduction for rKt) and Theorem 24 (search-to-decision reduction for Kt) complete the proof of Theorem 2 from Section 1.

The next result should be contrasted with the unconditional lower bound of [30] for estimating the rKt complexity of an input string x.

▶ **Theorem 25** (Efficient generation of rKt descriptions). *There is a randomized polynomial-time algorithm E such that on any input string $x \in \{0,1\}^n$, given as advice a string $\alpha = \alpha(x)$ of length $\leq \log\log n + O(1)$, $E(x,\alpha)$ outputs with probability ≥ 0.99 an* rKt *description of x of complexity $O(\mathsf{rKt}(x))$.*

Proof. Algorithm E computes on $x \in \{0,1\}^n$ as follows. It expects $\alpha \in \{0,1\}^{\log\log n + O(1)}$ to encode a tight approximation $\tilde{k}$ to the value $k \overset{\text{def}}{=} \mathsf{rKt}(x) \in [\gamma \cdot n]$, where $\gamma \in \mathbb{N}$ is a universal constant, and $\gamma \cdot n$ is an upper bound on $\max\{\mathsf{rKt}(x) \mid x \in \{0,1\}^n\}$. Since we are only aiming for a constant factor approximation of $\mathsf{rKt}(x)$, we use α to encode the smallest integer β such that $2^\beta \geq \mathsf{rKt}(x)$. Since $\beta \leq \log n + O_\gamma(1)$, α can be encoded with just $\log\log n + O(1)$ bits, and a value $\tilde{k}$ such that $k \leq \tilde{k} \leq 2k$ can be obtained from α. $E(x,\alpha)$ considers the universal sampler $U(1^{\tilde{k}})$ as the algorithm $A(1^{\tilde{k}})$ in Theorem 16. Furthermore, it sets $\delta \overset{\text{def}}{=} (1/\tilde{k}) \cdot 2^{-\tilde{k}} \cdot (2/3)$. By Claims 19 and 20, $A(1^{\tilde{k}}) \equiv U(1^{\tilde{k}})$ outputs x with probability at least δ and in time at most $T = 2^{O(\tilde{k})}$. As a consequence, from the "moreover" part of Theorem 16 it follows that $E(x,\alpha)$ can compute with probability at least 0.99 a valid rKt representation of x of complexity $O(\log(\tilde{k}) + \log(T) + \log(1/\delta)) = O(\tilde{k}) = O(\mathsf{rKt}(x))$ in time $\mathsf{poly}(|U|, \log(\tilde{k}), n, \log(1/\delta)) = \mathsf{poly}(n)$. ◀

The next corollary is immediate from Theorem 23.

▶ **Corollary 26** (Time-bounded short lists with short programs in short time). *Given an arbitrary string x of length n, it is possible to compute with high probability and in polynomial time a collection of at most $d = \log(n) + O(1)$ strings $w_1, \ldots, w_d$ such that at least one of these strings is a valid* rKt *encoding of x of complexity $O(\mathsf{rKt}(x))$.*

Proof. We can enumerate all values of $k \in [O(n)]$ that are a power of 2, and run the instance-based search-to-decision reduction from Theorem 23 on x using k as the query answer. One of such k will be a linear approximation of $\mathsf{rKt}(x)$, and the output of the search-to-decision reduction for this k will be a rKt description of x with complexity $O(\mathsf{rKt}(x))$. ◀

References

1 Eric Allender. Applications of time-bounded Kolmogorov complexity in complexity theory. In *Kolmogorov complexity and computational complexity*, pages 4–22. Springer, 1992.

2 Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *International Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 1–15. Springer, 2001.

3 Eric Allender. The complexity of complexity. In *Computability and Complexity*, pages 79–94. Springer, 2017.

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

5 Luis Filipe Coelho Antunes and Lance Fortnow. Worst-case running times for average-case algorithms. In *Conference on Computational Complexity* (CCC), pages 298–303, 2009.

6 Bruno Bauwens, Anton Makhlin, Nikolai K. Vereshchagin, and Marius Zimand. Short lists with short programs in short time. *Comput. Complex.*, 27(1):31–61, 2018.

7 Bruno Bauwens and Marius Zimand. Linear list-approximation for short programs (or the power of a few random bits). In *Conference on Computational Complexity* (CCC), pages 241–247, 2014.

8 Harry Buhrman, Troy Lee, and Dieter van Melkebeek. Language compression and pseudorandom generators. *Comput. Complex.*, 14(3):228–255, 2005.

9 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Conference on Computational Complexity* (CCC), pages 10:1–10:24, 2016.

10 Lijie Chen, Ce Jin, and R. Ryan Williams. Hardness magnification for all sparse NP languages. In *Symposium on Foundations of Computer Science* (FOCS), pages 1240–1255, 2019.

11 Lijie Chen, Ce Jin, and R. Ryan Williams. Sharp threshold results for computational complexity. In *Symposium on Theory of Computing* (STOC), 2020.

12 Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2006.

13 Lance Fortnow. Kolmogorov complexity and computational complexity. *Complexity of Computations and Proofs. Quaderni di Matematica*, 13, 2004.

14 Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. *J. Comput. Syst. Sci.*, 22(3):407–420, 1981.

15 David Gillman. A Chernoff bound for random walks on expander graphs. *SIAM J. Comput.*, 27(4):1203–1220, 1998.

16 Andrew V. Goldberg and Michael Sipser. Compression and ranking. *SIAM J. Comput.*, 20(3):524–536, 1991.

17 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Symposium on Foundations of Computer Science* (FOCS), pages 247–258, 2018.

18 Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In *Conference on Computational Complexity* (CCC), 2020.

19 Rahul Ilango. Connecting Perebor conjectures: Towards a search to decision reduction for minimizing formulas. In *Computational Complexity Conference* (CCC), 2020.

20 Rahul Ilango, Bruno Loff, and Igor C. Oliveira. NP-hardness of circuit minimization for multi-output functions. In *Computational Complexity Conference* (CCC), 2020.

21 Troy Lee. *Kolmogorov complexity and formula lower bounds*. PhD thesis, University of Amsterdam, 2006.

22 Leonid A. Levin. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Problemy Peredachi Informatsii*, 10(3):30–35, 1974.

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

24 Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer, 2008.

25 Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. *IACR Cryptol. ePrint Arch.*, 2020:423, 2020.

26 Luc Longpré and Sarah Mocas. Symmetry of information and one-way functions. *Inf. Process. Lett.*, 46(2):95–100, 1993.

27 Luc Longpré and Osamu Watanabe. On symmetry of information and polynomial time invertibility. *Inf. Comput.*, 121(1):14–22, 1995.

28 Zhenjian Lu and Igor Carboni Oliveira. An efficient coding theorem via probabilistic representations and its applications. *Electron. Colloquium Comput. Complex.*, 28:41, 2021.

29 James Maynard. The twin prime conjecture. *Japanese Journal of Mathematics*, 14:175–206, 2019.

30 Igor C. Oliveira. Randomness and intractability in Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 32:1–32:14, 2019.

31 Igor C. Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *Computational Complexity Conference* (CCC), pages 27:1–27:29, 2019.

32 Igor C. Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Symposium on Theory of Computing* (STOC), pages 665–677, 2017.

33 Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

34 Alexander Shen, Vladimir A. Uspensky, and Nikolay Vereshchagin. *Kolmogorov complexity and algorithmic randomness*. American Mathematical Society, 2017.

35 Michael Sipser. A complexity theoretic approach to randomness. In *Symposium on Theory of Computing* (STOC), pages 330–335, 1983.

36 Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Inf. Theory*, 42(6):1723–1731, 1996.

37 Terence Tao, Ernest Croot, III, and Harald Helfgott. Deterministic methods to find primes. *Math. Comp.*, 81(278):1233–1246, 2012.

38 Gérald Tenenbaum. *Introduction to analytic and probabilistic number theory*, volume 163. American Mathematical Society, 2015.

39 Luca Trevisan, Salil P. Vadhan, and David Zuckerman. Compression of samplable sources. *Comput. Complex.*, 14(3):186–227, 2005.

40 Gregory Valiant. Lecture Notes for CS265/CME309: Randomized Algorithms and Probabilistic Analysis (Lecture 6), 2019.

41 Thomas Watson. Time hierarchies for sampling distributions. *SIAM J. Comput.*, 43(5):1709–1727, 2014.