Exact Search-To-Decision Reductions for Time-Bounded Kolmogorov Complexity

Shuichi Hirahara
National Institute of Informatics, Tokyo, Japan

Valentine Kabanets
Simon Fraser University, Burnaby, Canada

Zhenjian Lu
University of Warwick, UK

Igor C. Oliveira
University of Warwick, UK

Abstract

A *search-to-decision* reduction is a procedure that allows one to find a solution to a problem from the mere ability to decide when a solution exists. The existence of a search-to-decision reduction for time-bounded Kolmogorov complexity, i.e., the problem of checking if a string x can be generated by a t-time bounded program of description length s, is a long-standing open problem that dates back to the 1960s.

In this work, we obtain new average-case and worst-case search-to-decision reductions for the complexity measure K^t and its randomized analogue rK^t:

1. (Conditional Errorless and Error-Prone Reductions for K^t) Under the assumption that E requires exponential size circuits, we design polynomial-time average-case search-to-decision reductions for K^t in both errorless and error-prone settings.
   In fact, under the easiness of deciding K^t under the uniform distribution, we obtain a search algorithm for any given polynomial-time samplable distribution. In the error-prone reduction, the search algorithm works in the more general setting of conditional K^t complexity, i.e., it finds a minimum length t-time bound program for generating x given a string y.
2. (Unconditional Errorless Reduction for rK^t) We obtain an unconditional polynomial-time average-case search-to-decision reduction for rK^t in the errorless setting. Similarly to the results described above, we obtain a search algorithm for each polynomial-time samplable distribution, assuming the existence of a decision algorithm under the uniform distribution.
   To our knowledge, this is the first unconditional sub-exponential time search-to-decision reduction among the measures K^t and rK^t that works with respect to any given polynomial-time samplable distribution.
3. (Worst-Case to Average-Case Reductions) Under the errorless average-case easiness of deciding rK^t, we design a worst-case search algorithm running in time $2^{O(n/\log n)}$ that produces a minimum length randomized t-time program for every input string $x \in \{0,1\}^n$, with the caveat that it only succeeds on some explicitly computed sub-exponential time bound $t \leq 2^{n^{\varepsilon}}$ that depends on x. A similar result holds for K^t, under the assumption that E requires exponential size circuits.

In these results, the corresponding search problem is solved *exactly*, i.e., a successful run of the search algorithm outputs a t-time bounded program for x of minimum length, as opposed to an approximately optimal program of slightly larger description length or running time.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases average-case complexity, Kolmogorov complexity, search-to-decision reductions

Digital Object Identifier 10.4230/LIPIcs.CCC.2024.29

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2024/059`

COMPUTATIONAL COMPLEXITY CONFERENCE

39th Computational Complexity Conference (CCC 2024).
Editor: Rahul Santhanam; Article No. 29; pp. 29:1–29:56
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding This work received support from the Royal Society University Research Fellowship URF\R1\191059; the UKRI Frontier Research Guarantee Grant EP/Y007999/1; and the Centre for Discrete Mathematics and its Applications (DIMAP) at the University of Warwick.

1 Introduction

The time-bounded Kolmogorov complexity $\mathsf{K}^t(x)$ of an input binary string x is defined as the length of a shortest program that prints out x within t time steps. The corresponding *search* version of the problem would be to find such a shortest program that prints x within time t. Both problems have been studied since the 1960s, and are conjectured to require brute-force (trivial) algorithms to solve them [32]. The existence of an efficient search-to-decision reduction for K^t, i.e., an algorithm to solve the search version of $\mathsf{K}^t(x)$ assuming an algorithm for the decision version of computing $\mathsf{K}^t(x)$, is also an old open problem going back to the 1960s.[1] In fact, it is consistent with current knowledge that there might exist an algorithm that computes $\mathsf{K}^t(x)$ in time linear in $n = |x|$, while any search algorithm for the problem requires time $2^{\Omega(n)}$.

In this work, we obtain new *average-case* and *worst-case* search-to-decision reductions for the measure K^t and its randomized analogue rK^t, which considers the length of the shortest randomized program that prints x with high probability within time t. Our search algorithms have two important features:

- they solve the search problem *exactly*: they find an optimally minimal-size program to print x within t steps (rather than an approximately optimal program of slightly larger size or running in slightly bigger than t time); and
- they succeed with high probability on any given *polynomial-time samplable* distribution (rather than being restricted to the uniform distribution).

It should be noted that such search-to-decision reductions are *necessary* for excluding Pessiland from Impagliazzo's five worlds [18], that is, for basing the security of a one-way function on the average-case hardness of NP. By the result of Liu and Pass [22], the existence of a one-way function is characterized by the average-case hardness of computing time-bounded Kolmogorov complexity over the uniform distribution. If Pessiland is eliminated, it follows that the average-case easiness of time-bounded Kolmogorov complexity implies that every NP search problem is easy on any polynomial-time samplable distribution [19, 3], and in particular, the search problems of finding short programs are also easy. Thus, designing such reductions can be seen as a progress towards excluding Pessiland from Impagliazzo's five worlds.

We describe our results in the next section. We compare them with the existing literature on exact and approximate search-to-decision reductions in Section 1.2.

1.1 Results

Informally, our main results give polynomial-time algorithms for solving the search versions of time-bounded Kolmogorov complexity measures K^t and rK^t, on *average* with respect to any given *polynomial-time samplable distribution*, under the assumption that the corresponding decision versions are easy on average with respect to the *uniform distribution*. For rK^t, such a

[1] One reason why such search-to-decision reductions may be possible to K^t is that the decision version of K^t is conjectured to be NP-complete, and efficient search-to-decision reductions for NP-complete problems are easy to get.

search-to-decision average-case reduction is unconditional, whereas for K^t we make a standard derandomization assumption. Our reduction for rK^t works in the *errorless* average-case setting. Our (conditional) reductions for K^t work in both *errorless* and *error-prone* settings.

We provide a more detailed description of our results in the following subsections.

1.1.1 Average-Case Search-to-Decision for K^t

Below we employ standard definitions of $\mathsf{K}^t(x)$ and $\mathsf{rK}^t(x)$, reviewed in Section 2.1. We let U denote the fixed universal Turing machine used in these definitions.

Let MINKT be the following *decision* problem: Given $(x, 1^s, 1^t)$, where $x \in \{0,1\}^*$ and $s, t \in \mathbb{N}$, decide whether $\mathsf{K}^t(x) \le s$. Let $\mathsf{Search\text{-}MINKT}$ be the corresponding *search* problem: Given $(x, 1^t)$, where $x \in \{0,1\}^*$ and $t \in \mathbb{N}$, find a K^t*-witness of* x, i.e., a program $M \in \{0,1\}^*$ such that $|M| = \mathsf{K}^t(x)$ and $U(M)$ outputs x within t steps.

In our average-case search-to-decision reductions for K^t we consider both *errorless* and *error-prone* settings, which correspond to the average-case complexity classes AvgBPP and $\mathsf{HeurBPP}$, respectively (cf. [4]).

The Errorless Setting. We shall use "$\mathsf{MINKT} \in \mathsf{AvgBPP}$", as an abbreviation for the statement that MINKT can be solved in polynomial time on average *without errors* over polynomial-time samplable distributions. Similarly, we shall use "$\mathsf{Search\text{-}MINKT} \in \mathsf{AvgBPP}$" to state that $\mathsf{Search\text{-}MINKT}$ can be solved in polynomial time on average *without errors* over polynomial-time samplable distributions. More formally, we have the following definitions.[2]

"$\mathsf{MINKT} \in \mathsf{AvgBPP}$": For every polynomial-time samplable distribution family $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{D}_n$ is over $\{0,1\}^n$, there exist a polynomial ρ and a polynomial-time algorithm A such that the following holds for all $n, s, k \in \mathbb{N}$, and all $t \ge \rho(n)$.

1. For all $x \in \{0,1\}^n$, $A(x, 1^s, 1^t, 1^k)$ outputs either $\mathsf{MINKT}(x, 1^s, 1^t)$ or $\perp$, and
2. $\mathbf{Pr}_{x\sim\mathcal{D}_n}\left[A(x, 1^s, 1^t, 1^k) = \mathsf{MINKT}(x, 1^s, 1^t)\right] \ge 1 - \frac{1}{k}$.

"$\mathsf{Search\text{-}MINKT} \in \mathsf{AvgBPP}$": For every polynomial-time samplable distribution family $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{D}_n$ is over $\{0,1\}^n$, there exist a polynomial ρ and a polynomial-time algorithm A such that the following holds for all $n, k \in \mathbb{N}$, and all $t \ge \rho(n)$.

1. For all $x \in \{0,1\}^n$, $A(x, 1^t, 1^k)$ outputs either a K^t-witness of x or $\perp$, and
2. $\mathbf{Pr}_{x\sim\mathcal{D}_n}\left[A(x, 1^t, 1^k) \text{ outputs a } \mathsf{K}^t\text{-witness of } x\right] \ge 1 - \frac{1}{k}$.

Before stating our first result, we recall the following widely believed complexity-theoretic assumption. We use $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$ to denote that there is a language $L \in \mathsf{E}$ and $\varepsilon > 0$ such that L requires Boolean circuits of size at least $2^{\varepsilon \cdot n}$ on every large enough input length n.

▶ **Theorem 1** (Errorless Average-Case Search-to-Decision for K^t)**.** *Assume* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. *Then*

$$\text{“}\mathsf{MINKT} \in \mathsf{AvgBPP}\text{”} \Longrightarrow \text{“}\mathsf{Search\text{-}MINKT} \in \mathsf{AvgBPP}\text{”}.$$

[2] In [4], AvgBPP denotes the class of all the pairs $(L, \mathcal{D})$ of problems L and distributions $\mathcal{D}$ that admit randomized average-polynomial-time algorithms (or, equivalently, randomized errorless heuristic schemes). Our statement "$\mathsf{MINKT} \in \mathsf{AvgBPP}$" deviates from this standard notation in that (1) we abbreviate the input distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$, and (2) the lower bound $\rho(n)$ of the time parameter t depends on the distribution $\mathcal{D}$.

The Error-Prone Setting. Theorem 1 shows an average-case search-to-decision reduction for MINKT in the *errorless* setting, under the assumption that $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. Can we also get a similar reduction in the *error-prone* setting? It turns out that such a search-to-decision reduction is implicit in a recent work by Liu and Pass [23]. However, it requires a stronger assumption saying that $\mathsf{E} \not\subseteq \mathsf{i.o.NSIZE}[2^{o(n)}]$, i.e., that there is a language in E that requires *non-deterministic* circuits of exponential size. We discuss this in more detail next.

We define "$\mathsf{MINKT} \in \mathsf{HeurBPP}$" and "$\mathsf{Search\text{-}MINKT} \in \mathsf{HeurBPP}$" to be the analogs of "$\mathsf{MINKT} \in \mathsf{AvgBPP}$" and "$\mathsf{Search\text{-}MINKT} \in \mathsf{AvgBPP}$", respectively, but in the regime where errors are allowed.[3]

"MINKT ∈ HeurBPP": For every polynomial-time samplable distribution family $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$, with each $\mathcal{D}_n$ over $\{0,1\}^n$, there is a polynomial ρ and a polynomial-time algorithm A such that for all $n, s, k \in \mathbb{N}$, and all $t \geq \rho(n,k)$, $\mathbf{Pr}_{x\sim\mathcal{D}_n}\left[A(x, 1^s, 1^t, 1^k) = \mathsf{MINKT}(x, 1^s, 1^t)\right] \geq 1 - \frac{1}{k}$.

"Search-MINKT ∈ HeurBPP": For every polynomial-time samplable distribution family $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$, with each $\mathcal{D}_n$ over $\{0,1\}^n$, there is a polynomial ρ and a polytime algorithm A such that for all $n, k \in \mathbb{N}$, and all $t \geq \rho(n,k)$, $\mathbf{Pr}_{x\sim\mathcal{D}_n}\left[A(x, 1^t, 1^k) \text{ outputs a } \mathsf{K}^t\text{-witness of } x\right] \geq 1 - \frac{1}{k}$.

As noted above, [23] proved "$\mathsf{MINKT} \in \mathsf{HeurBPP}$" $\Longrightarrow$ "$\mathsf{Search\text{-}MINKT} \in \mathsf{HeurBPP}$", assuming $\mathsf{E} \not\subseteq \mathsf{i.o.NSIZE}[2^{o(n)}]$. We strengthen their result by weakening their assumption to $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. Combined with Theorem 1, this yields average-case search-to-decision reductions for MINKT in both errorless and error-prone settings, under the assumption that $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$.

In fact, we show an even stronger result where we solve the *conditional* variant of the search problem, $\mathsf{Search\text{-}MINcKT}$, on average over polynomial-time samplable distributions, while using the same assumption on the decision problem. We describe this in more detail below.

For $x, y \in \{0,1\}^*$, we say that a program Π is a $\mathsf{K}^t(\cdot \mid y)$-witness of x if $|\Pi| = \mathsf{K}^t(x \mid y)$ and $U(\Pi, y)$ outputs x within t steps.

▶ **Theorem 2** (Error-Prone Average-Case Search-to-Decision for Conditional K^t). *Assume $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. If "$\mathsf{MINKT} \in \mathsf{HeurBPP}$" holds, then for every polynomial-time samplable distribution family $\{\mathcal{D}_{\langle n,m\rangle}\}_{n,m\in\mathbb{N}}$ supported over $\{0,1\}^n \times \{0,1\}^m$, there exist a polynomial ρ and a polynomial-time algorithm A such that for all $n, m, k \in \mathbb{N}$, and all $t \geq \rho(n,m,k)$,*

$$\mathop{\mathbf{Pr}}_{(x,y)\sim\mathcal{D}_{\langle n,m\rangle}}\left[A(x, y, 1^t, 1^k) \textit{ outputs a } \mathsf{K}^t(\cdot \mid y)\textit{-witness of } x\right] \geq 1 - \frac{1}{k}.$$

Note that Theorem 2 implies a search-to-decision reduction for K^t (without the conditional string) by considering the set of polynomial-time samplable distribution families $\{\mathcal{D}_{\langle n,m\rangle}\}_{n,m\in\mathbb{N}}$ restricted to $m = 0$.

While the search-to-decision reductions from Theorems 1 and 2 rely on the assumption $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$, we remark that it is possible to obtain weaker *unconditional* variants of these results using a simple win-win argument. Indeed, if the assumption does not hold then we can solve the corresponding search problem on infinitely many input lengths using

[3] For technical reasons, in the error-prone setting we let the function ρ depend on k in addition to n. (See Remark 31). Obtaining a reduction without this dependence (as in the errorless setting) is an interesting problem.

circuits of size $2^{o(n)}$ (see Appendix C for an implementation of this idea). Consequently, it follows that there are errorless and error-prone search-to-decision reductions for K^t computed by sub-exponential size Boolean circuits, on infinitely many input lengths.

1.1.2 Average-Case Search-to-Decision for rK^t

We use $\mathsf{rK}^t_\lambda(x)$ to denote the minimum length of a randomized program that outputs x with probability at least λ within t steps (see Section 2.1). We often omit λ in informal discussions, tacitly assuming that $\lambda = 2/3$.

Analogously to MINKT, one can also consider the problem of deciding whether $\mathsf{rK}^t(x) \leq s$ given $(x, 1^s, 1^t)$. However, this problem is not very "natural" in the sense that it can only be placed in the class $\exists \cdot \mathsf{PP}$. This is because the precise computation of the acceptance probability of a given machine is a computationally hard counting problem.

Here, we consider a more robust variant, which we call MINrKT, that can be shown to be in (promise) MA. We will then focus on the search version of MINrKT.

Let MINrKT be the following promise problem $(\mathsf{YES}, \mathsf{NO})$:

$$\begin{aligned} \mathsf{YES} &:= \left\{(x, \lambda, 1^s, 1^t, 1^\ell) \mid \mathsf{rK}^t_\lambda(x) \leq s\right\}, \\ \mathsf{NO} &:= \left\{(x, \lambda, 1^s, 1^t, 1^\ell) \mid \mathsf{rK}^t_{\lambda - 1/\ell}(x) > s\right\}. \end{aligned}$$

Next, we describe the search version of MINrKT. We first need some notation. For $x \in \{0,1\}^n$, $t \in \mathbb{N}$ and $0 < \varepsilon, \lambda \leq 1$, we say that a program M is an *ε-rK^t_λ-witness of x* if

- $|M| \leq \mathsf{rK}^t_\lambda(x)$, and
- $U(M, r)$ outputs x within t steps with probability at least $\lambda - \varepsilon$ over $r \sim \{0,1\}^t$.

Let $\mathsf{Search\text{-}MINrKT}$ be the following search problem: Given $(x, \lambda, 1^t, 1^\ell)$, where $x \in \{0,1\}^*$, $t, \ell \in \mathbb{N}$ and $\lambda \in [0,1]$, find an $(1/\ell)$-rK^t_λ-witness of x.

We introduce the statement "$\mathsf{MINrKT} \in \mathsf{AvgBPP}$", which states that MINrKT can be solved in probabilistic polynomial time on average (without errors) over polynomial-time samplable distributions. We first need to specify what it means by solving a *promise* problem in the average-case setting. For an algorithm A, $x \in \{0,1\}^*$, $\lambda \in [0,1]$, and $\ell, t, s \in \mathbb{N}$, we say that A *decides* MINrKT *on* $(x, \lambda, 1^s, 1^t, 1^\ell)$ if the following holds:

$$A(x, \lambda, 1^s, 1^t, 1^\ell) = \begin{cases} 1 & \text{if } \mathsf{rK}^t_\lambda(x) \leq s \\ 0 & \text{if } \mathsf{rK}^t_{\lambda - 1/\ell}(x) > s \\ \textbf{either 0 or 1} & \text{otherwise.} \end{cases}$$

For $\lambda \in \mathbb{R}$, we denote by $|\lambda|$ the bit complexity of λ.

"$\mathsf{MINrKT} \in \mathsf{AvgBPP}$": For every polynomial-time samplable distribution family $\{\mathcal{D}_n\}_{n \in \mathbb{N}}$, where each $\mathcal{D}_n$ is over $\{0,1\}^n$, there exist a polynomial ρ and a probabilistic polynomial-time algorithm A such that the following hold for all $\lambda \in (0,1)$, all $n, s, \ell, k \in \mathbb{N}$, and all $t \geq \rho(n) \cdot \log(1/(1-\lambda))$.

1. For all $x \in \{0,1\}^n$,

$$\mathop{\mathbf{Pr}}_A\left[A \text{ decides } \mathsf{MINrKT} \text{ on } (x, \lambda, 1^s, 1^t, 1^\ell) \textbf{ OR } A(x, \lambda, 1^s, 1^t, 1^\ell, 1^k) = \bot\right] \geq \frac{2}{3}.$$

2. With probability at least $1 - 1/k$ over $x \sim \mathcal{D}_n$,

$$\mathop{\mathbf{Pr}}_A\left[A \text{ decides } \mathsf{MINrKT} \text{ on } (x, \lambda, 1^s, 1^t, 1^\ell)\right] \geq \frac{2}{3}.$$

We also introduce the statement “SearchMINrKT $\in$ AvgBPP”, which states that SearchMINrKT can be solved in probabilistic polynomial time on average (without errors) over polynomial-time samplable distributions (cf. the definition of AvgBPP from [4]).

“SearchMINrKT $\in$ AvgBPP”: For every polynomial-time samplable distribution family $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{D}_n$ is over $\{0,1\}^n$, there exist a polynomial ρ and a probabilistic polynomial-time algorithm A such that the following hold for all all $\lambda \in (0,1)$, all $n, s, \ell, k \in \mathbb{N}$, and all $t \geq \rho(n) \cdot \log(1/(1-\lambda))$.

1. For all $x \in \{0,1\}^n$,

$$\Pr_A\left[A(x, \lambda, 1^t, 1^\ell, 1^k) \text{ outputs either an } (1/\ell)\text{-rK}^t_\lambda\text{-witness of } x \text{ or } \bot\right] \geq 1 - \frac{1}{2^k}.$$

2. With probability at least $1 - 1/k$ over $x \sim \mathcal{D}_n$,

$$\Pr_A\left[A(x, \lambda, 1^t, 1^\ell, 1^k) \text{ outputs an } (1/\ell)\text{-rK}^t_\lambda\text{-witness of } x\right] \geq 1 - \frac{1}{2^k}.$$

▶ **Theorem 3** (Errorless Average-Case Search-to-Decision for rK^t). *We have*

“MINrKT $\in$ AvgBPP” $\Longrightarrow$ “SearchMINrKT $\in$ AvgBPP”.[4]

In contrast to our main results for K^t (Theorems 1 and 2), the search-to-decision reduction for rK^t is unconditional in that it does not rely on a circuit lower bound assumption. To our knowledge, this is the first unconditional reduction for the measures K^t and rK^t that runs in less than exponential time and that works with respect to all polynomial-time samplable distributions.

1.1.3 Worst-Case to Average-Case Search-to-Decision

In our next results, we aim to obtain a *worst-case* search algorithm from the same *average-case* easiness assumptions considered before. Note that this is significantly more challenging than a typical (worst-case to worst-case) search-to-decision reduction.

▶ **Theorem 4** (Conditional Worst-Case to Average-Case Search-to-Decision for K^t). *Assume* $\mathsf{E} \not\subseteq$ i.o.SIZE$[2^{o(n)}]$. *If* “MINKT $\in$ AvgBPP” *holds, then for every $\varepsilon > 0$ and every polynomial β, there is an algorithm A such that for all $n \in \mathbb{N}$ and $x \in \{0,1\}^n$, $A(x)$ runs in time $2^{O(n/\log n)}$ and outputs a program M and an integer t that satisfy the following:*

- $\beta(n) \leq t \leq 2^{n^\varepsilon}$, *and*
- M *is a* K^t*-witness of* x.

▶ **Theorem 5** (Worst-Case to Average-Case Search-to-Decision for rK^t). *If* “MINrKT $\in$ AvgBPP” *holds, every polynomial β, there is a probabilistic algorithm A such that for all $n \in \mathbb{N}$, $x \in \{0,1\}^n$, all $\ell \in \mathbb{N}$, and all $\lambda \in (0,1)$ such that $\lambda \leq 1 - 1/2^{\mathsf{poly}(n)}$, $A(x, \lambda, 1^\ell)$ runs in time $2^{O(n/\log n)} \cdot \mathsf{poly}(|\lambda|, \ell)$ and, with probability at least $1 - 2^{-\ell}$, outputs a program M and an integer t that satisfy the following:*

- $\beta(n) \leq t \leq 2^{n^\varepsilon}$, *and*
- M *is an* $(1/\ell)$-rK$^t_\lambda$*-witness of* x.

In both results, we obtain a sub-exponential time search algorithm that works on every input string x. A caveat is that we have no control over the value of t on which the search algorithm succeeds, while ideally we would like it to succeed on every choice of t presented as an extra input parameter. On the positive side, in both results we make only an *average-case* easiness assumption on the decision problem, i.e., we obtain an interesting worst-case conclusion from a significantly weaker computational assumption.

1.1.4 Weaker Assumptions on the Decision Problems

In fact, for the results stated above, a much weaker assumption on the decision problem suffices to get the same consequence on the search problem. This is a consequence of the nature of our techniques, which we discuss in Section 1.3 below. Consider the following statements.

$(\mathsf{MINKT}, \mathcal{U}) \in \mathsf{HeurBPP}$: There exist a polynomial ρ and a polynomial-time algorithm A such that for all $n, s, k \in \mathbb{N}$, and all $t \geq \rho(n,k)$, $\mathbf{Pr}_{x\sim\{0,1\}^n}\left[A(x, 1^s, 1^t, 1^k) = \mathsf{MINKT}(x, 1^s, 1^t)\right] \geq 1 - \frac{1}{k}$.

$(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$: There exist a constant $c > 0$, a polynomial ρ and a probabilistic polynomial-time algorithm A such that the following hold for all sufficiently large n, all $t \geq \rho(n)$, and all $s \leq n - c \cdot \log\log t$.

1. For every $x \in \{0,1\}^n$ with $\mathsf{K}^t(x) \leq s$, we have $\mathbf{Pr}_A[A(x, 1^s, 1^t) = 1] \geq 2/3$.
2. With probability at least $1/n$ over $x \sim \{0,1\}^n$, we have $\mathbf{Pr}_A[A(x, 1^s, 1^t) = 0] \geq 2/3$.

It turns out that, as shown in the body of the paper, these weaker assumptions (see Proposition 11) suffice in the following search-to-decision reductions:

- Theorems 1 and 4 still hold if replacing "$\mathsf{MINKT} \in \mathsf{AvgBPP}$" with $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$.
- Theorems 3 and 5 still hold if replacing "$\mathsf{MINrKT} \in \mathsf{AvgBPP}$" with $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$.
- Theorem 2 still holds if replacing "$\mathsf{MINKT} \in \mathsf{HeurBPP}$" with $(\mathsf{MINKT}, \mathcal{U}) \in \mathsf{HeurBPP}$.

Consequently, in our search-to-decision reductions the existence of a decision algorithm for the *uniform* distribution provides a search algorithm for any *polynomial-time samplable* distribution.

1.2 Related Work

We now compare our results with prior work on search-to-decision reductions for time-bounded Kolmogorov complexity.

Approximate Reductions. Many previous results on search-to-decision for time-bounded Kolmogorov complexity have focused on approximate reductions (also known as gap reductions), where there is a weaker guarantee on the output of the search algorithm. More precisely, for a string $x \in \{0,1\}^n$ such that $\mathsf{K}^t(x) = s$, the search algorithm is allowed to output a program with the running time $t' \approx t$ and the program size $s' \approx s$.

In a recent development, [30] obtained a worst-case approximate reduction that produces a program with $t' = \mathsf{poly}(|x|, t, s)$, $s' \leq s + \log \mathsf{poly}(|x|, t, s)$, and that runs in randomized time $2^{\varepsilon \cdot s} \cdot \mathsf{poly}(|x|, t, s)$, for an arbitrarily small $\varepsilon > 0$. An advantage of the approximate reduction of [30] with respect to our exact reductions is that it invokes the decision algorithm in a black-box way, while our techniques require access to the code of the decision algorithm.

While approximate reductions are not the focus of this work, we note that some of our techniques can be used to obtain a *polynomial-time* reduction with similar parameters t' and s', under the assumption that E requires exponential size circuits. Although predicated on a hardness assumption, our search-to-decision reduction has essentially the best possible runtime. We refer to Appendix C for the details.

A statement related to the average-case to worst-case search-to-decision reduction for K^t (Theorem 4) appears in [13, Theorem 8.7]. Both results are conditional. However, in contrast to Theorem 4, where the search algorithm produces an *exact* solution, in [13, Theorem 8.7] the search algorithm outputs an *approximate* solution where $s' = s = \mathsf{K}^t(x)$ but t' can be as large as $2^{n/\log n}$ for $t \leq 2^{n^{0.99}}$.

For the time-bounded Kolmogorov complexity measures Kt and rKt, [25, 27] designed efficient reductions with $s' = O(s)$ such that, on a given input string x, the search algorithm only queries the decision algorithm on x.

In these approximate reductions, it is often possible to relax the requirement on the decision algorithm, i.e., the reduction still works when the latter only approximates $\mathsf{K}^t(x)$. Interestingly, this is also the case in our results as a consequence of the discussion in Section 1.1.4, though we obtain an exact solution to the search problem even under a relaxation of the decision algorithm.

Exact Average-Case Reductions. [22] (see also the alternate proof in [30]) established the first error-prone polynomial-time search-to-decision reduction for K^t over the uniform distribution. Another related result appears in [24], which showed that if polynomial-time symmetry of information holds for K^t (i.e., if $\mathsf{K}^t(x, y) \approx \mathsf{K}^{t^{O(1)}}(x) + \mathsf{K}^{t^{O(1)}}(y \mid x)$), then Search-MINKT admits an error-prone polynomial-time algorithm over the uniform distribution. In contrast, here we obtain both error-prone and errorless reductions for K^t for every given *polynomial-time samplable* distribution, under the assumption that E requires exponential size circuits.

While reductions restricted to the uniform distribution are not the focus of this work, complementing the results of [22, 30], which provide *error-prone* search-to-decision reductions for K^t under the uniform distribution, we describe in Appendix D an *errorless* search-to-decision reduction for K^t under the uniform distribution.

As discussed in Section 1.1.1, [23] implicitly established an error-prone search-to-decision reduction for K^t under any polynomial-time samplable distribution, under the assumption $\mathsf{E} \not\subseteq \mathsf{i.o.NSIZE}[2^{o(n)}]$. Our error-prone search-to-decision reduction for K^t weakens this circuit complexity assumption, and provides a search algorithm for the more general case of *conditional* K^t complexity. We note that [23] also establishes a search-to-decision reduction for the probabilistic Kolmogorov complexity measure pK^t, which we do not consider in this work.

Additional Related Work. In two recent works, [29] and [15] obtain *non-uniform* algorithms solving the exact search problem for K^t. In more detail, in these results the size of the non-uniform circuit is of order $2^{4n/5}$ and the circuit neither needs access to, nor assumes the existence of an algorithm for the decision problem. It is not known how to extend these results to uniform algorithms.

In another recent paper, [28] describes a non-uniform polynomial-size search-to-decision reduction when the decision procedure solves MINKT with respect to any underlying universal Turing machine U, given black-box access to it (see their paper for details about this setting).

Search-to-decision reductions have also been investigated in the related setting of circuit complexity theory, where the goal is to compute the complexity of a given input function. [16] investigated this problem for Boolean formulas (corresponding to MFSP, the Minimum Formula Size Problem), and designed a worst-case search-to-decision reduction that runs in time $O(2^{0.67n})$ on an input function of description length n. Additionally, [16] obtained an improved running time of $2^{O(n/\log\log n)}$ when the search algorithm is only required to succeed with high probability over the uniform distribution.

Finally, we note that efficient randomized search-to-decision reductions are known for the complexity class DistNP. Every DistNP search problem can be reduced to some DistNP decision problem [3]. However, such reductions typically do not preserve the problem they reduce from (except for certain DistNP-complete problems like the Bounded Halting Problem), and so do not seem to apply to the case of search-to-decision reductions for MINKT and MINrKT studied in our work.

1.3 Techniques

From a technical perspective, our most interesting results are an unconditional errorless average-case search-to-decision reduction for rK^t (Theorem 3) and a conditional error-prone average-case search-to-decision reduction for K^t (Theorem 2). However, for illustration, we start with a more complete overview of the proof of the conditional errorless average-case search-to-decision reduction for K^t (Theorem 1), which is simpler yet captures some key ideas behind most of our proofs.

Average-Case Search-to-Decision for K^t. Our starting point is the aforementioned result from [24], which showed that if polynomial-time symmetry of information holds for K^t, then Search-MINKT can be solved over the *uniform* distribution. By inspecting the proof more carefully, we observe that if polynomial-time symmetry of information holds for K^t, then given t and x, one can find a shortest t-time program for x in time exponential in the $(t, p(t))$-*computational depth of* x, i.e., $\mathsf{cd}^{t,p(t)}(x) := \mathsf{K}^t(x) - \mathsf{K}^{p(t)}(x)$, for some polynomial p.

To show this, consider $x \in \{0,1\}^n$ and any sufficiently large $t \in \mathbb{N}$. Let y_t be a shortest t-time program for generating x. By the assumed polynomial-time symmetry of information, we get that there is a polynomial p' such that the following holds:

$$\begin{aligned}
\mathsf{K}^{p'(2t)}(y_t \mid x) &\lesssim \mathsf{K}^{2t}(x, y_t) - \mathsf{K}^{p'(2t)}(x) && \text{(by polytime symmetry of information)}\\
&\lesssim |y_t| - \mathsf{K}^{p'(2t)}(x) && \text{(since } x \text{ is determined by } y_t\text{)}\\
&= \mathsf{K}^t(x) - \mathsf{K}^{p'(2t)}(x) && \text{(since } |y_t| = \mathsf{K}^t(x)\text{)}\\
&\leq \mathsf{K}^t(x) - \mathsf{K}^{p(t)}(x) && \text{(by monotonicity of } \mathsf{K}^t \text{ with respect to } t\text{)}\\
&= \mathsf{cd}^{t,p(t)}(x), && \text{(by definition of computational depth)}
\end{aligned}$$

where $p > p'$ is a polynomial. The above essentially says that there is a program Π_{y_t} of size at most $\mathsf{cd}^{t,p(t)}(x)$ such that $U(\Pi_{y_t}, x)$ outputs y_t within $p(t)$ steps.

Consider the following algorithm:

> For an integer s, enumerate all programs $\Pi \in \{0,1\}^{\leq s}$, and run $U(\Pi, x)$ for $p(t)$ steps to obtain a list of candidate K^t-witnesses y, which is guaranteed to include y_t if $\mathsf{cd}^{t,p(t)}(x) \leq s$. For each such candidate y, check if y is indeed a t-time program for x, and output a valid one of the smallest length.

This algorithm runs in time $2^s \cdot \mathsf{poly}(t)$, and finds a K^t-witness for every x with $\mathsf{cd}^{t,p(t)}(x) \leq s$.

Using ideas from prior work on meta-complexity [7, 9, 5, 13], one can show that (assuming $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$), if MINKT is easy on average (in the errorless setting), then polynomial-time symmetry of information for K^t holds (see Lemma 22 below).

Since the $(t, p(t))$-computational depth of x is small, i.e., $O(\log |x|)$, for a uniformly random x with high probability, the above yields a polynomial-time Search-MINKT algorithm over the uniform distribution. We want to extend to all *polynomial-time samplable* distributions.

To this end, we want to say that, for a polynomial-time samplable distribution $\mathcal{D}$, $\mathsf{cd}^{t,p(t)}(x)$ is small for almost all x sampled from $\mathcal{D}$. It turns out that this is true if the *coding theorem* holds for K^t, i.e., if for every $x \in \{0,1\}^n$ in the support of $\mathcal{D}$, $\mathsf{K}^t(x) \leq -\log \mathcal{D}(x) + O(\log n)$, for any sufficiently large $t \geq \mathsf{poly}(n)$. Combining the above with the well-known property of Kolmogorov complexity that for almost all x sampled from $\mathcal{D}$, $\mathsf{K}(x) \geq -\log \mathcal{D}(x) - O(\log n)$, we would get that $\mathsf{cd}^{t,p(t)}(x) = \mathsf{K}^t(x) - \mathsf{K}^{p(t)}(x) \leq \mathsf{K}^t(x) - \mathsf{K}(x) \leq O(\log n)$ for almost all x sampled from $\mathcal{D}$.

Again, using ideas from meta-complexity in prior work, assuming $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$ and average-case easiness of MINKT (in the errorless setting), we can obtain the requisite coding theorem for K^t (see Lemma 23).

Thus, by the coding theorem for K^t, we can already show that if MINKT is easy on average (and assuming $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$), one can efficiently solve Search-MINKT over polynomial-time samplable distributions. However, such an average-case algorithm can make errors for strings x whose $(t, p(t))$-computational depth is *not* small. We would like to recognize such strings x, and output $\perp$ on them. To this end, we will design a deterministic polynomial-time *computational depth certifying algorithm* A with the following two properties:

1. If $A(x)$ accepts, then indeed $\mathsf{cd}^{t,p(t)}(x) \leq O(\log n)$, and
2. For almost all x sampled from $\mathcal{D}$, $A(x)$ accepts.

Given A, our final errorless average-case algorithm for solving Search-MINKT is as follows:

> Given x and t, if the algorithm A accepts, which implies that $\mathsf{cd}^{t,p(t)}(x)$ is small, then we are guaranteed that the previously-mentioned procedure can output a K^t witness of x. Otherwise if algorithm A rejects, which happens with only small probability over $x \sim \mathcal{D}$, we output $\perp$.

It remains to explain how to get the requisite algorithm A. By known results in meta-complexity, if MINKT is easy on average and if $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$, then there is some polynomial q such that given x and t', one can compute in deterministic polynomial time an integer s' such that $\mathsf{K}^{q(t')}(x) \lesssim s' \leq \mathsf{K}^{t'}(x)$. By running this algorithm on both $(x, 1^{q^{-1}(t)})$ and $(x, 1^{p(t)})$, we obtain an integer s such that

$$\mathsf{K}^t(x) - \mathsf{K}^{p(t)}(x) \leq s \lesssim \mathsf{K}^{q^{-1}(t)}(x) - \mathsf{K}^{q(p(t))}(x)$$

(see Lemma 24). Let A be the algorithm that computes a number s as above, accepting if $s \leq O(\log n)$, and rejecting otherwise. By definition, A satisfies property (1) above. Also, A satisfies property (2) above, since as discussed earlier, by the coding theorem, $\mathsf{K}^{q^{-1}(t)}(x) - \mathsf{K}^{q(p(t))}(x) \leq O(\log n)$ for almost all x sampled from $\mathcal{D}$, provided that t (hence $q^{-1}(t)$) is sufficiently large.

Worst-Case Search-to-Decision for K^t. As described above, assuming average-case tractability of MINKT, one can find a K^t-witness of x in time exponential to $\mathsf{cd}^{t,p(t)}(x)$, where p is a polynomial. The observation is that for every x, there exists some good $t \leq 2^{n^{\varepsilon}}$ such that $\mathsf{cd}^{t,p(t)}(x)$ is at most $O(n/\log n)$. We show that using the above-described computational depth certifying algorithm, one can also find such a good t for a given x. Then for such a t, we can find a K^t-witness in time $2^{O(n/\log n)} \cdot \mathsf{poly}(t)$.

Average-Case Search-to-Decision for rK^t. One can use ideas from prior work on meta-complexity, and a known generator with rK^t-style reconstruction, to obtain symmetry of information, coding theorem, and a worst-to-average reduction for rK^t, albeit with an

$O(\log^3 n)$ overhead (as opposed to $O(\log n)$ in the case for K^t), just assuming the average-case easiness of MINrKT (and no derandomization assumptions). By using these tools and following a similar approach as described above for the average-case search-to-decision reduction for K^t, we get an average-case search-to-decision reduction for rK^t with time roughly $2^{O(\log^3 n)} \cdot \mathsf{poly}(t)$, which is quasi-polynomial; see Section B in the appendix for details.

A *polynomial-time* reduction, as stated in Theorem 3, is considerably more challenging to get, since we don't know the desired symmetry of information theorem and coding theorem for rK^t with an optimal $O(\log n)$ overhead. Our approach is to use the symmetry of information theorem (under an average-case easiness assumption for MINKT) and the coding theorem for pK^t with optimal $O(\log n)$ overheads. However, implementing this plan requires a delicate analysis. We consider two variants of computational depth defined as $\mathsf{rK}^t(x) - \mathsf{pK}^{\mathsf{poly}(t)}(x)$ and $\mathsf{pK}^t(x) - \mathsf{K}(x)$, and argue that

1. rK^t-witnesses can be found in time exponential in the computational depth $\mathsf{rK}^t(x) - \mathsf{pK}^{\mathsf{poly}(t)}(x)$ (Lemma 19),
2. the computational depth $\mathsf{rK}^t(x) - \mathsf{pK}^{\mathsf{poly}(t)}(x)$ is upper-bounded by $O(\mathsf{pK}^{t^{1/c}}(x) - \mathsf{K}(x) + \log n)$, for some constant $c > 0$ (Theorem 18).

Finally, using the optimal coding theorems for K and pK^t, we conclude that the running time exponential in $O(\mathsf{pK}^{t^{1/c}}(x) - \mathsf{K}(x) + \log n)$ is actually average polynomial time for every given $t^{1/c}$-time samplable distribution.

The proof of Theorem 18 requires a novel application of techniques from meta-complexity. The key idea is to combine the hitting-set generator $H_m \colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ of [8] and the disperser $G_m \colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ of [31]. The generator H_m has an *efficient* albeit *sub-optimal* reconstruction: if there is a randomized polynomial-time algorithm D that *avoids* $H_m(x,\text{-})$ (i.e., D outputs 0 on input $H_m(x,z)$ for every $z \in \{0,1\}^d$, yet D outputs 1 on most inputs), then $\mathsf{rK}^{\mathsf{poly}(n)}(x) \le O(m + \log n)$. The disperser G_m may be viewed as a hitting-set generator with an *inefficient* but *nearly optimal* reconstruction: if there is an algorithm D that avoids $G_m(x,\text{-})$, then $\mathsf{K}(x) \le m + O(\log n)$. For $x \in \{0,1\}^n$, we set $m \approx \mathsf{K}(x)$ and $m' \approx \mathsf{pK}^{t^{1/c}}(x) - \mathsf{K}(x)$. We then argue that the concatenated generator $G_m(x,z) \circ H_{m'}(x,z')$ (for seeds z and z') has an efficient distinguisher, based on an algorithm that approximates the pK-complexity of its input. On the other hand, $G_m(x,z) \circ \mathcal{U}_{m'}$ is "indistinguishable" from the uniform distribution $\mathcal{U}_{m+m'}$ (by our choice of m). This implies that there is an efficient algorithm that takes m bits of advice and avoids $H_{m'}(x,\text{-})$, which allows us to apply the reconstruction property of $H_{m'}$ to conclude the proof.

We should also point out another important difference between K^t and rK^t witness search. In the search-to-decision reduction for K^t, after generating a list of candidate K^t witnesses in the search algorithm, one can check whether each of them is a valid t-time program that outputs x. However, given a candidate randomized program y and λ, we cannot efficiently check whether y outputs x with probability at least λ or if this probability is less than λ, unless $\mathsf{PP} = \mathsf{BPP}$. However, we can distinguish the set of randomized programs that output x with probability at least λ and those that output x with probability less than $\lambda - (1/\ell)$, in time $\mathsf{poly}(\ell)$. This allows us to find an $(1/\ell)$-rK^t_λ-witness.

Error-Prone Average-Case Search-to-Decision for Conditional K^t. We first describe the proof ideas behind the (conditional) error-prone average-case search-to-decision reductions for K^t in [23] mentioned in Section 1.1.1.

First of all, it was shown in [22] that if MINKT is average-case easy (in the error-prone setting), then infinitely-often one-way functions do not exist. Also, implicit in [22, 23], if infinitely-often one-way functions do not exist, then there is an error-prone average-case algorithm for solving $\mathsf{Search\text{-}MINKT}$ over the "universal t-time-bounded distribution" where each x is assigned the probability mass $2^{-\mathsf{K}^t(x)}$. Thus, to get an average-case $\mathsf{Search\text{-}MINKT}$ algorithm over a *polynomial-time samplable distribution* $\mathcal{D}$, it suffices to argue that $\mathcal{D}$ is *dominated*[5] by the universal t-time-bounded distribution (for some polynomial t). The latter would follow from a coding theorem for $\mathsf{K}^{\mathsf{poly}}$.

While the coding theorem is not known to hold for $\mathsf{K}^{\mathsf{poly}}$, it does hold for $\mathsf{pK}^{\mathsf{poly}}$ [27]. Moreover, it is also known that $\mathsf{K}^{\mathsf{poly}}$ and $\mathsf{pK}^{\mathsf{poly}}$ are essentially equivalent under the derandomization assumption that $\mathsf{E} \not\subseteq \mathsf{i.o.NSIZE}[2^{o(n)}]$ [6]. As a result, assuming $\mathsf{E} \not\subseteq \mathsf{i.o.NSIZE}[2^{o(n)}]$, one gets a coding theorem for $\mathsf{K}^{\mathsf{poly}}$. Using these observations, [23] showed that assuming $\mathsf{E} \not\subseteq \mathsf{i.o.NSIZE}[2^{o(n)}]$, the average-case algorithms for solving $\mathsf{Search\text{-}MINKT}$ over the class of universal poly-time-bounded distributions also work for the class of polynomial-time samplable distributions.

Our key observation is that assuming only $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$, plus the non-existence of infinitely-often one-way functions, one can get an *average-case* coding theorem for $\mathsf{K}^{\mathsf{poly}}$; this result is implicit in [17]. We then show that such an average-case coding theorem for $\mathsf{K}^{\mathsf{poly}}$ implies that polynomial-time samplable distributions are dominated by universal poly-time-bounded distributions *on average*. In turn, this implies that the average-case $\mathsf{Search\text{-}MINKT}$ algorithms over universal poly-time-bounded distributions also work over polynomial-time samplable distributions.

Next, we explain how to generalize these ideas to get an average-case $\mathsf{Search\text{-}MINcKT}$ algorithm. For simplicity, consider a polynomial-time samplable distribution family $\{\mathcal{D}_n\}$ supported over $\{0,1\}^n \times \{0,1\}^n$. Also, let $\{\mathcal{C}_n\}$ be the the family of marginal distributions of $\{\mathcal{D}_n\}$ on the second part. That is, to sample from $\mathcal{C}_n$, we sample (x,y) from $\mathcal{D}_n$ and then output y. We observe the following equivalent way of sampling $\mathcal{D}_n$: First sample y from $\mathcal{C}_n$ and then sample x from $\mathcal{D}_n(\cdot \mid y)$, where $\mathcal{D}_n(\cdot \mid y)$ is the conditional distribution of $\mathcal{D}_n$ on the first part given that the second part is y.

First of all, by borrowing ideas from [22, 23], we show that non-existence of infinitely-often one-way functions implies that there is an (error-prone) average-case algorithm A such that, with high probability over $y \sim \mathcal{C}_n$, A outputs a $\mathsf{K}^t(\cdot \mid y)$-witness of x with high probability over the distribution $\mathcal{E}^t_y$ assigning each x the probability mass $2^{-\mathsf{K}^t(x|y)}$.

In [14], it was shown that if infinitely-often one-way functions do not exist, then one can get an average-case *conditional* coding theorem for $\mathsf{pK}^{\mathsf{poly}}$. By "derandomizing" the proof, we can show that assuming $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$, plus the non-existence of infinitely-often one-way functions, one gets an average-case conditional coding theorem for $\mathsf{K}^{\mathsf{poly}}$ which says that with high probability over $(x,y) \sim \mathcal{D}_n$,

$$\mathsf{K}^{\mathsf{poly}(n)}(x \mid y) \lesssim \frac{1}{\mathcal{D}_n(x \mid y)}. \tag{1}$$

Note that by an averaging argument, we get that with high probability over $y \sim \mathcal{C}_n$, Equation (1) holds with high probability over $x \sim \mathcal{D}_n(\cdot \mid y)$.

Now using this conditional coding theorem, we get that with high probability over $y \sim \mathcal{C}_n$, the distribution $\mathcal{E}^t_y$ dominates $\mathcal{D}_n(\cdot \mid y)$, again, on average, for any sufficiently large $t \geq \mathsf{poly}(n)$. By the same observation as discussed earlier, such "average-case domination" suffices for us to argue that the algorithm A, which works on average over $\mathcal{E}^t_y$, also works on average over the distribution $\mathcal{D}_n(\cdot \mid y)$. As a result, we get that with high probability over $y \sim \mathcal{C}_n$, A output a $\mathsf{K}^t(\cdot \mid y)$-witness of x with high probability over $x \sim \mathcal{D}_n(\cdot \mid y)$. This implies that A solves $\mathsf{Search\text{-}MINcKT}$ on average over $(x,y) \sim \mathcal{D}_n$.

[5] Recall that a distribution $\mathcal{D}$ dominates another distribution $\mathcal{D}'$ if $\mathcal{D}(x) \geq \mathcal{D}'(x)/\mathsf{poly}(n)$ for every x.

1.4 Concluding Remarks, Directions, and Open Problems

We have designed exact search-to-decision reductions for K^t and rK^t complexities in the average-case setting. The results for K^t hold under a widely believed hardness assumption, while the results for rK^t are unconditional. We have also made progress on worst-case to average-case search-to-decision reductions, where a worst-case search algorithm is obtained from an average-case easiness assumption on the decision problem. (As stated in Section 1.1.4, the assumptions on the decision problems in most results can be made considerably weaker, while maintaining the same conclusion.) A key contribution of our results is showing that search-to-decision reductions exist for any fixed polynomial-time samplable distribution. (We also describe new approximate reductions in Appendix C, and a new errorless reduction over the uniform distribution in Appendix D.) A summary of the existing average-case polynomial-time search-to-decision reductions for the measures K^t and rK^t appears in Table 1.

We would like to highlight the following problems and directions:

1. In the worst-case setting, it is currently possible that computing $\mathsf{K}^t(x)$ admits a linear time algorithm, while finding a minimum t-time bounded program for x requires time $2^{\Omega(|x|)}$. Are there sub-exponential time (exact) worst-case to worst-case search-to-decision reductions for K^t and rK^t?

2. Can we improve Theorems 4 and 5 so that the search algorithm works for every choice of the parameter t? Note that this would provide a positive solution to the previous problem.

3. Design an unconditional polynomial-time error-prone search-to-decision reduction for rK^t for polynomial-time samplable distributions.

4. Our search-to-decision reductions are non-black-box, i.e., the search algorithm relies on the code of the decision algorithm. Is it possible to obtain black-box search-to-decision reductions for the settings considered in our work?

5. Is it possible to combine our techniques for exact search-to-decision with the techniques from [30] and Appendix C for approximate search-to-decision to obtain stronger results?

Table 1 Summary of average-case polytime search-to-decision reductions for K^t and rK^t.

Assumption	Measure	Distribution	Errorless or Error-prone	Reference
None	K^t	Uniform	Error-prone	[22]
$\mathsf{E} \not\subseteq \mathsf{i.o.NSIZE}[2^{o(n)}]$	K^t	P-Samplable	Error-prone	[23]
None	K^t	Uniform	Errorless	Appendix D
$\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$	K^t	P-Samplable	Errorless	Theorem 1
$\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$	K^t	P-Samplable	Error-prone	Theorem 2
None	rK^t	Uniform	Error-prone	[30][6]
None	rK^t	P-Samplable	Errorless	Theorem 3

[6] The proof of [30, Theorem 1.3] via list recoverable codes extends to rK^t with simple modifications.

2 Preliminaries

2.1 Definitions and Notation

For a string $w \in \{0,1\}^*$, we use $|w| \in \mathbb{N}$ to denote its length. The empty string is denoted by ϵ.

Time-Bounded Kolmogorov Complexity. Let U be a Turing machine. Given a positive integer t and a string $x \in \{0,1\}^*$, we let

$$\mathsf{K}^t_U(x) = \min_{p \in \{0,1\}^*} \Big\{ |p| \;\mid\; U(p,\epsilon) \text{ outputs } x \text{ in at most } t \text{ steps} \Big\}.$$

We say that $\mathsf{K}^t_U(x)$ is the *t-time-bounded Kolmogorov complexity of x* (with respect to U). As usual, we fix U to be a time-optimal machine [21], i.e., a universal machine that is almost as fast and length efficient as any other universal machine, and drop the index U when referring to time-bounded Kolmogorov complexity measures.

We also consider a randomized variant of K^t where instead of having a deterministic machine that prints x, we consider a randomized machine that generates x with high probability. Given a probability parameter $\lambda \in [0,1]$ and a positive integer t, we let

$$\mathsf{rK}^t_\lambda(x) = \min_{p \in \{0,1\}^*} \Big\{ |p| \;\mid\; \mathop{\mathbf{Pr}}_{r \sim \{0,1\}^t}[U(p,r) \text{ outputs } x \text{ in at most } t \text{ steps}] \geq \lambda \Big\}.$$

denote the *t-time-bounded randomized Kolmogorov complexity of x*. Note that we do not require that $U(p,r)$ stops in time at most t on every r.[7] We assume that the random string r is given on a separate input tape.

Also, for $\lambda \in [0,1]$ and a positive integer t, we let

$$\mathsf{pK}^t_\lambda(x) = \min \Big\{ k \;\mid\; \mathop{\mathbf{Pr}}_{r \sim \{0,1\}^t}[\exists p \in \{0,1\}^k, U(p,r) \text{ outputs } x \text{ in at most } t \text{ steps}] \geq \lambda \Big\}.$$

denote the *t-time-bounded probabilistic Kolmogorov complexity of x*. For simplicity, in both definitions above, we omit λ when $\lambda = 2/3$.

For more information about different notions of randomized time-bounded Kolmogorov complexity and their applications, we refer to [26].

We use $\mathsf{K}(x)$ to denote the (time-unbounded) Kolmogorov complexity of x.

These definitions are extended to *conditional* Kolmogorov complexity measures in the usual way. For instance, in $\mathsf{rK}^t(x \mid y)$ the machine U is also given access to the string y as part of its input. We assume that the string y is given on a separate input tape.

Probability Distributions. We will consider distributions supported over pairs of strings. Let $\mathcal{D} = \{\mathcal{D}_{\langle n,m\rangle}\}_{n,m\in\mathbb{N}}$ be a family of polynomial-time samplable distributions[8], where each $\mathcal{D}_{\langle n,m\rangle}$ is supported over $\{0,1\}^n \times \{0,1\}^m$. For $y \in \{0,1\}^m$, we denote by $\mathcal{D}_{\langle n,m\rangle}(\cdot \mid y)$ the conditional distribution of $\mathcal{D}_{\langle n,m\rangle}$ on the first part given that the second part is y.

[7] This condition would be computationally difficult to check for a given randomized program. However, in a setting where it might be relevant, it can be achieved with a clocked program by storing the value t using $\log t$ bits, or an approximation of t (e.g., the exponent of the smallest power of 2 not smaller than t) using just $\log \log t$ bits.

[8] Recall that $\mathcal{D}$ can be sampled in polynomial time if there is a polynomial-time algorithm Samp such that $\mathsf{Samp}(1^{\langle n,m\rangle}, r)$ is distributed according to $\mathcal{D}_{\langle n,m\rangle}$ when r is a uniformly random string of length $\mathsf{poly}(n,m)$.

We use $\mathcal{D}_{\langle n,m\rangle}(x, y)$ to denote the probability that the pair (x, y) is sampled from $\mathcal{D}_{\langle n,m\rangle}$. Similarly, $\mathcal{D}_{\langle n,m\rangle}(x \mid y)$ denotes the probability that x is sampled from the conditional distribution $\mathcal{D}_{\langle n,m\rangle}(\cdot \mid y)$.

2.2 Basic Results in Kolmogorov Complexity

We will need the following results.

▶ **Fact 6.** *For every $x \in \{0,1\}^*$, time bound $t \in \mathbb{N}$, and $\lambda > 1/2$,*

$$\mathsf{K}(x) \leq \mathsf{rK}^t_\lambda(x).$$

Since we have not explicitly considered prefix-free encodings in our definitions, below we simply observe the following result, which is useful later.

▶ **Lemma 7** (“Kraft's Inequality for K”). *For all $n > 0$,*

$$\sum_{x\in\{0,1\}^n} 2^{-\mathsf{K}(x)} \leq n^{O(1)}.$$

Proof. For every $x \in \{0,1\}^n$, its Kolmogorov description of length $\mathsf{K}(x)$ can be encoded using a *prefix-free* code (where no codeword is a prefix of another codeword) at the expense of extra $O(\log n)$ bits (roughly, by adding the encoding of the integer value $\mathsf{K}(x) \leq n + O(1)$, using a simple prefix-free binary code where each bit of the message is repeated twice, and 10 is added at the end). Let $C(x)$ denote the length of this prefix-free encoding of x. Then we have

$$\begin{aligned}\sum_{x\in\{0,1\}^n} 2^{-\mathsf{K}(x)} &\leq \sum_{x\in\{0,1\}^n} 2^{-C(x)+O(\log n)} \\ &\leq n^{O(1)} \cdot \sum_{x\in\{0,1\}^n} 2^{-C(x)} \\ &\leq n^{O(1)},\end{aligned}$$

where the last step uses Kraft's inequality (saying that for every prefix-free binary code with lengths $C(x)$, we have $\sum_x 2^{-C(x)} \leq 1$). ◀

▶ **Theorem 8** (Coding Theorem for pK^t [27]). *There is a constant $c > 0$, such that the following holds. For any distribution family $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{D}_n$ is over $\{0,1\}^n$, samplable in time $p(n)$, we have $\mathsf{pK}^{p(n)^c}(x) \leq -\log \mathcal{D}_n(x) + O(\log p(n))$.*

▶ **Lemma 9** (See [14, Lemma 9]). *There exists a universal constant $b > 0$ such that for any distribution family $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{D}_n$ is over $\{0,1\}^n$, and $\gamma \in \mathbb{N}$,*

$$\Pr_{x\sim\mathcal{D}_n}\left[\mathsf{K}(x) < \log\frac{1}{\mathcal{D}_n(x)} - \gamma\right] < \frac{n^b}{2^\gamma}.$$

▶ **Lemma 10** (Success Amplification for rK^t). *For any string $x \in \{0,1\}^*$, time bound $t \in \mathbb{N}$, and $q \in \mathbb{N}$, we have*

$$\mathsf{rK}^{t'}_{1-1/q}(x) \leq \mathsf{rK}^t(x) + O(\log\log q),$$

where $t' := t \cdot O(\log q)$.

▶ **Proposition 11.** *The following hold.*

1. "$\mathsf{MINKT} \in \mathsf{HeurBPP}$" $\Longrightarrow$ $(\mathsf{MINKT},\mathcal{U}) \in \mathsf{HeurBPP}$.
2. "$\mathsf{MINKT} \in \mathsf{AvgBPP}$" $\Longrightarrow$ $(\mathsf{coMINKT},\mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$.
3. "$\mathsf{MINrKT} \in \mathsf{AvgBPP}$" $\Longrightarrow$ $(\mathsf{coMINKT},\mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$.

Proof. The implication from "$\mathsf{MINKT} \in \mathsf{HeurBPP}$" to $(\mathsf{MINKT},\mathcal{U}) \in \mathsf{HeurBPP}$ (Item 1) is immediate. Next, we show that "$\mathsf{MINrKT} \in \mathsf{AvgBPP}$" implies $(\mathsf{coMINKT},\mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ (Item 2).

Suppose "$\mathsf{MINrKT} \in \mathsf{AvgBPP}$" holds. Then it follows that there exist a polynomial ρ and a probabilistic polynomial-time algorithm A' such that the following hold for all $n, s \in \mathbb{N}$, and all $t \geq \rho(n)$.

- For all $x \in \{0,1\}^n$,

$$\Pr_{A'}\big[A' \text{ decides } \mathsf{MINrKT} \text{ on } (x, 2/3, 1^s, 1^t, 1^n) \textbf{ OR } A'(x, 2/3, 1^s, 1^t, 1^n) = \bot\big] \geq \frac{2}{3}. \quad (2)$$

- With probability at least $1 - 1/(2\log t)$ over $x \sim \{0,1\}^n$,

$$\Pr_{A'}\big[A' \text{ decides } \mathsf{MINrKT} \text{ on } (x, 2/3, 1^s, 1^t, 1^n)\big] \geq \frac{2}{3}. \quad (3)$$

Let A be the algorithm: On input $(x, 1^s, 1^t)$, A accepts if $A'(x, 2/3, 1^s, 1^t, 1^n)$ outputs 1 or $\bot$; otherwise it rejects. We claim that the algorithm A satisfies the conditions stated for $(\mathsf{coMINKT},\mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$.

Let $t \geq \rho(n)$ and $s \leq n - 2\log\log t$.

On the one hand, consider $x \in \{0,1\}^n$ such that $\mathsf{K}^t(x) \leq s$. Then we also have $\mathsf{rK}^t(x) \leq s$. This means that $(x, 2/3, 1^s, 1^t, 1^n)$ is a YES instance of MINrKT. Then by Equation (2), $A'(x, 2/3, 1^s, 1^t, 1^n)$ outputs 1 or $\bot$ with probability at least 2/3, which implies that $A(x, 1^s, 1^t)$ accepts with probability at least 2/3.

On the other hand, by a counting argument, we have that with probability at least $1 - 1/(2\log t)$ over $x \sim \{0,1\}^n$, $\mathsf{K}(x) \geq n - \log(2\log t) > s$. By Fact 6, we also get that

$$\mathsf{rK}^t_{2/3-1/n}(x) > s.$$

In this case, $(x, 2/3, 1^s, 1^t, 1^n)$ is a NO instance of MINrKT. Combining this fact with Equation (3) and using a union bound, we get that with probability at least $1 - 1/(2\log t) \geq 1/n$ over $x \sim \{0,1\}^n$, $A'(x, 2/3, 1^s, 1^t, 1^n)$ rejects with probability at least 2/3. Note that the above allows us to conclude that $(\mathsf{coMINKT},\mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ holds.

Item 3 can be shown in a similar way. We omit the details. ◀

We will also need the following lemma.

▶ **Lemma 12** (Computational Depth Upper Bound [11]). *For every $\varepsilon > 0$, every non-decreasing polynomials q_{dpt} and p_{dpt}, and every large enough $x \in \{0,1\}^n$, there exists a time bound t^* such that $q_{\mathsf{dpt}}(n) \leq t^* \leq 2^{n^\varepsilon}$ and*

$$\mathsf{K}^{t^*}(x) - \mathsf{K}^{p_{\mathsf{dpt}}(t^*)}(x) \leq O\left(\frac{n}{\log n}\right).$$

Moreover, the same holds if we replace in the above $\mathsf{K}^{t^*}(x) - \mathsf{K}^{p_{\mathsf{dpt}}(t^*)}(x)$ with $\mathsf{rK}^{t^*}(x) - \mathsf{rK}^{p_{\mathsf{dpt}}(t^*)}(x)$.

Proof. We show the proof for randomized time-bound Kolmogorov complexity. The proof can be easily adapted to the deterministic case.

Given $x \in \{0,1\}^n$ and polynomials q_{dpt} and p_{dpt}, define the polynomial $\tau := p_{\mathsf{dpt}} \circ q_{\mathsf{dpt}}$. For an integer $I \geq 1$, consider the following telescoping sum:

$$\begin{aligned}\mathsf{rK}^{\tau(n)}(x) - \mathsf{rK}^{\tau^{(I+1)}(n)}(x) &= \Big(\mathsf{rK}^{\tau(n)}(x) - \mathsf{rK}^{\tau^{(2)}(n)}(x)\Big) \\ &\quad + \Big(\mathsf{rK}^{\tau^{(2)}(n)}(x) - \mathsf{rK}^{\tau^{(3)}(n)}(x)\Big) + \cdots + \Big(\mathsf{rK}^{\tau^{(I)}(n)}(x) - \mathsf{rK}^{\tau^{(I+1)}(n)}(x)\Big),\end{aligned}$$

where $\tau^{(i)}$ denotes the composition of τ with itself i times. For any choice of x, q_{dpt}, and p_{dpt} as in the statement of the lemma, $\mathsf{rK}^{\tau(n)}(x) \leq n + d$, for some universal constant $d \geq 0$; hence, the above sum is at most $n + d$. By averaging, there is some index $i_0 \in [I]$ such that

$$\mathsf{rK}^{\tau^{(i_0)}(n)}(x) - \mathsf{rK}^{\tau^{(i_0+1)}(n)}(x) \leq \frac{n+d}{I}. \tag{4}$$

For this i_0, define $t^* := \tau^{(i_0)}(n)$. Note that $t^* \geq \tau(n) \geq (n)$, since $i_0 \geq 1$ and $p_{\mathsf{dpt}}(\ell) \geq \ell$ for every input ℓ. Letting $c \in \mathbb{N}$ be such that $\tau(n) \leq n^c$ for sufficiently large n, define

$$I := \log_c\!\left(\frac{n^\varepsilon}{\log n}\right).$$

Then $t^* \leq n^{c^I} = 2^{n^\varepsilon}$. Moreover,

$$\begin{aligned}\mathsf{rK}^{t^*}(x) - \mathsf{rK}^{p_{\mathsf{dpt}}(t^*)}(x) &\leq \mathsf{rK}^{t^*}(x) - \mathsf{rK}^{\tau(t^*)}(x) \\ &\leq O\!\left(\frac{n}{\log n}\right), && \text{(by Equation (4))}\end{aligned}$$

where the constant behind the $O(-)$ can depend on ε and c (and hence q_{dpt} and p_{dpt}). ◀

3 Errorless Average-Case Search-to-Decision Reduction for rK^t

Here we prove Theorem 3, re-stated in its stronger form below (cf. Proposition 11).

▶ **Theorem 13.**

$$(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP} \implies \text{“}\mathsf{SearchMINrKT} \in \mathsf{AvgBPP}\text{”}.$$

3.1 Technical Tools

A *randomized oracle* $D \colon \{0,1\}^m \to \{0,1\}$ is a family $\{D_q\}_{q \in \{0,1\}^m}$ of random variables D_q over $\{0,1\}$. When a query $q \in \{0,1\}^m$ is made to a randomized oracle D, a sample $a \sim D_q$ is returned independently.

We say that an algorithm $D \colon \{0,1\}^m \to \{0,1\}$ *ε-avoids* a generator $G \colon \{0,1\}^d \to \{0,1\}^m$ if D is 1 on at least ε fraction of its inputs, and yet $D(G(z)) = 0$ for all $z \in \{0,1\}^d$. Similarly, a randomized oracle D *ε-avoids* a generator G if $\mathbf{Pr}_D[D(w) = 1] \geq \frac{2}{3}$ for at least $\varepsilon 2^m$ inputs $w \in \{0,1\}^m$, and yet $\mathbf{Pr}_D[D(G(z)) = 0] \geq \frac{2}{3}$ for all $z \in \{0,1\}^d$.

We will use the following two hitting-set generators with reconstruction properties.

▶ **Lemma 14** (Implicit in [8]). *There exists a polynomial-time-computable family*

$$H = \left\{ H_{n,m} : \{0,1\}^n \times \{0,1\}^{d(n,m)} \to \{0,1\}^m \right\}_{n,m \in \mathbb{N}}$$

of functions such that $d(n,m) = O(\log^3 m + \log n)$ *and for any* $x \in \{0,1\}^n$ *and any randomized oracle* $D \colon \{0,1\}^m \to \{0,1\}$ *that* ε*-avoids* $H_{n,m}(x, \text{-})$*, it holds that*

$$\mathsf{K}^{t,D}(x) \le 2m + O(\log^3 m + \log n)$$

for $t := \mathsf{poly}(n,m)$*.*

Proof Sketch. For a deterministic oracle D, this is [8, Corollary 4.4]. By inspecting the proof, one can observe that the proof can be generalized to a randomized oracle D. ◀

We need the nearly optimal construction of a disperser obtained by [31]. We regard it as a hitting-set generator with an *inefficient* reconstruction property.

▶ **Lemma 15.** *There exists a polynomial-time-computable family*

$$G = \left\{ G_{n,m} : \{0,1\}^n \times \{0,1\}^{O(\log n)} \to \{0,1\}^m \right\}_{n,m \in \mathbb{N}}$$

of functions such that for any $x \in \{0,1\}^n$ *and any oracle* $D \colon \{0,1\}^m \to \{0,1\}$ *that* ε*-avoids* $G_{n,m}(x, \text{-})$*, it holds that*

$$\mathsf{K}^D(x) \le m + O(\log n).$$

Proof. We may assume without loss of generality that $m \le 2n$ because otherwise the conclusion is obvious. It is shown in [31, Theorem 1.4] that for every n, k and constant $\varepsilon > 0$, there exists a strongly explicit bipartite graph (V, W, E) with left degree $2^d = n^{O(1)}$ such that $V = [2^n]$, $|W| = \Theta(2^{k+d-3\log n})$, and every subset $A \subseteq V$ of size at least 2^k has at least $(1 - \varepsilon/2)|W|$ distinct neighbours in W. We let $|W| = 2^m$, where $m = k + d - 3\log n \pm \Theta(1)$, and view the vertices in W as m-bit strings. We define $G_{n,m}(x,z)$ to be the z-th neighbour of $x \in \{0,1\}^n \equiv V$ for every $z \in [2^d] \equiv \{0,1\}^d$.

Let A be the set of n-bit strings $x \in \{0,1\}^n$ such that $D(G_{n,m}(x,z)) = 0$ for every $z \in \{0,1\}^d$. We claim that the size of A is at most 2^k. Assume, towards a contradiction, that $|A| \ge 2^k$. Let Γ denote the set of the neighbours of A. By the property of the disperser, $|\Gamma| \ge (1 - \varepsilon/2)|W|$. By the definition of A, for every $w \in \Gamma$, we have $D(w) = 0$. This contradicts the assumption that $D(w) = 1$ for at least an ε fraction of $w \in \{0,1\}^m$.

Observe that the elements of A can be enumerated given $n, m \in \mathbb{N}$ and oracle access to D. Thus, we obtain $\mathsf{K}^D(x) \le \log|A| + O(\log nm) \le k + O(\log n) \le m + O(\log n)$ for every $x \in A$. ◀

▶ **Lemma 16** ([13, 5, 6]). *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$*, then there exists a randomized polynomial-time algorithm* M *such that for every* $x \in \{0,1\}^*$ *and every* $t \ge |x|$*,*

$$\mathsf{pK}^{t^{O(1)}}(x) - O(\log n) \le M(x, 1^t) \le \mathsf{pK}^t(x)$$

with high probability over the internal randomness of M*.*

▶ **Lemma 17** (Symmetry of Information for pK^t; implicit in [13, 6]). *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then there exist polynomials* p_{Sol} *and* p_0 *such that for all sufficiently large* $x, y \in \{0,1\}^*$ *and every* $t \ge p_0(|x| + |y|)$*,*

$$\mathsf{pK}^{p_{\mathsf{Sol}}(t)}(y \mid x) \le \mathsf{pK}^t(x,y) - \mathsf{pK}^{p_{\mathsf{Sol}}(t)}(x) + \log p_{\mathsf{Sol}}(|x| + |y|) + \log p_{\mathsf{Sol}}(\log t).$$

The Symmetry of Information statement for pK^t as in Lemma 17 above was proved in [6] under the stronger assumption that distributional NP is easy on average for randomized polynomial-time algorithms in the errorless setting. It turns out that the weaker assumption on the average-case errorless easiness of MINKT (rather than all problems in NP) suffices to get the same result, with the proof similar to that in [6]. For completeness, we give the proof of Lemma 17 in Appendix A.

3.2 On Computational Depth

The following is the key result enabling us to argue that an algorithm that runs in time $2^{O(\mathsf{rK}^{\mathsf{poly}(t)}(x)-\mathsf{K}(x)+\log n)}$ also runs in time $2^{O(\mathsf{pK}^{\mathsf{poly}(t)}(x)-\mathsf{K}(x)+\log n)}$. The latter runtime can be shown to be average-polynomial-time over any t-time samplable distribution.

▶ **Theorem 18.** *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$*, then for some polynomial* p*, for all* $n \in \mathbb{N}$*, all* $t \geq n$*, and all* $x \in \{0,1\}^n$*, it holds that*

$$\mathsf{rK}^{p(t)}(x) - \mathsf{K}(x) \leq O(\mathsf{pK}^t(x) - \mathsf{K}(x) + \log n).$$

Moreover, for every polynomial q*, there exists a randomized algorithm* M *such that, on input* (x,t)*, with probability at least* $1-o(1)$ *over the internal randomness of* M*, outputs* $v \in \mathbb{N}$ *such that*

$$\mathsf{rK}^{p(t)}(x) - \mathsf{pK}^{q(t)}(x) - O(\log n) \leq v \leq O(\mathsf{pK}^t(x) - \mathsf{K}(x) + \log n).$$

in time $2^{O(\mathsf{pK}^t(x)-\mathsf{K}(x)+\log n)}$.

Proof. Let G be the function of Lemma 15. Let H be the black-box hitting set generator construction of Lemma 14. The idea is to avoid $G_{n,m}(x,z) \circ H_{n,m'}(x,z')$ by measuring its Kolmogorov complexity for some m and m'. Let M be the algorithm of Lemma 16.

Define $m := \mathsf{K}(x) - c\log n$ and $m' = \mathsf{pK}^t(x) - \mathsf{K}(x) + \log^3 m' + c'\log n$ for sufficiently large constants c, c'. Observe that there exists a polynomial q such that

$$\begin{aligned}\mathsf{pK}^{q(t)}(G_{n,m}(x,z) \circ H_{n,m'}(x,z')) &\leq \mathsf{pK}^t(x) + |z'| + O(\log n)\\ &\leq m + m' - (c' - c - O(1))\log n.\end{aligned}$$

Let D_0 be an algorithm that takes a string $w \in \{0,1\}^{m+m'}$ and outputs 0 if and only if

$$M(w, 1^{q(t)}) \leq m + m' - (c' - c - O(1))\log n.$$

Then, $D_0(G_{n,m}(x,z) \circ H_{n,m'}(x,z')) = 0$ because

$$\begin{aligned}M(G_{n,m}(x,z) \circ H_{n,m'}(x,z'), 1^{q(t)}) &\leq \mathsf{pK}^{q(t)}(G_{n,m}(x,z) \circ H_{n,m'}(x,z'))\\ &\leq m + m' - (c' - c - O(1))\log n.\end{aligned}$$

On the other hand, for a uniformly random $w \in \{0,1\}^{m+m'}$, we have $D_0(w) = 1$ with probability at least $1 - \varepsilon$ for a small $\varepsilon > 0$.

Let D' be an (inefficient) algorithm that takes $w \in \{0,1\}^m$ and checks whether

$$\Pr_{w' \sim \{0,1\}^{m'}, D_0}[D_0(w \circ w') = 0] \leq 2\varepsilon.$$

By Markov's inequality, with probability at least $\frac{1}{2}$ over $w \sim \{0,1\}^m$, it holds that $D'(w) = 1$. If D' $\frac{1}{2}$-avoids $G_{n,m}(x,\text{-})$, by Lemma 15, we would obtain

$$\begin{aligned}\mathsf{K}(x) &\leq \mathsf{K}^{D'}(x) + O(1)\\ &\leq m + O(\log n)\\ &= \mathsf{K}(x) - (c - O(1))\log n,\end{aligned}$$

which is a contradiction for a sufficiently large constant c. Thus, D' does not avoid $G_{n,m}(x,\text{-})$ and so there exists $z \in \{0,1\}^{O(\log n)}$ such that $D'(G_{n,m}(x,z)) = 1$. That is,

$$\mathop{\mathbf{Pr}}_{w'\sim\{0,1\}^{m'},D_0}[D_0(G_{n,m}(x,z)\circ w') = 0] \leq 2\varepsilon.$$

Next define a randomized oracle D as follows. On input $w' \in \{0,1\}^{m'}$, $D(w') = 1$ if and only if $D_0(G_{n,m}(x,z)\circ w') = 1$. Note that D $(1-2\varepsilon)$-avoids $H_{n,m'}(x,\text{-})$, and so, by Lemma 14, we obtain

$$\begin{aligned}\mathsf{rK}^{p(t),D}(x) &\leq 2m' + O(\log^3 m' + \log n)\\ &\leq O(m' + \log n).\end{aligned}$$

Finally, observe that

$$\begin{aligned}\mathsf{rK}^{t^{O(1)}}(x) &\leq \mathsf{rK}^{p(t),D}(x) + m + O(\log m)\\ &\leq m + O(m' + \log n),\end{aligned}$$

because D can be computed by hard-wiring the fixed string $G_{n,m}(x,z) \in \{0,1\}^m$. By the definitions of m and m', we obtain that

$$\mathsf{rK}^{t^{O(1)}}(x) - \mathsf{K}(x) \leq O(\mathsf{pK}^t(x) - \mathsf{K}(x) + \log n).$$

This completes the proof of the first part.

To see the "moreover" part, we compute $\tilde{m}$ such that

$$\mathsf{K}(x) - c\log n \leq \tilde{m} \leq \mathsf{pK}^{q(t)}(x) + O(\log n).$$

This can be done in randomized polynomial time by using the algorithm M. For every $m \leq \tilde{m}$, we define D_0 to be the algorithm that takes a string w of length $m + m'$ and outputs 1 if and only if $M(w, 1^{t^{O(1)}}) \leq m + m' - (c'/2)\log n$. We compute the maximum integer m such that there exists z such that $\mathbf{Pr}_{w'}[D_0(G_{n,m}(x,z)\circ w') = 0] \leq 2\varepsilon$. Note that m can be approximately computed in polynomial time by using random sampling. By the proof above, we have

$$\mathsf{K}(x) - c\log n \leq m \leq \tilde{m} \leq \mathsf{pK}^{q(t)}(x) + O(\log n).$$

Next, we compute the maximum integer m' such that $D_0(G_{n,m}(x,z)\circ H_{n,m'}(x,z')) = 1$ for all $z' \in \{0,1\}^{O(\log^3 m' + \log n)}$. This can be computed in quasi-polynomial time in m'. By the proof above, we have $m' \leq \mathsf{pK}^t(x) - m + O(\log^3 m' + \log n)$. Finally, we define the output v to be m'. As in the proof above, we obtain

$$\mathsf{rK}^{t^{O(1)}}(x) \leq m + O(m' + \log n),$$

from which it follows that

$$\begin{aligned}\mathsf{rK}^{t^{O(1)}}(x) - \mathsf{pK}^{q(t)}(x) - O(\log n) &\le \mathsf{rK}^{t^{O(1)}}(x) - m\\ &\le O(v + \log n)\\ &\le O(\mathsf{pK}^t(x) - m + \log n)\\ &\le O(\mathsf{pK}^t(x) - \mathsf{K}(x) + \log n),\end{aligned}$$

as required. ◀

3.3 Finding rK^t-Witnesses for Strings of Small Computational Depth

We call a 0-$\mathsf{rK}^t_\lambda(x)$-witness a $\mathsf{rK}^t_\lambda(x)$-witness.

▶ **Lemma 19.** *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$*, then for some polynomial* p'*, there exists a randomized polynomial-time algorithm* A *that, on input* $(x, \lambda, 1^t, 1^k)$*, outputs a list of strings that contains an* rK^t_λ*-witness of* x *with probability at least* $1-o(1)$ *over the internal randomness of* A *if*

$$\mathsf{rK}^{t/O(\log(1/(1-\lambda)))}(x) - \mathsf{pK}^{p'(t)}(x) + O(\log|x| + \log\log t + \log\log(1/(1-\lambda))) \le \log k.$$

Proof. We assume without loss of generality that $\lambda \ge 2/3$. The proof can be easily adapted to the case where $\lambda \le 2/3$.

The algorithm A operates as follows.

On input $(x, \lambda, 1^t, 1^k)$, repeat the following $k^{O(1)}$ times: Choose a uniformly random string r (of length t), run $U(z, r, x)$ for $\mathsf{poly}(t)$ steps, for each string $z \in \{0,1\}^{\le \log k}$, and add its output to the list.

To prove the correctness, let y be the lexicographically first rK^t_λ-witness of x. Note that $|y| = \mathsf{rK}^t_\lambda(x)$. By Lemma 17, we have

$$\mathsf{pK}^{p_{\mathsf{Sol}}(2t)}(y \mid x) \le \mathsf{pK}^{2t}(x, y) - \mathsf{pK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(|x| + |y|) + \log p_{\mathsf{Sol}}(\log t).$$

Observe that (x, y) can be described by y. Thus, we obtain

$$\begin{aligned}\mathsf{pK}^{2t}(x, y) &\le |y| + O(1)\\ &= \mathsf{rK}^t_\lambda(x) + O(1) && \text{(by the definition of } y\text{)}\\ &\le \mathsf{rK}^{t/O(\log(1/(1-\lambda)))}(x) + O(\log\log(1/(1-\lambda))). && \text{(by Lemma 10)}\end{aligned}$$

Combining these inequalities, we obtain

$$\begin{aligned}\mathsf{pK}^{p_{\mathsf{Sol}}(2t)}(y \mid x) &\le \mathsf{rK}^{t/O(\log(1/(1-\lambda)))}(x) - \mathsf{pK}^{p'(t)}(x)\\ &\qquad + O(\log|x| + \log\log t + \log\log(1/(1-\lambda)))\\ &\le \log k,\end{aligned}$$

which implies that A adds the witness y to its list with high probability. ◀

▶ **Lemma 20.** *Suppose* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$*. Then for every polynomial-time samplable distribution family* $\{\mathcal{D}_n\}_n$ *supported over* $\{0,1\}^n$*, there exist a polynomial* ρ*, a randomized algorithm* A*, and a time function* T *such that, for all* $n \in \mathbb{N}$*,* $\lambda \in \mathbb{R}$*, and*

$$t \ge \rho(n) \cdot \log(1/(1-\lambda))),$$

the following conditions hold:

- *For every $x \in \{0,1\}^n$, with probability at least $2/3$ over its randomness, $A(x, \lambda, 1^t)$ stops within $T(x, \lambda, t)$ steps and outputs a list of strings that contains an rK^t_λ-witness of x.*
- *For some constant $\varepsilon > 0$,*

$$\mathop{\mathbf{E}}_{x \sim \mathcal{D}_n}[T(x, \lambda, t)^\varepsilon] \le \mathsf{poly}(n, |\lambda|, t).$$

Proof. Throughout the proof, we will assume $t \ge \rho(n) \cdot \log(1/(1-\lambda)))$, for some sufficiently large polynomial ρ to be specified later.

Let p the polynomial of Theorem 18. Also, let M be the algorithm from Theorem 18, instantiated with a sufficiently large polynomial q to be specified later. Let A and p' be the algorithm and polynomial of Lemma 19, respectively.

We define a new algorithm A' as follows.

On input $(x, \lambda, 1^t)$, let t_0 be the maximum integer t_0 such that

$$t \ge p(t_0) \cdot O(\log(1/(1-\lambda))).$$

Run M on input $(x, 1^{t_0})$ to obtain $v := M(x, 1^{t_0})$, and then simulate A on input $(x, \lambda, 1^t, 1^k)$ for

$$k := 2^{O(v + \log|x| + \log\log t + \log\log(1/(1-\lambda)))}$$

and output what A outputs.

By Theorem 18, we get that with probability at least $1 - o(1)$, the value v obtained in the algorithm satisfies

$$\mathsf{rK}^{p(t_0)}(x) - \mathsf{pK}^{q(t_0)}(x) - O(\log n) \le v \le O(\mathsf{pK}^{t_0}(x) - \mathsf{K}(x) + \log n). \tag{5}$$

Therefore, our algorithm will run in time

$$T(x, \lambda, t) := 2^{O(\mathsf{pK}^{t_0}(x) - \mathsf{K}(x) + \log n + \log t + \log|\lambda|))}.$$

Also, by letting q be a sufficiently large polynomial, we have

$$\mathsf{rK}^{t/O(\log(1/(1-\lambda)))}(x) - \mathsf{pK}^{p'(t)}(x) \le \mathsf{rK}^{p(t_0)}(x) - \mathsf{pK}^{q(t_0)}(x) \le O(v + \log n).$$

Thus, we have

$$\begin{aligned} &\mathsf{rK}^{t/O(\log(1/(1-\lambda)))}(x) - \mathsf{pK}^{p'(t)}(x) + O(\log|x| + \log\log t + \log(1/(1-\lambda)))\\ &\le O(v + \log n) + O(\log|x| + \log\log t + \log\log(1/(1-\lambda)))\\ &\le \log k, \end{aligned}$$

which means that the condition of Lemma 19 is satisfied.

As a result, we get that with probability at least $2/3$, the algorithm A' runs in time $T(x, \lambda, t)$ and outputs a list of strings that contains an rK^t_λ-witness of x.

We claim that for every polynomial-time samplable distribution family $\{\mathcal{D}_n\}$, there exists a polynomial ρ such that for all large $n \in \mathbb{N}$, A' is an average-polynomial-time algorithm on input $(x, \lambda, 1^t)$ over $x \sim \mathcal{D}_n$ if $t \ge \rho(n) \cdot \log(1/(1-\lambda))$. Fix the parameters n, λ and t such that $t \ge \rho(n) \cdot \log(1/(1-\lambda)))$. We have

$$
\begin{aligned}
&\mathop{\mathbf{E}}_{x\sim D_n}[T(x,\lambda,t)^{\varepsilon}]\\
&\leq \sum_{x} \mathcal{D}_n(x)\cdot 2^{\varepsilon\cdot O(\mathsf{pK}^{t_0}(x)-\mathsf{K}(x)+\log n+\log t+\log|\lambda|))}\\
&\leq n\cdot|\lambda|\cdot t\cdot\sum_{x}\mathcal{D}_n(x)\cdot 2^{\mathsf{pK}^{t_0}(x)-\mathsf{K}(x)} && \text{(for sufficiently small } \varepsilon>0)\\
&\leq n\cdot|\lambda|\cdot t\cdot\sum_{x}2^{-\mathsf{K}(x)} && \text{(by Coding Theorem for } \mathsf{pK}^t \text{ (Theorem 8))}\\
&\leq n^{O(1)}\cdot|\lambda|\cdot t. && \text{(by ``Kraft's Inequality for } \mathsf{K}\text{'' (Lemma 7))}
\end{aligned}
$$

Note that the penultimate inequality holds for ρ that is a sufficiently large polynomial. ◀

3.4 Proof of Theorem 3

By using Lemma 20, we obtain an errorless average-case polynomial-time algorithm for finding $(1/\ell)$-rK^t-witnesses.

Proof of Theorem 13. Let $\{\mathcal{D}_n\}$ be a polynomial-time samplable distribution familiy.

Consider the algorithm A in Lemma 20. We first amplify the success probability of A, as follows. Given $(x,\lambda,1^t,1^k)$, we maintain $\mathsf{poly}(k)$ executions of $A(x,\lambda,1^t)$ *in parallel* (each with its own randomness). After half of the executions have stopped, we take the union of the outputs of these executions. By standard concentration bounds, we get an algorithm A' such that

$$\mathop{\mathbf{E}}_{x\sim D_n}[T'(x,\lambda,t,k)^{\varepsilon}]\leq \mathsf{poly}(n,|\lambda|,t,k),$$

where $\varepsilon>0$ is a constant, and T' satisfies that for all x, with probability at least $1-2^{-k}/2$ over its randomness, $A'(x,\lambda,1^t,1^k)$ stops within $T'(x,\lambda,t,k)$ steps and outputs a list of strings that contains an rK^t_λ-witness of x.

By Markov's inequality, we get that for every k, with probability at least $1-1/k$ over $x\sim\mathcal{D}_n$, $A'(x,\lambda,1^t,1^k)$ runs in time $T_k:=\mathsf{poly}(n,|\lambda|,t,k)$ and outputs a list of strings that contains an rK^t_λ-witness of x, with probability at least $1-2^{-k}/2$ (over the internal randomness of A').

Consider the algorithm A'' that, on input $(x,\lambda,1^t,1^\ell,1^k)$, simulates $A'(x,\lambda,1^t,1^{k+1})$. If it does not stop within T_k steps, we output $\perp$; otherwise, we obtain a list of programs.

Note that for every x, we will either get $\perp$ or obtain a list of programs that contains an rK^t_λ-witness of x, with probability at least $1-1/2^{-k}/2$ (over the internal randomness of A'').

Also, with probability at least $1-1/k$ over $x\sim\mathcal{D}_n$, we will obtain a list of programs that contains an rK^t_λ-witness of x, with probability at least $1-2^{-k}/2$ (over the internal randomness of A''). We aim to find an $(1/\ell)$-rK^t_λ-witness of x in this case.

We need one more tool. Given $x\in\{0,1\}^n$, a randomized program y and a time bound $t\in\mathbb{N}$, we will need to check whether y is a valid randomized program that outputs x with probability at least $\lambda-1/\ell$.

▷ Claim 21. There is a polynomial-time algorithm Valid that takes as input $(x,y,\lambda,1^t,1^\ell,1^{k'})$, where $x,y\in\{0,1\}^*$, $\lambda\in(0,1)$, and $t,\ell,k'\in\mathbb{N}$, and with probability at least $1-2^{-k'}$,

- accepts if y is a randomized program that outputs x within t steps with probability at least λ, and
- rejects if y is a randomized program that outputs x within t steps with probability less than $\lambda-1/\ell$.

Proof Sketch of Claim 21. The algorithm repeatedly simulates the randomized program y for t steps, for $\mathsf{poly}(\ell, k')$ simulations and counts the fraction of times that x is obtained. If this number is greater than $\lambda - 1/(2\ell)$, the algorithm accepts; otherwise it rejects. The correctness can be easily shown using Chernoff bounds. ◁

Using the algorithm Valid in Claim 21, we can easily obtain, from a good list output by the algorithm A'', an $(1/\ell)$-rK^t_λ-witness of x, with probability at least $1 - 2^{-k}/2$, by outputting the first y in the list so that $\mathsf{Valid}(x, y, \lambda, 1^\ell, 1^{k'})$ accepts, where k' is set appropriately.

It is easy to verify our final algorithm has polynomial running time. The correctness follows from a union bound. ◀

4 Errorless Average-Case Search-to-Decision Reduction for K^t

In this section we prove Theorem 1.

4.1 Technical Tools

The lemmas stated in this subsection are implicit in prior work, e.g., [7, 9, 5, 13]. The proof ideas are similar to those in Appendix B.1, but instead of using a generator with rK^t-style reconstruction, we use a generator with K^t reconstruction (assuming $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$). (See also Lemma 53.) We omit the details of the proofs since no new ideas are needed.

▶ **Lemma 22.** *Assume* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then there exist polynomials* p_{Sol} *and* p_0 *such that for all sufficiently large* $x, y \in \{0,1\}^*$ *and every* $t \geq p_0(|x| + |y|)$,

$$\mathsf{K}^t(x, y) > \mathsf{K}^{p_{\mathsf{Sol}}(t)}(x) + \mathsf{K}^{p_{\mathsf{Sol}}(t)}(y \mid x) - \log p_{\mathsf{Sol}}(t).$$

▶ **Lemma 23.** *Assume* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then for every polynomial-time samplable distribution family* $\{\mathcal{D}_n\}_n$, *there exists a polynomial* p_{code} *such that for every* $n \in \mathbb{N}$ *and* $x \in \mathsf{Support}(\mathcal{D}_n)$,

$$\mathsf{K}^{p_{\mathsf{code}}(n)}(x) \leq \log \frac{1}{\mathcal{D}_n(x)} + \log p_{\mathsf{code}}(n).$$

▶ **Lemma 24.** *Assume* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then there exist a constant* $c > 0$, *a polynomial* τ *and an algorithm* $\mathsf{Approx\text{-}depth}$ *that, on input* $(x, 1^{t_1}, 1^{t_2})$, *where* $x \in \{0,1\}^n$, $t_1, t_2 \in \mathbb{N}$ *with* $t_1, t_2 \geq cn$, *runs in time* $\mathsf{poly}(n, t_1, t_2)$ *and outputs an integer* s *such that*

$$\mathsf{K}^{\tau(t_1)}(x) - \mathsf{K}^{t_2}(x) \leq s \leq \mathsf{K}^{t_1}(x) - \mathsf{K}^{\tau(t_2)}(x) + \log \tau(t_1) + \log \tau(t_2).$$

4.2 Proof of Theorem 1

The following implies Theorem 1 via Proposition 11.

▶ **Theorem 25.** *Assume* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then for every polynomial-time samplable distribution family* $\{\mathcal{D}_n\}_{n \in \mathbb{N}}$, *where each* $\mathcal{D}_n$ *is over* $\{0,1\}^n$, *there exist a polynomial* ρ *and a polynomial-time algorithm* A *such that the following holds for all* $n, k \in \mathbb{N}$, *and all* $t \geq \rho(n)$.

1. *For all* $x \in \{0,1\}^n$, $A(x, 1^t, 1^k)$ *outputs either a* K^t*-witness of* x *or* $\bot$,
2. *and*

$$\Pr_{x \sim \mathcal{D}_n}\left[A(x, 1^t, 1^k) = \bot\right] \leq \frac{1}{k}.$$

Proof. Throughout the proof, we will assume that $t \geq \rho(n)$ for some polynomial ρ, which will be specified later.

Let $t \in \mathbb{N}$ be such that $t \geq p_0(3n)$, where p_0 is the polynomial from Lemma 22. Consider any $x \in \{0,1\}^n$, and let y_t be a K^t-witness of x. That is, y_t is the shortest t-time program that outputs x.

First of all, by symmetry of information (Lemma 22), there exists a polynomial p_{Sol},

$$\begin{aligned} \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(y_t \mid x) &\leq \mathsf{K}^{2t}(x, y_t) - \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) \\ &\leq |y_t| - \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + O(1) \\ &= \mathsf{K}^{t}(x) - \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + O(1) \end{aligned} \tag{6}$$

where the second inequality follows from the fact that given y_t, one can also output x within t steps.

Let $d > 0$ be some constant specified later, we say that $x \in \{0,1\}^n$ is (t,k)-*good* if

$$\mathsf{K}^{t}(x) - \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(x) \leq d \cdot \log t + \log k. \tag{7}$$

Consider any x, t, k such that x is (t,k)-good. Equation (6) implies that

$$\begin{aligned} \mathsf{K}^{t^d}(y_t \mid x) &\leq \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(y_t \mid x) \\ &\leq \mathsf{K}^{t}(x) - \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + O(1) \\ &\leq 2d \log t + \log k, \end{aligned} \tag{8}$$

provided that d is a sufficiently large constant (which depends on p_{Sol}).

Given Equation (8), we get that for some sufficiently large constant $c > d$, there is a program Π_{y_t} of length at most

$$s := c \cdot \log t + \log k \tag{9}$$

that, given x, outputs y_t within $T := t^c \cdot k^c$ steps. We aim to find such a y_t. Let A' be the following algorithm that, given $(x, 1^t)$ such that x is (t,k)-good, aims to output a K^t-witness of x.

Algorithm 1 Search for K^t-Witnesses for Good x's.

```
procedure $A'(x, 1^t)$
    $n := |x|$
    $M := 0^{2n}$
    $s := c \cdot \log t + \log k$, where $c$ is the constant from Equation (9).
    $T := t^c \cdot k^c$

    for $\Pi \in \{0,1\}^{\leq s}$ do
        $y :=$ the output of $U(\Pi, x)$ after running $T$ steps.
        if $|y| < |M|$ and $U(y)$ outputs $x$ within $t$ steps then
            $M := y$
    Output $M$
```

It is easy to verify that $A'(x, 1^t)$ runs in time $\mathsf{poly}(n, t, k)$. Next, we argue that if x is (t,k)-good, then the above algorithm outputs a K^t-witness of x.

Note that the algorithm A' always outputs some program M that can output x within t steps. Also, if x is (t,k)-good, then as described in previous paragraphs there is a program Π_{y_t} of length at most $s := c \cdot \log t + \log k$ such that $U(\Pi_{y_t}, x)$ outputs y_t within $T := t^c \cdot k^c$ steps. For such an x, we will have that $|M| \leq |y_t| = \mathsf{K}^t(x)$ when $\Pi = \Pi_{y_t}$ in the for loop.

We now describe our final algorithm A in the theorem. Let τ be the polynomial in Lemma 24, and let $\mathsf{Approx\text{-}depth}$ be the algorithm from Lemma 24. Our final algorithm A works as follows.

On input $(x, 1^t, 1^k)$, we first check if

$$\mathsf{Approx\text{-}depth}\left(x, 1^{\lfloor \tau^{-1}(t) \rfloor}, 1^{p_{\mathsf{Sol}}(2t)}\right) \leq d \cdot \log t + \log k,$$

where d is the constant in Equation (7). If yes, we output $A'(x, 1^t, 1^k)$. Otherwise, we output $\perp$.

We argue that the algorithm A above satisfies the two conditions stated in the theorem.

For the first condition, we consider two cases. Suppose x is not (t,k)-good, meaning that

$$\mathsf{K}^t(x) - \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(x) > d \cdot \log t + \log k.$$

Note that by Lemma 24, in this case $\mathsf{Approx\text{-}depth}\left(x, 1^{\lfloor \tau^{-1}(t) \rfloor}, 1^{p_{\mathsf{Sol}}(2t)}\right)$ outputs some s that satisfies

$$\begin{aligned} s &\geq \mathsf{K}^{\tau(\lfloor \tau^{-1}(t) \rfloor)}(x) - \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(x) \\ &\geq \mathsf{K}^t(x) - \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(x) \\ &> d \cdot \log t + \log k. \end{aligned}$$

Therefore, our algorithm will output $\perp$ in this case. Now suppose x is (t,k)-good. As discussed above, for such x, $A'(x, 1^t, 1^k)$ will output a K^t-witness of x. Therefore, our algorithm will always output $\perp$ or a K^t-witness of x.

For the second condition, we will show that in the above algorithm the criteria using $\mathsf{Approx\text{-}depth}$ will fail (hence output $\perp$) with probability at most $1/k$ over $x \sim \mathcal{D}_n$. To show this, we claim the following.

▷ Claim 26. For every $t, k \in \mathbb{N}$ such that $t \geq \rho(n)$, with probability at least $1 - 1/k$ over $x \sim \mathcal{D}_n$, we have

$$\mathsf{Approx\text{-}depth}\left(x, 1^{\lfloor \tau^{-1}(t) \rfloor}, 1^{p_{\mathsf{Sol}}(2t)}\right) \leq d \cdot \log t + \log k.$$

Proof of Claim 26. Recall the coding theorem for K^t (Lemma 23). By letting ρ be a sufficiently large polynomial so that for all $t \geq \rho(n)$, it is satisfied that $\lfloor \tau^{-1}(t) \rfloor \geq p_{\mathsf{code}}(n)$, where p_{code} is the quasi-polynomial from Lemma 23, we get that for every $x \in \mathsf{Support}(\mathcal{D}_n)$,

$$\mathsf{K}^{\lfloor \tau^{-1}(t) \rfloor}(x) \leq \mathsf{K}^{p_{\mathsf{code}}(n)}(x) \leq \log \frac{1}{\mathcal{D}_n(x)} + \log p_{\mathsf{code}}(n). \tag{10}$$

On the other hand, by Lemma 9, with probability at least $1 - 1/k$ over $x \sim \mathcal{D}_n$, we have

$$\mathsf{K}(x) \geq \log \frac{1}{\mathcal{D}_n(x)} - b \log n - \log k,$$

where $b > 0$ is a constant. In particular, this implies

$$\mathsf{K}^{\tau(p_{\mathsf{Sol}}(2t))}(x) \geq \mathsf{K}(x) \geq \log \frac{1}{\mathcal{D}_n(x)} - b \log n - \log k. \tag{11}$$

Finally, we get that with probability at least $1 - 1/k$ over $x \sim \mathcal{D}_n$,

$$\begin{aligned}
&\mathsf{Approx\text{-}depth}\left(x, 1^{\lfloor \tau^{-1}(t) \rfloor}, 1^{p_{\mathsf{Sol}}(2t)}\right) \\
&\leq \mathsf{K}^{\lfloor \tau^{-1}(t) \rfloor}(x) - \mathsf{K}^{\tau(p_{\mathsf{Sol}}(2t))}(x) + \log \tau(\lfloor \tau^{-1}(t) \rfloor) + \log \tau(p_{\mathsf{Sol}}(2t)) && \text{(by Lemma 24)} \\
&\leq \left(\log \frac{1}{\mathcal{D}_n(x)} + \log p_{\mathsf{code}}(n)\right) - \left(\log \frac{1}{\mathcal{D}_n(x)} - b \log n - \log k\right) + \log t + \log \tau(p_{\mathsf{Sol}}(2t)) \\
& && \text{(by Equation (10) and Equation (11))} \\
&= \log p_{\mathsf{code}\mathsf{t}}(n) + b \log n + \log k + \log t + \log \tau(p_{\mathsf{Sol}}(2t)) \\
&\leq d \cdot \log t + \log k,
\end{aligned}$$

where the last inequality holds by letting d be a sufficiently large constant. ◁

Claim 26 implies that for at least $1 - 1/k$ fraction of the x sampled from $\mathcal{D}_n$, our algorithm will output something other than $\perp$, as desired. ◀

5 Error-Prone Average-Case Search-to-Decision Reduction for Conditional K^t

In this section, we prove Theorem 2. We start with some technical tools.

5.1 Technical Tools

▶ **Lemma 27.** *Assume*

- $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$, *and*
- *infinitely-often one-way functions do not exist.*

Then for every polynomial-time samplable distribution family $\{\mathcal{C}_{\langle n,m \rangle}\}$, where each $\mathcal{C}_{\langle n,m \rangle}$ is over $\{0,1\}^m$, there exists a polynomial-time algorithm A such that for all $n, m, t, k \in \mathbb{N}$ with $t \geq n^{1.01}$, with probability at least $1 - 1/k$ over $y \sim \mathcal{C}_{\langle n,m \rangle}$,

$$\sum_{x \in \{0,1\}^n} 2^{-\mathsf{K}^t(x|y)} \cdot \mathbb{1}_{[A(x,y,1^t,1^k) \notin \mathsf{Search\text{-}MINKT}(x,y,1^t)]} \leq \frac{\mathsf{poly}(n)}{k}. \tag{12}$$

Proof. Let $c > 0$ be a constant so that $\mathsf{K}^t(x) \leq n + c$ for every $x \in \{0,1\}^n$ and $t \geq n^{1.01}$. Let S be the sampler for $\{\mathcal{C}_{\langle n,m \rangle}\}$ that takes $u := \mathsf{poly}(n, m)$ random bits.

Let f be a polynomial-time computable function defined as follows.

On input (ℓ, Π, r, r_1, r_2), where $\ell \in \{0,1\}^{\log(n+c)}$, $\Pi \in \{0,1\}^{n+c}$, $r \in \{0,1\}^u$, $r_1 \in \{0,1\}^t$ and $r_2 \in \{0,1\}^k$, we first obtain $y := \mathsf{S}(r)$. We then run $U(\Pi_{[\ell]}, y)$ for t steps and obtain a string x. If x is of length n, we output $(\ell, x, y, 1^t, 1^k)$; otherwise output $(\ell, 0^n, y, 1^t, 1^k)$.

Since we assume that $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$ and that infinitely-often one-way functions do not exist (which implies infinitely-often weak one-way functions do not exist), there is a *deterministic* polynomial-time algorithm A' such that for all $n, m, t, k \in \mathbb{N}$, it holds that

$$\mathbf{Pr}\left[A'(\ell, x, y, 1^t, 1^k) \text{ succeeds}\right] \geq 1 - \frac{1}{k^2},$$

where $(\ell, x, y, 1^t, 1^k)$ is sampled according to f and "$A'(\ell, x, y, 1^t, 1^k)$ succeeds" means $A'(\ell, x, y, 1^t, 1^k)$ outputs a pre-image of $(\ell, x, y, 1^t, 1^k)$. By an averaging argument, we get that with probability at least $1 - 1/k$ over $y \sim \mathcal{C}_{\langle n,m \rangle}$ (i.e., over $r \sim \{0,1\}^u$), it holds that

$$\mathbf{Pr}\big[A'(\ell, x, y, 1^t, 1^k) \text{ succeeds}\big] \geq 1 - \frac{1}{k}, \tag{13}$$

where the above probability is only over ℓ and x. In what follows, fix a *good* y such that Equation (13) holds.

By a union bound, Equation (13) yields that for all $\ell \in \{0,1\}^{\log(n+c)}$,

$$\mathbf{Pr}\big[A'(\ell, x, y, 1^t, 1^k) \text{ succeeds}\big] \geq 1 - \frac{n+c}{k}, \tag{14}$$

where now the probability is only over x.

Next, for any fixed ℓ, consider the following distribution $\mathcal{D}_{(y,\ell)}$:

1. Pick $\Pi \sim \{0,1\}^{n+c}$.
2. Run $U(\Pi_{[\ell]}, y)$ for t steps and obtain a string x. If x is of length n, output x; otherwise output 0^n.

Then Equation (14) implies that for all $\ell \in \{0,1\}^{\log(n+c)}$,

$$\mathop{\mathbf{Pr}}_{(x) \sim \mathcal{D}_{(y,\ell)}}\big[A'(\ell, x, y, 1^t, 1^k) \text{ fails}\big] < \frac{n+c}{k}. \tag{15}$$

Now consider the following algorithm A:

> On input $(x, y, 1^t, 1^k)$, we try $\ell = 1, 2, \ldots, n+c$, and finds the smallest ℓ such that $A'(\ell, x, y, 1^t, 1^k)$ returns some (ℓ, Π, r, r_1, r_2) for which $y = \mathsf{S}(r)$ and $U(\Pi_{[\ell]}, y)$ outputs x within t steps. Then we output $\Pi_{[\ell]}$.

We claim that the algorithm A satisfies the condition stated in Equation (12) for all good y. For the sake of contradiction, suppose there exists some good y such that

$$\sum_{x \in \{0,1\}^n} 2^{-\mathsf{K}^t(x|y)} \cdot 1_{[A(x,\, y,\, 1^t,\, 1^k)\, \notin\, \mathsf{Search\text{-}MINcKT}(x,\, y,\, 1^t)]} > \frac{n^b}{k}, \tag{16}$$

where $b > 0$ is a constant specified later.

Note that for every fixed y and ℓ, the support of $\mathcal{D}_{(y,\ell)}$ consists of only strings whose $\mathsf{K}^t(\cdot \mid y)$-complexity is at most ℓ. Also, for every $x \in \{0,1\}^n$ with $\mathsf{K}^t(x \mid y) = \ell$, $\mathcal{D}_{(y,\ell)}$ outputs x with probability at least $2^{-\mathsf{K}^t(x|y)}$. In other words, for every such x, we have

$$2^{-\mathsf{K}^t(x|y)} \leq \mathcal{D}_{(y,\ell)}(x). \tag{17}$$

Also, for every $x \in \{0,1\}^n$ with $\mathsf{K}^t(x \mid y) = \ell$, if $A'(\ell, x, y, 1^t, 1^k)$ succeeds, then $A(x, y, 1^t, 1^k) \in \mathsf{Search\text{-}MINcKT}(x, y, 1^t)$.

Then we have

$$\begin{aligned}
\frac{n^b}{k} &\leq \sum_{\ell} \sum_{x:\mathsf{K}^t(x|y)=\ell} 2^{-\mathsf{K}^t(x|y)} \cdot 1_{[A(x,\, y,\, 1^t,\, 1^k)\, \notin\, \mathsf{Search\text{-}MINcKT}(x,\, y,\, 1^t)]} && \text{(by Equation (16))} \\
&\leq \sum_{\ell} \sum_{x:\mathsf{K}^t(x|y)=\ell} \mathcal{D}_{(y,\ell)}(x) \cdot 1_{[A(x,\, y,\, 1^t,\, 1^k)\, \notin\, \mathsf{Search\text{-}MINcKT}(x,\, y,\, 1^t)]} && \text{(by Equation (17))} \\
&\leq \sum_{\ell} \sum_{x:\mathsf{K}^t(x|y)=\ell} \mathcal{D}_{(y,\ell)}(x) \cdot 1_{[A'(\ell,x,y,1^t,1^k) \text{ fails}]}.
\end{aligned}$$

By averaging, the above implies that there exists some ℓ such that

$$\sum_{x:\mathsf{K}^t(x|y)=\ell} \mathcal{D}_{(y,\ell)}(x) \cdot 1_{[A'(\ell,x,y,1^t,1^k) \text{ fails}]} \geq \frac{n^b}{(n+c)\cdot k},$$

which contradicts Equation (15) by letting b be a sufficiently large constant. ◀

▶ **Lemma 28** (Implicit in [22]). *If* $(\mathsf{MINKT}, \mathcal{U}) \in \mathsf{HeurBPP}$ *holds, then infinitely-often one-way functions do not exist.*

▶ **Lemma 29** (Following [14]; see the proof of [14, Lemma 14]). *Assume*

- $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$, *and*
- *infinitely-often one-way functions do not exist.*

Then for every polynomial-time samplable distribution family $\{\mathcal{D}_{\langle n,m\rangle}\}$ *supported over* $\{0,1\}^n \times \{0,1\}^m$, *there exists a polynomial* p *such that for all* $n, m, k \in \mathbb{N}$,

$$\Pr_{(x,y)\sim\mathcal{D}_{\langle n,m\rangle}}\left[\mathsf{K}^{p(n,m,k)}(x \mid y) \leq \log\frac{1}{\mathcal{D}_{\langle n,m\rangle}(x \mid y)} + \log p(n,m,k)\right] \geq 1 - \frac{1}{k}.$$

Proof Sketch. First of all, [14, Lemma 14] gives that if infinitely-often one-way functions do not exist, then one can get average-case coding theorem for $\mathsf{pK}^{\mathsf{poly}}$. (See also [14, Section 1.3] for an exposition). The proof here is done by "derandomizing" that of [14, Lemma 14]. More specifically, it is not hard to adapt the proof of [14, Lemma 14] to show the following. If infinitely-often one-way functions do not exist, then for every polynomial-time samplable distribution family $\{\mathcal{D}_{\langle n,m\rangle}\}$ supported over $\{0,1\}^n \times \{0,1\}^m$, there exists a *deterministic* polynomial-time algorithm Rec, such that for all $n, m, k \in \mathbb{N}$, with probability at least $1-1/k$ over $(x,y) \sim \mathcal{D}_{\langle n,m\rangle}$,

$$\Pr_{\substack{w\sim\{0,1\}^{\mathsf{poly}(n)} \\ r_{\mathsf{Rec}}\sim\{0,1\}^{\mathsf{poly}(n,m,k)}}}\left[\mathsf{Rec}(H_w(x), y, w, 1^k; r_{\mathsf{Rec}}) = x\right] \geq \frac{2}{3}, \tag{18}$$

Where H_w is a function from a pairwise independent hash family, mapping n bits to

$$s := \log\frac{1}{\mathcal{D}_{\langle n,m\rangle}(x \mid y)} + O(\log n)$$

bits, and is indexed by the string w. Moreover, given w and x, $H_w(x)$ can be computed in time $\mathsf{poly}(n)$.

Fix any (x,y) such that Equation (18) holds, we show that given y and an advice of length

$$\log\frac{1}{\mathcal{D}_{\langle n,m\rangle}(x \mid y)} + O(\log nk),$$

we can output x in time $\mathsf{poly}(n,m,k)$. This will conclude the proof of the lemma.

The idea is to derandomize Equation (18). Consider the circuit D that takes as input $w \in \{0,1\}^{\mathsf{poly}(n)}$ and $r_{\mathsf{Rec}} \in \{0,1\}^{\mathsf{poly}(n,m,k)}$, and such that

$$D(w, r_{\mathsf{Rec}}) = 1 \iff \mathsf{Rec}(H_w(x), y, w, 1^k) = x.$$

Note that D can be implemented as a circuit of size $\mathsf{poly}(n,m,k)$. Also, by Equation (18), we have

$$\Pr_{w,r_{\mathsf{Rec}}}\left[D(w, r_{\mathsf{Rec}}) = x\right] \geq \frac{2}{3}. \tag{19}$$

Assuming $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$, there is a pseudorandom generator G of seed length $O(\log s)$ that can derandomize circuits of size at most s [20]. In particular,

$$\Pr_{z \sim \{0,1\}^{O(\log(nmk))}}[D(G(z)) = 1] - \Pr_{w, r_{\mathsf{Rec}}}[D(w, r_{\mathsf{Rec}}) = 1] \geq \frac{1}{10}.$$

Together with Equation (19), the above yields that there exists some $z \in \{0,1\}^{O(\log(nmk))}$ such that for $(w, r_{\mathsf{Rec}}) := G(z)$, we have

$$\mathsf{Rec}(H_w(x), y, w, 1^k; r_{\mathsf{Rec}}) = x.$$

Note that $|H_w(x)| = s$. As a result, given y, z and $H_w(x)$, we can recover x in time $\mathsf{poly}(n, m, k)$, as desired. ◀

5.2 Proof of Theorem 2

We prove the following which implies Theorem 2.

▶ **Theorem 30.** *Assume* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. *If* $(\mathsf{MINKT}, \mathcal{U}) \in \mathsf{HeurBPP}$ *holds, then for every polynomial-time samplable distribution family* $\{\mathcal{D}_{\langle n,m\rangle}\}_{n,m\in\mathbb{N}}$ *supported over* $\{0,1\}^n \times \{0,1\}^m$*, there exist a polynomial* ρ *and a polynomial-time algorithm* A *such that for all* $n, m, k \in \mathbb{N}$*, and all* $t \geq \rho(n, m, k)$*,*

$$\Pr_{(x,y)\sim\mathcal{D}_{\langle n,m\rangle}}\left[A(x, y, 1^t, 1^k) \text{ outputs a } \mathsf{K}^t(\cdot \mid y)\text{-witness of } x\right] \geq 1 - \frac{1}{k}.$$

Proof. Let $\{\mathcal{D}_{\langle n,m\rangle}\}$ be a polynomial-time samplable distribution family. Let $\{\mathcal{C}_{\langle n,m\rangle}\}$ be the family of marginal distributions of $\{\mathcal{D}_{\langle n,m\rangle}\}$ on the second part. That is, to sample from $\mathcal{C}_{\langle n,m\rangle}$, we sample (x, y) from $\mathcal{D}_{\langle n,m\rangle}$ and then output y. Note that $\{\mathcal{C}_{\langle n,m\rangle}\}$ is polynomial-time samplable and is supported over $\{0,1\}^m$. Also, let $n, m, k \in \mathbb{N}$, and all $t \geq \rho(n, m, k)$, where ρ is a polynomial specified later.

We show how to solve $\mathsf{Search\text{-}MINcKT}$ with probability at least $1 - 1/k$ over $\mathcal{D}_{\langle n,m\rangle}$.

First of all, since we assume that $(\mathsf{MINKT}, \mathcal{U}) \in \mathsf{HeurBPP}$ holds, by Lemma 28, we get that infinitely-often one-way functions do not exist. Let A' be the polynomial-time algorithm in Lemma 27. We have that with probability at least $1 - 1/(4k)$ over $y \sim \mathcal{C}_{\langle n,m\rangle}$,

$$\sum_{x\in\{0,1\}^n} 2^{-\mathsf{K}^t(x|y)} \cdot \mathbb{1}_{[A'(x,y,1^t,1^{(nm)^b \cdot k^b}) \notin \mathsf{Search\text{-}MINcKT}(x,y,1^t)]} \leq \frac{1}{k^b \cdot (nm)^b}. \quad (20)$$

where $b > 0$ is a constant specified later.

Also, by Lemma 29 and an averaging argument, there exists a polynomial p such that, with probability at least $1 - 1/(4k)$ over $y \sim \mathcal{C}_{\langle n,m\rangle}$,

$$\Pr_{x\sim\mathcal{D}_{\langle n,m\rangle}(\cdot|y)}\left[\mathsf{K}^{p(n,m,16k^2)}(x \mid y) \leq \log\frac{1}{\mathcal{D}_{\langle n,m\rangle}(x \mid y)} + \log p(n, m, 16k^2)\right] \geq 1 - \frac{1}{4k}. \quad (21)$$

Fix any *good* y such that both Equation (20) and Equation (21) hold. Note that y is good with probability at least $1 - 1/(2k)$ when sampled from $\mathcal{C}_{\langle n,m\rangle}$. We claim that

$$\Pr_{x\sim\mathcal{D}_{\langle n,m\rangle}(\cdot|y)}\left[A'(x, y, 1^t, 1^{(nm)^b \cdot k^b}) \text{ outputs a } \mathsf{K}^t(\cdot \mid y)\text{-witness of } x\right] \geq 1 - \frac{1}{2k}. \quad (22)$$

Note that this suffices to show the theorem, since sampling $(x, y) \sim \mathcal{D}_{\langle n,m\rangle}$ is equivalent to first sampling $y \sim \mathcal{C}_{\langle n,m\rangle}$ and then sampling $x \sim \mathcal{D}_{\langle n,m\rangle}(\cdot \mid y)$.

Suppose, for the sake of contradiction, Equation (22) is not true. Then

$$\Pr_{x\sim\mathcal{D}_{\langle n,m\rangle}(\cdot|y)}\left[A'(x,y,1^t,1^{(nm)^b\cdot k^b})\notin \textsf{Search-MINcKT}(x,y,1^t)\right] > \frac{1}{2k}. \tag{23}$$

Let $\mathcal{E}(x)$ be the event that both the following hold.

- $A'(x,y,1^t,1^{(nm)^b\cdot k^b})\notin \textsf{Search-MINcKT}(x,y,1^t)$
- $\mathsf{K}^{p(n,m,16k^2)}(x\mid y)\leq \log\frac{1}{\mathcal{D}_{\langle n,m\rangle}(x|y)}+\log p(n,m,16k^2)$.

By Equation (23) and Equation (21), we get that

$$\sum_{x\in\{0,1\}^n}\mathcal{D}_{\langle n,m\rangle}(x\mid y)\cdot 1_{\mathcal{E}(x)}\geq\frac{1}{4k}. \tag{24}$$

Note that whenever $\mathcal{E}(x)$ holds, we have

$$\mathcal{D}_{\langle n,m\rangle}(x\mid y)\leq\frac{p(n,m,16k^2)}{2^{\mathsf{K}^{p(n,m,16k^2)}(x|y)}}. \tag{25}$$

Now we have

$$\begin{aligned}
\frac{1}{4k} &\leq \sum_{x\in\{0,1\}^n}\mathcal{D}_{\langle n,m\rangle}(x\mid y)\cdot 1_{\mathcal{E}(x)} && \text{(by Equation (24))}\\
&\leq \sum_{x\in\{0,1\}^n}\frac{p(n,m,16k^2)}{2^{\mathsf{K}^{p(n,m,k)}(x|y)}}\cdot 1_{\mathcal{E}(x)} && \text{(by Equation (25))}\\
&\leq p(n,m,16k^2)\cdot\sum_{x\in\{0,1\}^n}2^{-\mathsf{K}^{p(n,m,16k^2)}(x|y)}\cdot 1_{\mathcal{E}(x)}\\
&\leq p(n,m,16k^2)\cdot\sum_{x\in\{0,1\}^n}2^{-\mathsf{K}^{p(n,m,16k^2)}(x|y)}\cdot 1_{[A'(x,1^t,1^{(nm)^b\cdot k^b})\notin\textsf{Search-MINcKT}(x,1^t)]}\\
&\leq p(n,m,16k^2)\cdot\sum_{x\in\{0,1\}^n}2^{-\mathsf{K}^t(x|y)}\cdot 1_{[A'(x,y,1^t,1^{(nm)^b\cdot k^b})\notin\textsf{Search-MINcKT}(x,y,1^t)]},
\end{aligned}$$

where the last inequality holds if $t\geq p(n,m,16k^2)$. By rearranging, we get

$$\sum_{x\in\{0,1\}^n}2^{-\mathsf{K}^t(x|y)}\cdot 1_{[A'(x,y,1^t,1^{(nm)^b\cdot k^b})\notin\textsf{Search-MINcKT}(x,y,1^t)]}\geq\frac{1}{2k^2\cdot p(n,m,16k^2)}.$$

However, this contradicts Equation (20) by letting b be a sufficiently large constant. ◀

▶ Remark 31. In Theorem 30, our search algorithm only works for $t\geq\rho(n,m,k)$ instead of $t\geq\rho(n,m)$, where ρ is some polynomial (depending on the distribution family) and k is the parameter controlling the success probability of the algorithm. The reason for the dependency of k is that in the proof of Theorem 30, we need to apply the average-case conditional coding theorem (Lemma 29) with success probability at least $1-1/(4k)$ (see Equation (21)), and as a result, the time bound in the coding theorem is at least $\textsf{poly}(n,m,k)$. As shown at the end of the proof, we need t to be greater than this time bound.

6 Worst-Case to Average-Case Search-to-Decision Reductions

6.1 Worst-Case to Average-Case Search-to-Decision for rK^t

In this subsection, we show the following which implies Theorem 5 via Proposition 11.

▶ **Theorem 32.** *If* $(\mathsf{coMINKT},\mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then for every* $\varepsilon > 0$ *and every polynomial* β*, there is a probabilistic algorithm* A *such that for all* $n \in \mathbb{N}$*,* $x \in \{0,1\}^n$*, all* $\ell \in \mathbb{N}$*, and all* $\lambda \in (0,1)$ *such that* $\lambda \leq 1 - 1/2^{\mathsf{poly}(n)}$*,* $A(x, \lambda, 1^\ell)$ *runs in time* $2^{O(n/\log n)} \cdot \mathsf{poly}(|\lambda|, \ell)$ *and, with probability at least* $1 - 2^{-\ell}$*, outputs a program* M *and an integer* t *that satisfy the following:*

- $\beta(n) \leq t \leq 2^{n^\varepsilon}$*, and*
- M *is an* $(1/\ell)$*-*rK^t_λ*-witness of* x*.*

Proof. Without loss of generality, we assume $\lambda \geq 2/3$. The proof can be easily adapted to the case where $\lambda \leq 2/3$.

Let $0 < \varepsilon < 1$ and let β be a polynomial. Let $t \in \mathbb{N}$ be such that $t \geq p_0(3n) \cdot \log^2(1/(1-\lambda))$, where p_0 is the polynomial from Lemma 43. Consider any $x \in \{0,1\}^n$, and let y_t be a rK^t_λ-witness of x. That is, y_t is a program such that $U(y_t, r)$ outputs x within t steps with probability at least λ over $r \sim \{0,1\}^t$ and $|y_t| = \mathsf{rK}^t_\lambda(x)$. Also, let $q := \lceil 1/(1-\lambda) \rceil$. Note that $\log(q) \leq O(|\lambda|)$.

By symmetry of information (Lemma 43), we have, for some polynomial p_{Sol},

$$\begin{aligned}
&\mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(y_t \mid x)\\
&\leq \mathsf{rK}^{2t}(x, y_t) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(n)\\
&\leq |y_t| - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(n) + O(1)\\
&= \mathsf{rK}^t_\lambda(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(n) + O(1) && \text{(by the definition of } y_t\text{)}\\
&\leq \mathsf{rK}^t_{1-1/q}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(n) + O(1)\\
&\leq \mathsf{rK}^{t/O(\log q)}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + O(\log\log q) && \text{(by Lemma 10)}\\
&\leq \mathsf{rK}^{\sqrt{t}}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(t^2)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(n) + O(\log\log q),
\end{aligned}$$

where the second inequality follows from the fact that given y_t, one can also output x within t steps with probability at least $2/3$, and the last inequality uses that $t \geq p_0(3n) \cdot O(\log^2(1/(1-\lambda)))$. Then by the above, we have

$$\mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(y_t \mid x) \leq \mathsf{rK}^{\sqrt{t}}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(t^2)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(n) + O(\log|\lambda|). \tag{26}$$

We note that the above holds for *all* $t \geq p_0(3n) \cdot \log^2(1/(1-\lambda))$.

We claim the following.

▷ Claim 33. There is an algorithm B that, on input $x \in \{0,1\}^n$ and $\ell \in \mathbb{N}$, runs in time $O\big(2^{n^\varepsilon}\big) \cdot \mathsf{poly}(\ell)$ and with probability at least $1 - 2^{-\ell}$, outputs an integer t_{good} such that

- $\max\{p_0(3n) \cdot \log^2(1/(1-\lambda)), \beta(n)\} \leq t_{\mathsf{good}} \leq 2^{n^\varepsilon}$, and
- $\mathsf{rK}^{\sqrt{t_{\mathsf{good}}}}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}\left(t^2_{\mathsf{good}}\right)}(x) \leq dn/\log n$, where $d \geq 1$ is a constant.

Proof of Claim 33. Let $\mathsf{Approx\text{-}depth}$ be the algorithm from Lemma 47, and let $\ell' := \ell + \lceil n^{\varepsilon/2} \rceil$. Also, let $d \geq 1$ be a constant specified later.

The algorithm B works as follows.

On input $x \in \{0,1\}^n$, we enumerate all

$$t_0 \in \left[\max\{p_0(3n) \cdot \log^2(1/(1-\lambda), \beta(n)\}, 2^{n^{\varepsilon/2}}\right]$$

and consider the first t_0 such that $\mathsf{Approx\text{-}depth}(x, 1^{t_0}, 1^{\tau(p_{\mathsf{Sol}}(t_0^4))}, 1^{\ell'}) \leq dn/\log n$. If such t_0 is found, we output $t_{\mathsf{good}} := \tau(t_0^2)$, where τ is the polynomial from Lemma 47. Otherwise, we output $\perp$.

It is easy to see that the running time of this algorithm is $O\big(2^{n^\varepsilon}\big) \cdot \mathsf{poly}(\ell)$.

We now argue its correctness. First of all, by a union bound, we get that with probability except $2^{-\ell'} \cdot 2^{n^{\varepsilon/2}} \leq 2^{-\ell}$, $\mathsf{Approx\text{-}depth}(x, 1^{t_0}, 1^{p_{\mathsf{Sol}}(t_0^2)}, 1^{k'})$ will succeed (meaning that it outputs an answer that satisfies the condition stated in Lemma 47) on all $t_0 \leq 2^{n^{\varepsilon/2}}$. In what follows, we assume that this is the case.

Now consider Lemma 12 instantiated with the parameter $\varepsilon/2$, polynomials q_{dpt} such that $q_{\mathsf{dpt}}(n) \geq \max\{p_0(3n) \cdot \log^2(1/(1-\lambda)), \beta(n)\}$, and p_{dpt} such that $p_{\mathsf{dpt}}(z) \geq \tau^{(2)}(p_{\mathsf{Sol}}(z^4))$. We have that there exists some t^* such that $q_{\mathsf{dpt}}(n) \leq t^* \leq 2^{n^{\varepsilon/2}}$ and that

$$\mathsf{rK}^{t^*}(x) - \mathsf{rK}^{p_{\mathsf{dpt}}(t^*)}(x) \leq \frac{d_0 \cdot n}{\log n}, \tag{27}$$

by choosing d_0 to be a large enough constant. For such t^*, $\mathsf{Approx\text{-}depth}(x, 1^{t^*}, 1^{\tau(p_{\mathsf{Sol}}((t^*)^4))}, 1^{k'})$ outputs some s that satisfies

$$\begin{aligned} s &\leq \mathsf{rK}^{t^*}(x) - \mathsf{rK}^{\tau^{(2)}(p_{\mathsf{Sol}}((t^*)^4))}(x) + \log \tau(t^*) + \log \tau\big(p_{\mathsf{Sol}}\big((t^*)^4\big)\big) + \log^3 \tau(n) \\ &\leq \mathsf{rK}^{t^*}(x) - \mathsf{rK}^{p_{\mathsf{dpt}}(t^*)}(x) + \log \tau(t^*) + \log \tau\big(p_{\mathsf{Sol}}\big((t^*)^4\big)\big) + \log^3 \tau(n) \\ &\leq \frac{2d_0 n}{\log n}. && \text{(by Equation (27))} \end{aligned}$$

In other words, if we let $d \geq 2d_0$, there is at least one t_0 (in particular, t^*) that can pass the test using $\mathsf{Approx\text{-}depth}$. Also, by the property of $\mathsf{Approx\text{-}depth}$, for any t_0 that passes the test, we have

$$\mathsf{rK}^{\tau(t_0)}(x) - \mathsf{rK}^{\tau(p_{\mathsf{Sol}}((t_0)^4))}(x) \leq dn/\log n.$$

Recall that we will output $t_{\mathsf{good}} := \tau(t_0^2)$. Then by the above, we have

$$\mathsf{rK}^{\sqrt{t_{\mathsf{good}}}}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(t_{\mathsf{good}}^2)}(x) \leq dn/\log n,$$

as desired. ◁

Suppose we run the above algorithm B on x and obtain an integer t_{good} that satisfies the condition stated in Claim 33. Now by Equation (26), where we let $t := t_{\mathsf{good}}$, we get

$$\begin{aligned} &\mathsf{rK}^{p_{\mathsf{Sol}}(2t_{\mathsf{good}})}(y_{t_{\mathsf{good}}} \mid x) \\ &\leq \mathsf{rK}^{\sqrt{t_{\mathsf{good}}}}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(t_{\mathsf{good}}^2)}(x) + \log p_{\mathsf{Sol}}(2t_{\mathsf{good}}) + \log^3 p_{\mathsf{Sol}}(n) + O(1) \\ &\leq \frac{2dn}{\log n}, \end{aligned} \tag{28}$$

provided that d is a sufficiently large constant.

Given Equation (28) and using amplification techniques (Lemma 10), we get that for some large constant $c \geq 1$, there is a randomized program Π_{y_t} of length at most

$$s := cn/\log n + c \cdot \log \log \ell \tag{29}$$

that, given x, outputs y_t within $T := 2^{cn^{\varepsilon}} \cdot \ell^c$ steps with probability at least $1 - 2^{-\ell}/4$, where $t := t_{\mathsf{good}}$ and y_t is a rK^t-witness of x. We aim to find such a y_t.

Let Valid be the algorithm from Claim 21. Consider the following algorithm A that, on input (x, λ, ℓ), aims to output a program M and an integer t such that M is a $(1/\ell)$-$\mathsf{rK}^t_{1-\lambda}$-witness of x.

Algorithm 2 Search for rK^t-Witnesses.

```
procedure $A(x, \lambda, 1^\ell)$
    $n := |x|$
    $M := 0^{2n}$
    $s := cn/\log n + c\log\log \ell$, where $c$ is the constant from Equation (29).
    $T := 2^{cn^\varepsilon} \cdot \ell^c$

    $t := B\left(x, 1^{\ell+2}\right)$, where $B$ is the algorithm in Claim 33.

    for $\Pi \in \{0,1\}^{\leq s}$ do
        $r :=$ a uniformly random string in $\{0,1\}^T$.
        $y :=$ the output of $U(\Pi, x, r)$ after running $T$ steps.
        if $|y| < |M|$ and $\mathsf{Valid}\left(x, y, \lambda, 1^t, 1^\ell, 1^{\ell+s+3}\right)$ then
            $M := y$
    Output $M$ and $t$
```

First of all, it is easy to verify that the above algorithm runs in time $2^{O(n/\log n)} \cdot \mathsf{poly}(\ell)$. Next, we show its correctness.

Note that if the algorithm B succeeds (meaning that it returns an integer t such that there is a randomized program $\Pi_{y_t} \in \{0,1\}^{\leq s}$ that outputs y_t within T steps with probability at least $1 - 2^{-\ell}/4$, where y_t is a rK^t-witness of x), which happens with probability at least $1 - 2^{-\ell}/4$, then our algorithm will succeed if both of the following are true.

1. The algorithm Valid succeeds in all of the $m := \sum_{i=1}^{s} 2^i \leq 2^{s+1}$ executions, which happens with probability at most least $1 - 2^m \cdot 2^{-\ell-s-3} = 1 - 2^{-\ell}/4$.
2. For $\Pi = \Pi_{y_t}$, $U(\Pi, x, r)$ outputs y_t within T steps, which happens with probability at least $1 - 2^{-\ell}/4$ over $r \sim \{0,1\}^T$.

To see this, if the first item is true, then the randoized program M output by the algorithm is a "valid" one that outputs x within t steps with probability at least $\lambda - 1/\ell$. If the second item is true, then $|M| \leq |y_t| = \mathsf{rK}^t_\lambda(x)$, since $\mathsf{Valid}(x, y_t, \lambda, 1^t, 1^\ell, 1^{\ell+s+3}) = 1$ (for a successful execution of Valid).

The correctness of the algorithm then follows by a union bound. ◀

6.2 Worst-Case to Average-Case Search-to-Decision for K^t

The following implies Theorem 4 via Proposition 11.

▶ **Theorem 34.** *Assume* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then for every* $\varepsilon > 0$ *and every polynomial* β, *there is an algorithm* A *such that for all* $n \in \mathbb{N}$, $x \in \{0,1\}^n$, $A(x)$ *runs in time* $2^{O(n/\log n)}$ *and outputs a program* M *and an integer* t *that satisfy the following:*

- $\beta(n) \leq t \leq 2^{n^\varepsilon}$, *and*
- M *is a* K^t*-witness of* x.

Proof. The proof follows closely to that of Theorem 32.

Let $0 < \varepsilon < 1$ and let β be a polynomial.

Fix any $t \in \mathbb{N}$ such that $t \geq p_0(3n)$, where p_0 is the polynomial from Lemma 22. Consider any $x \in \{0,1\}^n$, and let y_t be a K^t-witness of x. That is, y_t is a shortest program such that $U(y_t)$ outputs x within t steps.

By symmetry of information (Lemma 22), we have, for some polynomial p_{Sol},

$$\begin{aligned}\mathsf{K}^{p_{\mathsf{Sol}}(2t)}(y_t \mid x) &\le \mathsf{K}^{2t}(x, y_t) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t)\\ &\le |y_t| - \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + O(1)\\ &= \mathsf{K}^{t}(x) - \mathsf{K}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + O(1) \qquad \text{(by the definition of } y_t\text{)}\end{aligned}$$

where the second inequality follows from the fact that given y_t, one can also output x within t steps. Then by the above, we have

$$\mathsf{K}^{p_{\mathsf{Sol}}(2t)}(y_t \mid x) \le \mathsf{K}^{t}(x) - \mathsf{K}^{p_{\mathsf{Sol}}(t^2)}(x) + \log p_{\mathsf{Sol}}(2t) + O(1). \tag{30}$$

We show the following claim.

▷ Claim 35. There is an algorithm B that, on input $x \in \{0,1\}^n$, runs in time $O\big(2^{n^{\varepsilon}}\big)$ and outputs an integer t_{good} such that

- $\max\{p_0(3n), \beta(n)\} \le t_{\mathsf{good}} \le 2^{n^{\varepsilon}}$, and
- $\mathsf{K}^{t_{\mathsf{good}}}(x) - \mathsf{K}^{p_{\mathsf{Sol}}\left(t_{\mathsf{good}}^2\right)}(x) \le dn/\log n$, where $d \ge 1$ is a constant.

Proof of Claim 35. Let Approx-depth be the algorithm from Lemma 24. Also, let $d \ge 1$ be a constant specified later.

The algorithm B works as follows.

On input $x \in \{0,1\}^n$, we enumerate all

$$t_0 \in \left[\max\{p_0(3n), \beta(n)\}, 2^{n^{\varepsilon/2}}\right]$$

and consider the first t_0 such that $\mathsf{Approx\text{-}depth}(x, 1^{t_0}, 1^{\tau(p_{\mathsf{Sol}}(t_0^2))}) \le dn/\log n$. If such t_0 is found, we output $t_{\mathsf{good}} := \tau(t_0)$, where τ is the polynomial from Lemma 47. Otherwise, we output $\perp$.

It is easy to verify that the running time of this algorithm is $O\big(2^{n^{\varepsilon}}\big)$. Next, we argue its correctness.

First of all, consider Lemma 12 instantiated with the parameter $\varepsilon/2$, polynomials q_{dpt} such that $q_{\mathsf{dpt}}(n) \ge \max\{p_0(3n), \beta(n)\}$, and p_{dpt} such that $p_{\mathsf{dpt}}(z) \ge \tau^{(2)}(p_{\mathsf{Sol}}(z^2))$. We have that there exists some t^* such that $q_{\mathsf{dpt}}(n) \le t^* \le 2^{n^{\varepsilon/2}}$ and that

$$\mathsf{K}^{t^*}(x) - \mathsf{K}^{p_{\mathsf{dpt}}(t^*)}(x) \le \frac{d_0 \cdot n}{\log n}, \tag{31}$$

by choosing d_0 to be a large enough constant. For such t^*, $\mathsf{Approx\text{-}depth}(x, 1^{t^*}, 1^{\tau(p_{\mathsf{Sol}}((t^*)^2))})$ outputs some s that satisfies the following.

$$\begin{aligned} s &\le \mathsf{K}^{t^*}(x) - \mathsf{K}^{\tau^{(2)}(p_{\mathsf{Sol}}((t^*)^2))}(x) + \log \tau(t^*) + \log \tau\big(p_{\mathsf{Sol}}\big((t^*)^2\big)\big)\\ &\le \mathsf{K}^{t^*}(x) - \mathsf{K}^{p_{\mathsf{dpt}}(t^*)}(x) + \log \tau(t^*) + \log \tau\big(p_{\mathsf{Sol}}\big((t^*)^2\big)\big)\\ &\le \frac{2d_0 n}{\log n}. \qquad \text{(by Equation (31))}\end{aligned}$$

In other words, if we let $d \ge 2d_0$, there is at least one t_0 (in particular, t^*) that can pass the test using Approx-depth. Also, by the property of Approx-depth, for any t_0 that passes the test, we have

$$\mathsf{K}^{\tau(t_0)}(x) - \mathsf{K}^{\tau(p_{\mathsf{Sol}}((t_0)^2))}(x) \le dn/\log n.$$

Recall that we will output $t_{\mathsf{good}} := \tau(t_0)$. Then by the above, we have

$$\mathsf{K}^{t_{\mathsf{good}}}(x) - \mathsf{K}^{p_{\mathsf{Sol}}(t_{\mathsf{good}}^2)}(x) \leq dn/\log n,$$

as desired. ◁

Suppose we run the above algorithm B on x and obtain an integer t_{good} that satisfies the condition stated in Claim 35. Now by Equation (30), where we let $t := t_{\mathsf{good}}$, we get

$$\begin{aligned} \mathsf{K}^{p_{\mathsf{Sol}}(2t_{\mathsf{good}})}(y_{t_{\mathsf{good}}} \mid x) &\leq \mathsf{K}^{t_{\mathsf{good}}}(x) - \mathsf{K}^{p_{\mathsf{Sol}}(t_{\mathsf{good}}^2)}(x) + \log p_{\mathsf{Sol}}(2t_{\mathsf{good}}) + O(1) \\ &\leq \frac{2dn}{\log n}, \end{aligned} \tag{32}$$

provided that d is a sufficiently large constant.

Given Equation (32), we get that for some large constant $c \geq 1$, there is a program Π_{y_t} of length at most

$$s := cn/\log n \tag{33}$$

that, given x, outputs y_t within $T := 2^{cn^{\varepsilon}}$ steps, where $t := t_{\mathsf{good}}$ and y_t is a K^t-witness of x. We aim to find such a y_t.

Consider the following algorithm A that, on input x, aims to output a program M and an integer t such that M is a K^t-witness of x.

Algorithm 3 Search for K^t-Witnesses.

```
procedure $A(x)$
    $n := |x|$
    $M := 0^{2n}$
    $s := cn/\log n$, where $c$ is the constant from Equation (33).
    $T := 2^{cn^{\varepsilon}}$

    $t := B(x)$, where $B$ is the algorithm in Claim 35.

    for $\Pi \in \{0,1\}^{\leq s}$ do
        $y :=$ the output of $U(\Pi, x)$ after running $T$ steps.
        if $|y| < |M|$ and $U(y)$ outputs $x$ within $t$ steps then
            $M := y$
    Output $M$ and $t$
```

First of all, it is easy to verify that the above algorithm runs in time $2^{O(n/\log n)}$. Next, we show its correctness.

Note that the algorithm B, on input x, will return a integer t that is "good" for x so that Equation (32) holds. For such t, there is some program Π_{y_t} of length at most $s := cn/\log n$ that outputs y_t, which is K^t-witness of x, within $T := 2^{cn^{\varepsilon}}$ steps. Since we enumerate all programs of size s, we will encounter Π_{y_t} and hence obtain y_t. This ensures that our algorithm can always find a t-time program for x, and the final program output by the algorithm has size at most $|y_t| = \mathsf{K}^t(x)$. ◀

References

1 Eric Allender, Joshua A. Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum circuit size, graph isomorphism, and related problems. *SIAM J. Comput.*, 47(4):1339–1372, 2018. doi:10.1137/17M1157970.

2 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. doi:10.1109/CCC.2009.12.

3 Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the theory of average case complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992. doi:10.1016/0022-0000(92)90019-F.

4 Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1), 2006. doi:10.1561/0400000004.

5 Halley Goldberg and Valentine Kabanets. A simpler proof of the worst-case to average-case reduction for polynomial hierarchy via symmetry of information. *Electron. Colloquium Comput. Complex.*, TR22-007:1–14, 2022. URL: https://eccc.weizmann.ac.il/report/2022/007, arXiv:TR22-007.

6 Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Probabilistic Kolmogorov complexity with applications to average-case complexity. In *Computational Complexity Conference* (CCC), pages 16:1–16:60, 2022. doi:10.4230/LIPIcs.CCC.2022.16.

7 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. doi:10.1109/FOCS.2018.00032.

8 Shuichi Hirahara. Characterizing average-case complexity of PH by worst-case meta-complexity. In *Symposium on Foundations of Computer Science* (FOCS), pages 50–60, 2020. doi:10.1109/FOCS46700.2020.00014.

9 Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In *Conference on Computational Complexity* (CCC), pages 20:1–20:47, 2020. doi:10.4230/LIPIcs.CCC.2020.20.

10 Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Symposium on Theory of Computing* (STOC), pages 1038–1051, 2020. doi:10.1145/3357713.3384251.

11 Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *Symposium on Theory of Computing* (STOC), pages 292–302, 2021. doi:10.1145/3406325.3451065.

12 Shuichi Hirahara. Meta-computational average-case complexity: A new paradigm toward excluding heuristica. *Bull. EATCS*, 136, 2022. URL: http://bulletin.eatcs.org/index.php/beatcs/article/view/688.

13 Shuichi Hirahara. Symmetry of information from meta-complexity. In *Computational Complexity Conference* (CCC), pages 26:1–26:41, 2022. doi:10.4230/LIPIcs.CCC.2022.26.

14 Shuichi Hirahara, Rahul Ilango, Zhenjian Lu, Mikito Nanashima, and Igor C. Oliveira. A duality between one-way functions and average-case symmetry of information. In *Symposium on Theory of Computing* (STOC), pages 1039–1050, 2023. doi:10.1145/3564246.3585138.

15 Shuichi Hirahara, Rahul Ilango, and Ryan Williams. Beating brute force for compression problems. *Electron. Colloquium Comput. Complex.*, 171:1–30, 2023. URL: https://eccc.weizmann.ac.il/report/2023/171/, arXiv:TR23-171.

16 Rahul Ilango. Connecting Perebor conjectures: Towards a search to decision reduction for minimizing formulas. In *Computational Complexity Conference* (CCC), pages 31:1–31:35, 2020. doi:10.4230/LIPIcs.CCC.2020.31.

17 Rahul Ilango, Hanlin Ren, and Rahul Santhanam. Hardness on any samplable distribution suffices: New characterizations of one-way functions by meta-complexity. *Electron. Colloquium Comput. Complex.*, page 82, 2021. URL: https://eccc.weizmann.ac.il/report/2021/082, arXiv:TR21-082.

18 Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853.

19 Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Symposium on Theory of Computing* (STOC), pages 812–821, 1990. doi:10.1109/FSCS.1990.89604.

20 Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on Theory of Computing* (STOC), pages 220–229. ACM, 1997. doi:10.1145/258533.258590.

21 Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition.* Texts in Computer Science. Springer, 2019. doi:10.1007/978-3-030-11298-1.

22 Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *Symposium on Foundations of Computer Science* (FOCS), pages 1243–1254, 2020. doi:10.1109/FOCS46700.2020.00118.

23 Yanyi Liu and Rafael Pass. One-way functions and the hardness of (probabilistic) time-bounded Kolmogorov complexity w.r.t. samplable distributions. In *Annual Cryptology Conference* (CRYPTO), pages 645–673, 2023. doi:10.1007/978-3-031-38545-2_21.

24 Luc Longpré and Osamu Watanabe. On symmetry of information and polynomial time invertibility. *Inf. Comput.*, 121(1):14–22, 1995. doi:10.1006/inco.1995.1120.

25 Zhenjian Lu and Igor C. Oliveira. An efficient coding theorem via probabilistic representations and its applications. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 94:1–94:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.94.

26 Zhenjian Lu and Igor C. Oliveira. Theory and applications of probabilistic Kolmogorov complexity. *Bull. EATCS*, 137, 2022. URL: http://bulletin.eatcs.org/index.php/beatcs/article/view/700.

27 Zhenjian Lu, Igor C. Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 92:1–92:14, 2022. doi:10.4230/LIPIcs.ICALP.2022.92.

28 Noam Mazor and Rafael Pass. A note on the universality of black-box MK^{t}P solvers. *Electron. Colloquium Comput. Complex.*, 192:1–11, 2023. URL: https://eccc.weizmann.ac.il/report/2023/192/, arXiv:TR23-192.

29 Noam Mazor and Rafael Pass. The non-uniform perebor conjecture for time-bounded Kolmogorov complexity is false. In *Innovations in Theoretical Computer Science* (ITCS), pages 80:1–80:20, 2024. doi:10.4230/LIPIcs.ITCS.2024.80.

30 Noam Mazor and Rafael Pass. Search-to-decision reductions for Kolmogorov complexity. *Electron. Colloquium Comput. Complex.*, 3:TR24–003, 2024. URL: https://eccc.weizmann.ac.il/report/2024/003.

31 Amnon Ta-Shma, Christopher Umans, and David Zuckerman. Lossless condensers, unbalanced expanders, and extractors. *Combinatorica*, 27(2):213–240, 2007. doi:10.1007/s00493-007-0053-2.

32 Boris A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *IEEE Ann. Hist. Comput.*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036.

A Symmetry of Information for pK^t

▶ **Lemma 36** (Symmetry of Information for pK^t)**.** *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then there exist polynomials* p_{Sol} *and* p_0 *such that for all sufficiently large* $x, y \in \{0,1\}^*$ *and every* $t \geq p_0(|x| + |y|)$*,*

$$\mathsf{pK}^{p_{\mathsf{Sol}}(t)}(y \mid x) \leq \mathsf{pK}^t(x, y) - \mathsf{pK}^{p_{\mathsf{Sol}}(t)}(x) + \log p_{\mathsf{Sol}}(|x| + |y|) + \log p_{\mathsf{Sol}}(\log t).$$

▶ **Definition 37** (Direct Product Generator [11, Definiton 3.10])**.** *For $k, n \in \mathbb{N}$, we define the* k-wise direct product generator *to be the function*

$$\mathsf{DP}_k \colon \{0,1\}^n \times \{0,1\}^{nk} \to \{0,1\}^{nk+k}$$

such that

$$\mathsf{DP}_k(x; z^1, \ldots, z^k) := (z^1, \ldots, z^k, x \cdot z^1, \ldots, x \cdot z^k).$$

▶ **Lemma 38** (pK^t Reconstruction Lemma [6, Lemma 22])**.** *For $\varepsilon > 0$, $x \in \{0,1\}^n$, $s \in \mathbb{N}$, and $k \in \mathbb{N}$ satisfying $k \leq 2n$, let D be a randomized algorithm that takes an advice string β and runs in time t_D such that D ε-distinguishes $\mathsf{DP}_k(x; \mathcal{U}_{nk})$ from $\mathcal{U}_{nk+k}$. Then there is a polynomial p_{DP} such that*

$$\mathsf{pK}^{\widetilde{O}(t_D) \cdot p_{\mathsf{DP}}(n/\varepsilon)}(x \mid \beta) \leq k + \log p_{\mathsf{DP}}(n/\varepsilon) + \log p_{\mathsf{DP}}(\log t_D).$$

▶ Remark 39. One difference between Lemma 38 and [6, Lemma 22] is that the pK^t bound in Lemma 38 has an additive term $O(\log\log t_D)$ instead of $O(\log t_D)$ in [6, Lemma 22], where t_D is the running time of the distinguisher. The reason why we have an additive $O(\log t_D)$ term in [6, Lemma 22] is because in the reconstruction procedure we need to encode the number t_D, which takes $O(\log t_D)$ bits. However, we can assume without loss of generality that t_D is a power of two. Hence we can encode it using only $O(\log\log t_D)$ bits.

Proof of Lemma 36. Let τ be the smallest power of two that is at least t. Note that τ can be encoded using $O(\log\log\tau)$ bits.

Let $x \in \{0,1\}^n$, $y \in \{0,1\}^\ell$, and $k, k' \in \mathbb{N}$ to be defined later. Let $\mathsf{DP}_{(-)}$ be the generator from Definition 37. Also, let $c \geq 1$ be a sufficiently large constant specified later.

To begin, observe that there exist a polynomial p_0 and a constant $d \geq 1$ such that for any $t \geq p_0(n, \ell)$, any choice of $z \in \{0,1\}^{nk}$ and $z' \in \{0,1\}^{\ell k'}$,

$$\mathsf{pK}^{2\tau}(\mathsf{DP}_k(x; z) \circ \mathsf{DP}_{k'}(y; z')) \leq \mathsf{pK}^t(x, y) + |z| + |z'| + d \log(n\ell). \tag{34}$$

In particular, $p_0(n, \ell)$ reflects the time required to deterministically compute $\mathsf{DP}_k(x; z) \circ \mathsf{DP}_{k'}(y; z')$ given xy, z, z', and $d\log(n\ell)$ bits of information to delineate x from y. In what follows, we will give a lower bound on $\mathsf{pK}^{2\tau}(\mathsf{DP}_k(x; z) \circ \mathsf{DP}_{k'}(y; z'))$ and thereby a lower bound on $\mathsf{pK}^t(x, y)$.

Since we assume that $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ holds, there exist a constant $c' > 0$, a polynomial ρ and a probabilistic polynomial-time algorithm B such that the following hold for all sufficiently large n', all $t' \geq \rho(n')$, and $s' \leq n' - c' \cdot \log\log t'$, and .

1. If $r \in \{0,1\}^{n'}$ and $\mathsf{K}^{t'}(r) \leq s'$, then $\mathbf{Pr}_B[B(r, 1^{s'}, 1^{t'}) = 1] \geq 1 - \frac{1}{10n'}$.
2. With probability at least $1/n'$ over $r \sim \{0,1\}^{n'}$, $\mathbf{Pr}_B[B(r, 1^{s'}, 1^{t'}) = 0] \geq 1 - \frac{1}{10n'}$.

Let $c > 0$ be a sufficiently large constant to be specified later, and consider the following parameters.

- $k := \mathsf{pK}^{q(\tau)}(x) - \log q(\log\tau) - \log p_G(n\ell) - 1$ and $k' := \mathsf{pK}^{q(\tau)}(y \mid x) - \log q(\log\tau) - \log q(n\ell) - 1$, where q is a sufficiently large polynomial specified later.
- $n' := nk + k + \ell k' + k' + \tau^c$.
- $s' := nk + k + \ell k' + k' + \tau^c - c \cdot \log\log\tau - c \cdot \log(n\ell)$.
- $t' := \tau^{2c}$.

We show the following which will imply a lower bound on $\mathsf{rK}^{2t}(\mathsf{DP}_k(x; z) \circ \mathsf{DP}_{k'}(y; z'))$.

▷ Claim 40. There exist $z \in \{0,1\}^{nk}$ and $z' \in \{0,1\}^{\ell k'}$ such that

$$\Pr_{w\sim\{0,1\}^{t^c}}\left[\mathsf{K}^{t'}(\mathsf{DP}_k(x;z)\circ\mathsf{DP}_{k'}(y;z')\circ w)\leq s'\right] < 1-\frac{1}{10n'}.$$

Proof of Claim 40. We first claim the following:

$$\Pr_{z,v,w,B}\left[B(\mathsf{DP}_k(x;z)\circ vw, 1^{s'}, 1^{t'})=1\right] \leq 1-\frac{4}{10n'}. \tag{35}$$

Toward a contradiction, suppose

$$\Pr_{z,v,w,B}\left[B(\mathsf{DP}_{k'}(x;z)\circ vw, 1^{s'}, 1^{t'})=1\right] > 1-\frac{4}{10n'}. \tag{36}$$

By the property of the algorithm B (Item 2), we have

$$\Pr_{u,v,w,B}\left[B(uvw, 1^{s'}, 1^{t'})=0\right] \geq \frac{1}{2n'}.$$

In this case, comparing with Equation (36), we get a randomized distinguisher for $\mathsf{DP}_k(x;\mathcal{U}_{nk})$ with advantage $1/10n'$, defined by sampling $v\sim\mathcal{U}_{\ell k'+k'}$, $w\sim\mathcal{U}_{t^c}$, and outputting $B(-\circ vw, 1^{s'}, 1^{t'})$. By Lemma 38, there exists some polynomial q such that

$$\mathsf{pK}^{q(\tau)}(x)\leq k+\log q(\log\tau)+\log p_{_G}(n\ell). \tag{37}$$

Recall that $k=\mathsf{pK}^{q(t)}(x)-\log q(\log\tau)-\log q(n\ell)-1$, so Equation (37) gives a contradiction. This shows Equation (35).

Now, toward a contradiction, suppose that for *all* z, z',

$$\Pr_{w}\left[\mathsf{K}^{t'}(\mathsf{DP}_k(x;z)\circ\mathsf{DP}_{k'}(y;z')\circ w)\leq s'\right] \geq 1-\frac{1}{10n'}.$$

By the property of B (Item 1), this implies that

$$\Pr_{z,z',w,B}\left[B(\mathsf{DP}_k(x;z)\circ\mathsf{DP}_{k'}(y;z'))\circ w, 1^{s'}, 1^{t'})=1\right] \geq 1-\frac{2}{10n'}.$$

In this case, comparing with Equation (35), we get a randomized distinguisher for $\mathsf{DP}_{k'}(y;\mathcal{U}_{\ell k'})$ with advantage $(2/10n')$, defined by sampling $z\sim\mathcal{U}_{k'}$, $w\sim\mathcal{U}_{\tau^c}$, and outputting $B(\mathsf{DP}_k(x;z)\circ - \circ w, 1^{s'}, 1^{t'})$. Again, by Lemma 38, we have

$$\mathsf{pK}^{q(\tau)}(y\mid x)\leq k'+\log q(\log\tau)+\log q(n\ell). \tag{38}$$

Recall that $k'=\mathsf{pK}^{q(\tau)}(y\mid x)-\log q(\tau)-\log q(n\ell)-1$, so that Equation (38) gives a contradiction. This completes the proof of Claim 40. ◁

Next, we show that Claim 40 implies there exist z, z' such that

$$\mathsf{pK}^{\tau^c}_{1-1/10n'}(\mathsf{DP}_k(x;z)\circ\mathsf{DP}_{k'}(y;z')) > s =: |z|+k+|z'|+k'-2c\cdot\log\log\tau.$$

Suppose this is not the case. Then there exists a *deterministic* program M of length at most s such that $U(M,w)$ outputs $\mathsf{DP}_k(x;z)\circ\mathsf{DP}_{k'}(y;z')$ within τ^c steps for at least $1-1/(10n')$ of the $w\in\{0,1\}^{\tau^c}$, which implies

$$\Pr_{w\sim\{0,1\}^{\tau^c}}\left[\mathsf{K}^{\tau^{2c}}(\mathsf{DP}_k(x;z)\circ\mathsf{DP}_{k'}(y;z')\circ w)\leq s+\tau^c+O(\log(c\cdot\log\tau))\right] \geq 1-\frac{1}{10n'}. \tag{39}$$

On the other hand, we have

$$\begin{aligned} s + \tau^c + O(\log(c \log \tau)) &= (nk + k + \ell k' + k' - 2c \cdot \log\log\tau) + \tau^c + O(\log(c \cdot \log \tau)) \\ &\leq s', \end{aligned} \tag{40}$$

where the last inequality holds if we choose c to be a sufficiently large constant. Equation (39) and Equation (40) together imply that

$$\mathop{\mathbf{Pr}}_{w \sim \{0,1\}^{\tau^c}}\left[\mathsf{K}^{t'}(\mathsf{DP}_k(x;z) \circ \mathsf{DP}_{k'}(y;z') \circ w) \leq s'\right] \geq 1 - \frac{1}{10n'},$$

which contradicts Claim 40.

Therefore, there exist z, z' such that

$$\mathsf{pK}^{\tau^c}_{1-1/10n'}(\mathsf{DP}_k(x;z) \circ \mathsf{DP}_{k'}(y;z')) > |z| + k + |z'| + k' - 2c \cdot \log\log\tau.$$

By amplification techniques (Lemma 10), the above implies

$$\mathsf{pK}^{2\tau}(\mathsf{DP}_k(x;z) \circ \mathsf{DP}_{k'}(y;z')) > |z| + k + |z'| + k' - 2c \cdot \log\log\tau - O(\log\log n'). \tag{41}$$

Finally, we get

$$\begin{aligned} \mathsf{pK}^t(x,y) &\geq \mathsf{pK}^{2\tau}(\mathsf{DP}_k(x;z) \circ \mathsf{DP}_{k'}(y;z')) - |z| - |z'| - d\log(n\ell) && \text{(by Equation (34))} \\ &> k + k' - 2c \cdot \log\log\tau - O(\log\log n') - d\log(n\ell) && \text{(by Equation (41))} \\ &= \left(\mathsf{pK}^{q(\tau)}(x) - \log q(\log \tau) - \log p_G(n\ell) - 1\right) \\ &\quad + \left(\mathsf{pK}^{q(\tau)}(y \mid x) - \log q(\log\tau) - \log q(n\ell) - 1\right) \\ &\quad - 2c \cdot \log\log\tau - O(\log\log n') - d\log(n\ell) \\ &\geq \mathsf{pK}^{p_{\mathsf{Sol}}(t)}(x) + \mathsf{pK}^{p_{\mathsf{Sol}}(t)}(y \mid x) - \log p_{\mathsf{Sol}}(|x| + |y|) - \log p_{\mathsf{Sol}}(\log t), \end{aligned}$$

where the last inequality holds by letting p_{Sol} be a large enough polynomial. ◀

B Quasi-Polynomial-Time Average-Case Search-to-Decision Reduction for rK^t

We introduce the following statement.

"$\mathsf{MINrKT} \in \mathsf{AvgBPTIME}[2^{O(\log^3 n)}]$": For every polynomial-time samplable distribution family $\{\mathcal{D}_n\}_{n \in \mathbb{N}}$, where each $\mathcal{D}_n$ is over $\{0,1\}^n$, there exist a polynomial ρ and a probabilistic algorithm A such that the following hold for all all $\lambda \in (0,1)$, all $n, s, \ell, k \in \mathbb{N}$, and all $t \geq \rho(n) \cdot \log(1/(1-\lambda)))$.

1. For all $x \in \{0,1\}^n$, $A(x, \lambda, 1^t, 1^\ell, 1^k)$ runs in time $2^{O(\log^3 n)} \cdot \mathsf{poly}(|\lambda|, t, \ell, k)$.
2. For all $x \in \{0,1\}^n$,

$$\mathop{\mathbf{Pr}}_A\left[A(x, \lambda, 1^t, 1^\ell, 1^k) \text{ outputs either an } (1/\ell)\text{-}\mathsf{rK}^t_\lambda\text{-witness of } x \text{ or } \bot\right] \geq 1 - \frac{1}{2^k}.$$

3. With probability at least $1 - 1/k$ over $x \sim \mathcal{D}_n$,

$$\mathop{\mathbf{Pr}}_A\left[A(x, \lambda, 1^t, 1^\ell, 1^k) \text{ outputs an } (1/\ell)\text{-}\mathsf{rK}^t_\lambda\text{-witness of } x\right] \geq 1 - \frac{1}{2^k}.$$

In this section we prove the following quasipolynomial-time version of Theorem 3.

▶ **Theorem 41.** *We have*

$$\text{“}\mathsf{MINrKT} \in \mathsf{AvgBPP}\text{”} \implies \text{“}\mathsf{MINrKT} \in \mathsf{AvgBPTIME}[2^{O(\log^3 n)}]\text{”}.$$

B.1 Technical Tools

We begin with some technical tools.

B.1.1 A Generator with rK^t Reconstruction

We will use the following pseudorandom generator construction.

▶ **Lemma 42** (see e.g., [13] and [14, Lemma 26]). *There exists a polynomial p such that, for all sufficiently large $n, m, t \in \mathbb{N}$ such that $m \le 2n$ and $t \ge n$, there exists a "pseudorandom generator construction"*

$$G_m \colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$$

such that for every $x \in \{0,1\}^n$ and any function $D \colon \{0,1\}^m \times \{0,1\}^t \to \{0,1\}$, if

$$\left| \Pr_{\substack{z\sim\{0,1\}^d \\ w'\sim\{0,1\}^t}}[D(G_m(x;z);w')=1] - \Pr_{\substack{w\sim\{0,1\}^m \\ w'\sim\{0,1\}^t}}[D(w;w')=1] \right| \ge \frac{1}{m},$$

then

$$\mathsf{rK}^{p(t),D}(x) \le m + O(\log^3 n).$$

Here, $d = O(\log^3 n)$ and G_m can be computed in time $\mathsf{poly}(n)$.

B.1.2 Symmetry of Information for rK^t

▶ **Lemma 43** (Symmetry of Information for rK^t). *If $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ holds, then there exist polynomials p_{Sol} and p_0 such that for all sufficiently large $x, y \in \{0,1\}^*$ and every $t \ge p_0(|x|+|y|)$,*

$$\mathsf{rK}^t(x,y) > \mathsf{rK}^{p_{\mathsf{Sol}}(t)}(x) + \mathsf{rK}^{p_{\mathsf{Sol}}(t)}(y \mid x) - \log p_{\mathsf{Sol}}(t) - \log^3 p_{\mathsf{Sol}}(|x|\cdot|y|).$$

Proof. The proof follows closely to that of Lemma 36.

Let $x \in \{0,1\}^n$, $y \in \{0,1\}^\ell$, and $m, m' \in \mathbb{N}$ to be defined later. Let $G_{(-)}$ be the generator from Lemma 42. Also, let $c \ge 1$ be a sufficiently large constant specified later.

To begin, observe that there exist a polynomial p_0 and a constant $d \ge 1$ such that for any $t \ge p_0(n,\ell)$, any choice of $z \in \{0,1\}^{O(\log^3 n)}$ and $z' \in \{0,1\}^{O(\log^3 \ell)}$,

$$\mathsf{rK}^{2t}(G_m(x;z) \circ G_{m'}(y;z')) \le \mathsf{rK}^t(x,y) + |z| + |z'| + d\log t. \tag{42}$$

In particular, $p_0(n,\ell)$ reflects the time required to deterministically compute $G_m(x;z) \circ G_{m'}(y;z')$ given xy, z, z', and $d\log t$ bits of information to delineate x from y. In what follows, we will give a lower bound on $\mathsf{rK}^{2t}(G_m(x;z) \circ G_{m'}(y;z'))$ and thereby a lower bound on $\mathsf{rK}^t(x,y)$.

Since we assume that $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ holds, there exist a constant $c' > 0$, a polynomial ρ and a probabilistic polynomial-time algorithm B such that the following hold for all sufficiently large n', all all $t' \ge \rho(n')$, and all $s' \le n' - c' \cdot \log\log t'$.

1. If $r \in \{0,1\}^{n'}$ and $\mathsf{K}^{t'}(r) \le s'$, then $\mathbf{Pr}_B[B(r, 1^{s'}, 1^{t'}) = 1] \ge 1 - \frac{1}{10n'}$.
2. With probability at least $1/n'$ over $r \sim \{0,1\}^{n'}$, $\mathbf{Pr}_B[B(r, 1^{s'}, 1^{t'}) = 0] \ge 1 - \frac{1}{10n'}$.

Let $c > 0$ be a sufficiently large constant to be specified later, and consider the following parameters.

- $m := \mathsf{rK}^{p_G(t)}(x) - \log p_G(t) - \log^3 p_G(n\ell) - 1$ and $m' := \mathsf{rK}^{p_G(t)}(y \mid x) - \log p_G(t) - \log^3 p_G(n\ell) - 1$, where p_G is a sufficiently large polynomial specified later.
- $n' := m + m' + t^c$.
- $s' := m + m' + t^c - c^2 \cdot \log(t) - c \cdot \log^3(n\ell)$.
- $t' := t^{2c}$.

We show the following which will imply a lower bound on $\mathsf{rK}^{2t}(G_m(x;z) \circ G_{m'}(y;z'))$.

▷ Claim 44. There exist $z \in O(\log^3 n)$ and $z' \in O(\log^3 \ell)$ such that

$$\Pr_{w \sim \{0,1\}^{t^c}}\left[\mathsf{K}^{t'}(G_m(x;z) \circ G_{m'}(y;z') \circ w) \le s'\right] < 1 - \frac{1}{10n'}.$$

Proof of Claim 44. We first claim the following:

$$\Pr_{z,v,w,B}\left[B(G_m(x;z) \circ vw, 1^{s'}, 1^{t'}) = 1\right] \le 1 - \frac{4}{10n'}. \tag{43}$$

Toward a contradiction, suppose

$$\Pr_{z,v,w,B}\left[B(G_m(x;z) \circ vw, 1^{s'}, 1^{t'}) = 1\right] > 1 - \frac{4}{10n'}. \tag{44}$$

By the property of the algorithm B (Item 2), we have

$$\Pr_{u,v,w,B}\left[B(uvw, 1^{s'}, 1^{t'}) = 0\right] \ge \frac{1}{2n'}.$$

In this case, comparing with Equation (44), we get a randomized distinguisher for $G_m(x;\mathcal{U}_{O(\log^3 n)})$ with advantage $1/10n'$, defined by sampling $v \sim \mathcal{U}_{m'}$, $w \sim \mathcal{U}_{t^c}$, and outputting $B(- \circ vw, 1^{s'}, 1^{t'})$. By Lemma 42, there exists some polynomial p_G such that

$$\mathsf{rK}^{p_G(t)}(x) \le m + \log p_G(t) + \log^3 p_G(n\ell). \tag{45}$$

Recall that $m = \mathsf{rK}^{p_G(t)}(x) - \log p_G(t) - \log^3 p_G(n\ell) - 1$, so Equation (45) gives a contradiction. This shows Equation (43).

Now, toward a contradiction, suppose that for *all* z, z',

$$\Pr_{w}\left[\mathsf{K}^{t'}(G_m(x;z) \circ G_{m'}(y;z') \circ w) \le s'\right] \ge 1 - \frac{1}{10n'}.$$

By the property of B (Item 1), this implies that

$$\Pr_{z,z',w,B}\left[B(G_m(x;z) \circ G_{m'}(y;z')) \circ w, 1^{s'}, 1^{t'}) = 1\right] \ge 1 - \frac{2}{10n'}.$$

In this case, comparing with Equation (43), we get a randomized distinguisher for $G_{m'}(y;\mathcal{U}_{O(\log^3 \ell)})$ with advantage $(2/10n')$, defined by sampling $z \sim \mathcal{U}_{O(\log^3 \ell)}$, $w \sim \mathcal{U}_{t^c}$, and outputting $B(G_m(x;z) \circ - \circ w, 1^{s'}, 1^{t'})$. Again, by Lemma 42, we have

$$\mathsf{rK}^{p_G(t)}(y \mid x) \le m' + \log p_G(t) + \log^3 p_G(n\ell). \tag{46}$$

Recall that $m' = \mathsf{rK}^{p_G(t)}(y \mid x) - \log p_G(t) - \log^3 p_G(n\ell) - 1$, so that Equation (46) gives a contradiction. This completes the proof of Claim 44. ◁

Next, we show that Claim 44 implies there exist z, z' such that

$$\mathsf{rK}^{t^c}_{1-1/10n'}(G_m(x;z) \circ G_{m'}(y;z')) > s =: |z| + m + |z'| + m' - c^3 \log(t).$$

Suppose this is not the case. Then there exists a *deterministic* program M of length at most s such that $U(M, w)$ outputs $G_m(x;z) \circ G_{m'}(y;z')$ within t^c steps for at least $1 - 1/(10n')$ of the $w \in \{0,1\}^{t^c}$, which implies

$$\mathop{\mathbf{Pr}}_{w \sim \{0,1\}^{t^c}}\left[\mathsf{K}^{t^{2c}}(G_m(x;z) \circ G_{m'}(y;z') \circ w) \leq s + t^c + O(c \log t)\right] \geq 1 - \frac{1}{10n'}. \tag{47}$$

On the other hand, we have

$$\begin{aligned} s + t^c + O(c \log t) &= \big(m + m' + O(\log^3 n) + O(\log^3 \ell) - c^3 \log(t)\big) + t^c + O(c \log t) \\ &\leq s', \end{aligned} \tag{48}$$

where the last inequality holds if we choose c to be a sufficiently large constant. Equation (47) and Equation (48) together imply that

$$\mathop{\mathbf{Pr}}_{w \sim \{0,1\}^{t^c}}\left[\mathsf{K}^{t'}(G_m(x;z) \circ G_{m'}(y;z') \circ w) \leq s'\right] \geq 1 - \frac{1}{10n'},$$

which contradicts Claim 44.

Therefore, there exist z, z' such that

$$\mathsf{rK}^{t^c}_{1-1/10n'}(G_m(x;z) \circ G_{m'}(y;z')) > |z| + m + |z'| + m' - c^2 \log(t).$$

By amplification techniques (Lemma 10), the above implies

$$\mathsf{rK}^{2t}(G_m(x;z) \circ G_{m'}(y;z')) > |z| + m + |z'| + m' - c^2 \log(t) - O(\log\log n'). \tag{49}$$

Finally, we get

$$\begin{aligned} \mathsf{rK}^t(x,y) &\geq \mathsf{rK}^{2t}\big(G_m(x;z) \circ G_{m'}(y;z')\big) - |z| - |z'| - d \log t && \text{(by Equation (42))} \\ &> m + m' - c\log^3(n\ell) - c^3 \log(t) - O(\log\log n') - d \log t && \text{(by Equation (49))} \\ &= \big(\mathsf{rK}^{p_G(t)}(x) - \log p_G(t) - \log^3 p_G(n\ell) - 1\big) + \big(\mathsf{rK}^{p_G(t)}(y \mid x) - \log p_G(t) - \log^3 p_G(n\ell) - 1\big) \\ &\quad - c\log^3(n\ell) - c^3 \log(t) - O(\log\log n') - d \log t \\ &\geq \mathsf{rK}^{p_{\mathsf{Sol}}(t)}(x) + \mathsf{rK}^{p_{\mathsf{Sol}}(t)}(y \mid x) - \log p_{\mathsf{Sol}}(t) - \log^3 p_{\mathsf{Sol}}(n\ell), \end{aligned}$$

where the last inequality holds by letting p_{Sol} be a large enough polynomial. ◀

B.1.3 Coding Theorem for rK^t

▶ **Lemma 45** (An Efficient Coding Theorem for rK^t). *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then for every polynomial-time samplable distribution family* $\{\mathcal{D}_n\}_n$*, there exists a polynomial* p_{code} *such that for every* $n \in \mathbb{N}$ *and* $x \in \mathsf{Support}(\mathcal{D}_n)$*,*

$$\mathsf{rK}^{p_{\mathsf{code}}(n)}(x) \leq \log \frac{1}{\mathcal{D}_n(x)} + \log^3 p_{\mathsf{code}}(n).$$

We need the following technical lemma.

▶ **Lemma 46** ([2, 1]; See also [11, Lemma 9.7]). *Let $\{\mathcal{D}_n\}_n$ be any polynomial-time samplable family of distributions. Then, there exist polynomials p and q such that, for every $n \in \mathbb{N}$ and every $x \in \mathsf{Support}(\mathcal{D}_n)$,*

$$\Pr_{r\sim\{0,1\}^{q(n)}}\left[\mathsf{K}^{p(n)}(x,r) \le \frac{1}{\mathcal{D}_n(x)} + |r| + \log p(n)\right] \ge \frac{1}{4}.$$

We now show Lemma 45.

Proof of Lemma 45. The proof is essentially the same as that of [11, Corollary 9.8].

Note that for any $x \in \{0,1\}^*$ and $t \in \mathbb{N}$, we have $\mathsf{rK}^t(x) \le \mathsf{K}^t(x)$. On the one hand, by Lemma 46, we have for some polynomials p and q and for every $x \in \mathsf{Support}(\mathcal{D}_n)$,

$$\Pr_{r\sim\{0,1\}^{q(n)}}\left[\mathsf{rK}^{p(n)}(x,r) \le \frac{1}{\mathcal{D}_n(x)} + |r| + \log p(n)\right] \ge \frac{1}{4}. \tag{50}$$

On the other hand, by symmetry of information (Lemma 43), for every $x \in \{0,1\}^n$ and $r \in \{0,1\}^{q(n)}$, we have

$$\mathsf{rK}^{p(n)}(x,r) \ge \mathsf{rK}^{p_{\mathsf{SoI}}(p(n))}(x) + \mathsf{rK}^{p_{\mathsf{SoI}}(p(n))}(r \mid x) - \log p_{\mathsf{SoI}}(p(n)) - \log^3 p_{\mathsf{SoI}}(n \cdot q(n)). \tag{51}$$

Also, by a simple counting argument, we have for any fixed $x \in \{0,1\}^*$,

$$\Pr_{r\sim\{0,1\}^{q(n)}}\left[\mathsf{rK}^{p_{\mathsf{SoI}}(p(n))}(r \mid x) \ge |r| - \log n\right] > \frac{3}{4}. \tag{52}$$

Combining Equations (51) and (52), we get that with probability greater than 3/4,

$$\mathsf{rK}^{p(n)}(x,r) \ge \mathsf{rK}^{p_{\mathsf{SoI}}(p(n))}(x) + |r| - \log p_{\mathsf{SoI}}(p(n)) - \log^3 p_{\mathsf{SoI}}(n \cdot q(n)) - \log n,$$

which, together with Equation (50), implies that there exists some r such that

$$\mathsf{rK}^{p_{\mathsf{SoI}}(p(n))}(x) \le \log \frac{1}{\mathcal{D}_n(x)} + \log p(n) + \log p_{\mathsf{SoI}}(p(n)) + \log^3 p_{\mathsf{SoI}}(n \cdot q(n)) + \log n.$$

The desired conclusion follows by choosing p_{code} to be a sufficiently large polynomial. ◀

B.1.4 Approximate Computational Depth for rK^t

In this subsection, we show an algorithm that can approximate the (randomized) computational depth of a given string.

▶ **Lemma 47.** *If $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ holds, then there exist a constant $c > 0$, a polynomial τ and an algorithm $\mathsf{Approx\text{-}depth}$ that, on input $(x, 1^{t_1}, 1^{t_2}, 1^k)$, where $x \in \{0,1\}^n$, $t_1, t_2, k \in \mathbb{N}$ with $t_1, t_2 \ge cn$, runs in time $\mathsf{poly}(n, t_1, t_2, k)$ and with probability $1-2^{-k}$ outputs an integer s such that*

$$\mathsf{rK}^{\tau(t_1)}(x) - \mathsf{rK}^{t_2}(x) \le s \le \mathsf{rK}^{t_1}(x) - \mathsf{rK}^{\tau(t_2)}(x) + \log \tau(t_1) + \log \tau(t_2) + \log^3 \tau(n).$$

To show Lemma 47, we need the following “worst-case-to-average-case reduction” result.

▶ **Lemma 48.** *If $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ holds, then there exists a polynomial τ such that the following promise problem is in prBPP:*

$$\mathsf{YES} := \left\{(x, 1^s, 1^t) \mid \mathsf{rK}^t(x) \le s\right\},$$
$$\mathsf{NO} := \left\{(x, 1^s, 1^t) \mid \mathsf{rK}^{\tau(t)}(x) > s + \log \tau(t) + \log^3 \tau(n)\right\}.$$

Proof. Fix an instance $(x, 1^s, 1^t)$, where $x \in \{0,1\}^n$. Without loss generality, we assume that $t \geq |x|$. Let $G_{(-)}$ be the generator from Lemma 42.

Since we assume that $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ holds, There exist a constant $c' > 0$, a polynomial ρ and a probabilistic polynomial-time algorithm B such that the following hold for all sufficiently large n', all$t' \geq \rho(n')$ and all $s' \leq n' - c' \cdot \log\log t'$, and .

1. If $y \in \{0,1\}^{n'}$ and $\mathsf{K}^{t'}(y) \leq s'$, then $\mathbf{Pr}_B[B(y, 1^{s'}, 1^{t'}) = 1] \geq 1 - \frac{1}{10n'}$.
2. With probability at least $1/n'$ over $y \in \{0,1\}^{n'}$, $\mathbf{Pr}_B[B(y, 1^{s'}, 1^{t'}) = 0] \geq 1 - \frac{1}{10n'}$.

Let $c > 0$ be a sufficiently large constant and consider the following parameters.

- $m := s + c^3 \cdot \log t + c \log^3 n$.
- $n' := m + t^c$.
- $s' := s + t^c + c^2 \cdot \log t + c \log^3 n$.
- $t' := t^{2c}$.

Now, define an algorithm B' as follows:

> On input $(x, 1^s, 1^t)$ with $x \in \{0,1\}^n$, sample $z \sim \{0,1\}^{O(\log^3 n)}$ and $w \sim \{0,1\}^{t^c}$, and then output $B\Big(G_m(x; z) \circ w, 1^{s'}, 1^{t'}\Big)$.

Below, we show that B' solves $(\mathsf{YES}, \mathsf{NO})$ correctly with high probability in the worst case.

First, consider the case that $(x, 1^s, 1^t) \in \mathsf{YES}$, i.e., $\mathsf{rK}^t(x) \leq s$. By amplification techniques (Lemma 10), we get that

$$\mathsf{rK}^{t^c}_{1-1/10n'}(x) \leq s + O(\log\log n').$$

In other words, there exists a *deterministic* program M of length at most $s + O(\log\log n')$ such that $U(M, w)$ outputs x within t^c steps for at least $1 - 1/10n'$ of the $w \in \{0,1\}^{t^c}$. This implies that for any choice of $z \in \{0,1\}^{O(\log^3 n)}$,

$$\mathop{\mathbf{Pr}}_{w \sim \{0,1\}^{t^c}}\Big[\mathsf{K}^{t^{2c}}(G_m(x; z) \circ w) \leq s + O(\log\log n') + t^c + O(\log^3 n) + O(c \log t)\Big] \geq 1 - \frac{1}{10n'}. \tag{53}$$

Also, note that by letting c be a sufficiently large constant, we have

$$s + O(\log\log n') + t^c + O(\log^3 n) + O(c \log t) \leq s' \tag{54}$$

Equation (53) and Equation (54) together imply that

$$\mathop{\mathbf{Pr}}_{w \sim \{0,1\}^{t^c}}\Big[\mathsf{K}^{t'}(G_m(x; z) \circ w) \leq s'\Big] \geq 1 - \frac{1}{10n'}.$$

Then by the property of B (Item 1) and a union bound, we have

$$\mathop{\mathbf{Pr}}_{w,z,B}\Big[B(G_m(x; z) \circ w, 1^{s'}, 1^{t'}) = 1\Big] \geq 1 - \frac{1}{5n'},$$

and so

$$\mathop{\mathbf{Pr}}_{B'}\big[B'(x, 1^s, 1^t) = 1\big] \geq 1 - \frac{1}{5n'}. \tag{55}$$

Now Let τ be a sufficiently large polynomial specified later, and consider any $(x, 1^s, 1^t) \in \mathsf{NO}$, i.e., $\mathsf{rK}^{\tau(t)}(x) > s + \log \tau(t)$. We will show that

$$\mathop{\mathbf{Pr}}_{B'}\big[B'(x, 1^s, 1^t) = 1\big] \leq 1 - \frac{2}{5n'}. \tag{56}$$

Note that by combining Equation (55) and Equation (56), B' yields an polynomial-time algorithm for $(\mathsf{YES}, \mathsf{NO})$ via standard success amplification techniques.

Suppose, for the sake of contradiction, Equation (56) is not true. Then by the definition of B', we have

$$\mathop{\mathbf{Pr}}_{w,z,B}\left[B(G_m(x;z)\circ w, 1^{s'}, 1^{t'}) = 1\right] > 1 - \frac{2}{5n'}. \tag{57}$$

On the other hand, by the property of B (Item 2), we get

$$\mathop{\mathbf{Pr}}_{u,w,B}\left[B(u\circ w, 1^{s'}, 1^{t'}) = 1\right] < 1 - \frac{1}{2n'}. \tag{58}$$

Comparing Equation (57) and Equation (58), it is clear that $B(-\circ \mathcal{U}_{t^c}, 1^{s'}, 1^{t'})$ distinguishes $G_m(x;\mathcal{U}_{O(\log^3 n)})$ from $\mathcal{U}_m$ with advantage $1/10n'$. By Lemma 42 and by letting τ be a sufficiently large polynomial, we get

$$\begin{aligned}\mathsf{rK}^{\tau(t)}(x) &\le m + O(\log^3 n) + O(\log t') \\ &\le s + c^3\cdot \log t + c\log^3 n + O(\log^3 n) + O(c\log t) \\ &\le s + \log\tau(t) + \log^3\tau(n).\end{aligned}$$

This means $(x, 1^s, 1^t)$ is *not* in NO, which gives the desired contradiction. ◀

▶ **Corollary 49.** *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then there exist a constant* $c > 0$*, a polynomial* τ*, and a probabilistic polynomial-time algorithm* $\mathsf{Approx}\text{-}\mathsf{rK}$ *that, on input* $(x, 1^t, 1^k)$ *where* $x \in \{0,1\}^n$*,* $t \ge cn$ *and* $k \in \mathbb{N}$*, with probability at least* $1 - 2^{-k}$ *outputs an integer* s *such that*

$$\mathsf{rK}^{\tau(t)}(x) - \log\tau(t) - \log^3\tau(n) \ \le\ s\ \le\ \mathsf{rK}^t(x).$$

Proof. Consider a randomized polynomial-time algorithm A that solves the promise problem from Lemma 48. By standard error reduction techniques, assume without loss of generality that on the inputs satisfying the promise its error is at most $2^{-k}/n^2$, where $n = |x|$. Note that the running time of A becomes $\mathsf{poly}(n,k)$. Our algorithm $\mathsf{Approx}\text{-}\mathsf{rK}$ runs A on $(x, 1^s, 1^t)$ for $s = 1, 2, \ldots, n + \log n$, and outputs the first s such that $A(x, 1^s, 1^t) = 1$.

The correctness of $\mathsf{Approx}\text{-}\mathsf{rK}$ follows by a union bound. Indeed, if $s < \mathsf{rK}^{\tau(t)}(x) - \log\tau(t) - \log^3\tau(n)$, i.e., $\mathsf{rK}^{\tau(t)}(x) > s + \log\tau(t) + \log^3\tau(n)$, using the promise we get that $\mathbf{Pr}_A[A(x, 1^s, 1^t) = 1] \le 2^{-k}/n^2$. On the other hand, if $s = \mathsf{rK}^t(x)$, which implies that $\mathsf{rK}^t(x) \le s$ and the promise is satisfied, we have $\mathbf{Pr}_A[A(x, 1^s, 1^t) = 1)] \ge 1 - 2^{-k}/n^2$. Since $\mathsf{rK}^t(x) \le n + \log n$, if $t \ge cn$, where $c \ge 1$ is a sufficiently large constant, then with high probability over the internal randomness of $\mathsf{Approx}\text{-}\mathsf{rK}$, it outputs a value s such that $\mathsf{rK}^{\tau(t)}(x) - \log\tau(t) - \log^3\tau(n) \le s \le \mathsf{rK}^t(x)$. ◀

We are now ready to prove Lemma 47.

Proof of Lemma 47. Let τ' be the polynomial from Corollary 49, and let $\mathsf{Approx}\text{-}\mathsf{rK}$ be the algorithm from Corollary 49. By running $\mathsf{Approx}\text{-}\mathsf{rK}(x, 1^{t_1}, 1^{k+1})$, with probability at least $1 - 2^{-k}/2$, we get some integer s_0 such that

$$\mathsf{rK}^{\tau'(t_1)}(x) - \log\tau'(t_1) - \log^3\tau'(n) \ \le\ s_1\ \le\ \mathsf{rK}^{t_1}(x).$$

Similarly, by running $\mathsf{Approx}\text{-}\mathsf{rK}\big(x, 1^{t_2}, 1^{k+1}\big)$, with probability at least $1 - 2^{-k}/2$, we get some integer s_2 such that

$$\mathsf{rK}^{\tau'(t_2)}(x) - \log\tau'(t_2) - \log^3\tau'(n) \ \le\ s_2\ \le\ \mathsf{rK}^{t_2}(x).$$

Then with probability at least $1 - 2^{-k}$, we have

$$s_1 - s_2 \le \mathsf{rK}^{t_1}(x) - \Big(\mathsf{rK}^{\tau'(t_2)}(x) - \log \tau'(t_2) - \log^3 \tau'(n)\Big) \tag{59}$$

and

$$s_1 - s_2 \ge \Big(\mathsf{rK}^{\tau'(t_1)}(x) - \log \tau'(t_1) - \log^3 \tau'(n)\Big) - \mathsf{rK}^{t_2}(x). \tag{60}$$

We can then output

$$s := s_1 - s_2 + \log \tau'(t_1) + \log^3 \tau'(n).$$

Note that using Equations (59) and (60), we have

$$\begin{aligned} s &\le \mathsf{rK}^{t_1}(x) - \mathsf{rK}^{\tau'(t_2)}(x) + \log \tau'(t_1) + \log \tau'(t_2) + 2 \cdot \log^3 \tau'(n) \\ &\le \mathsf{rK}^{t_1}(x) - \mathsf{rK}^{\tau(t_2)}(x) + \log \tau(t_1) + \log \tau(t_2) + \log^3 \tau(n), \end{aligned}$$

and $s \ge \mathsf{rK}^{\tau'(t_1)}(x) - \mathsf{rK}^{t_2}(x) \ge \mathsf{rK}^{\tau(t_1)}(x) - \mathsf{rK}^{t_2}(x)$, where in the above we let $\tau > \tau'$ be a large polynomial. ◀

B.2 Proof of Theorem 41

In this subsection, we prove the following, which implies Theorem 41 via Proposition 11.

▶ **Theorem 50.** *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then for every polynomial-time samplable distribution family* $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$*, where each* $\mathcal{D}_n$ *is over* $\{0,1\}^n$*, there exist a polynomial* ρ *and a probabilistic algorithm* A *such that the following hold for all* $\lambda \in (0,1)$*, all* $n, s, \ell, k \in \mathbb{N}$*, and all* $t \ge \rho(n) \cdot \log(1/(1-\lambda))$*.*

1. *For all* $x \in \{0,1\}^n$*,* $A(x, \lambda, 1^t, 1^\ell, 1^k)$ *runs in time* $2^{O(\log^3 n)} \cdot \mathsf{poly}(|\lambda|, t, \ell, k)$ *and outputs either a program or* $\bot$*.*
2. *For all* $x \in \{0,1\}^n$*,*
$$\Pr_A\big[A(x, \lambda, 1^t, 1^\ell, 1^k) \text{ outputs neither an } (1/\ell)\text{-}\mathsf{rK}^t_\lambda\text{-witness of } x \text{ nor } \bot\big] \le 2^{-k}.$$
3. *With probability at least* $1 - 1/k$ *over* $x \sim \mathcal{D}_n$*,*
$$\Pr_A\big[A(x, \lambda, 1^t, 1^\ell, 1^k) = \bot\big] \le 2^{-k}.$$

Proof. Throughout the proof, we will assume that $t \ge \rho(n)\cdot\log(1/(1-\lambda))$ for some polynomial ρ, which will be specified later. Here we assume without loss of generality that $\lambda \ge 2/3$. The proof can be easily adapted to the case where $\lambda \le 2/3$.

Let $t \in \mathbb{N}$ be such that $t \ge p_0(3n)$, where p_0 is the polynomial from Lemma 43. Consider any $x \in \{0,1\}^n$ and let y_t be a rK^t_λ-witness of x. That is, y_t is a program such that $U(y_t, r)$ outputs x within t steps with probability at least λ over $r \sim \{0,1\}^t$ and $|y_t| = \mathsf{rK}^t_\lambda(x)$. Also, let $q := \lceil 1/(1-\lambda) \rceil$. Note that $\log(q) \le O(|\lambda|)$.

By symmetry of information (Lemma 43), we have, for some polynomial p_{Sol},

$$\begin{aligned} &\mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(y_t \mid x) \\ &\le \mathsf{rK}^{2t}(x, y_t) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(3n) \\ &\le |y_t| - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(3n) + O(1) \\ &= \mathsf{rK}^t_\lambda(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(3n) + O(1) \qquad \text{(by the definition of } y_t) \\ &\le \mathsf{rK}^{t/O(\log q)}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(3n) + O(\log\log q), \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by Lemma 10)} \end{aligned}$$

where the second inequality follows from the fact that given y_t, one can also output x within t steps with probability at least $2/3$.

Let $t' := t/O(\log(1/(1-\lambda))$. Then we have

$$\mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(y_t \mid x) \leq \mathsf{rK}^{t'}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(3n) + O(\log|\lambda|). \tag{61}$$

Let $d > 0$ be some constant specified later, we say that $x \in \{0,1\}^n$ is (t,k)-*good* if

$$\mathsf{rK}^{t'}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) \leq d \cdot (\log t + \log^3 n) + \log k. \tag{62}$$

Consider any x, t, k such that x is (t,k)-good. Equation (61) implies that

$$\begin{aligned}\mathsf{rK}^{t^d}(y_t \mid x) &\leq \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(y_t \mid x)\\ &\leq \mathsf{rK}^{t'}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + \log^3 p_{\mathsf{Sol}}(3n) + O(\log\log q)\\ &\leq 2d \cdot (\log t + \log^3 n) + \log k + d \cdot \log|\lambda|, \end{aligned} \tag{63}$$

provided that d is a sufficiently large constant (which depends on p_{Sol}).

Given Equation (63) and using standard success amplification techniques (Lemma 10), we get that for some sufficiently large constant $c > d$, there is a randomized program Π_{y_t} of length at most

$$s := c \cdot \left(\log^3 n + \log t + \log k + \log|\lambda|\right) \tag{64}$$

that, given x, outputs y_t within $T := t^c \cdot k^c$ steps with probability at least $1 - 2^{-k}/2$. We aim to find such a y_t.

Let Valid be the algorithm from Claim 21, and let A' be the following algorithm that, given $(x, \lambda, 1^t, 1^\ell, 1^k)$ such that x is (t,k)-good, aims to output an $(1/\ell)$-rK^t_λ-witness of x.

Algorithm 4 Search for rK^t-Witnesses for Good x's.

```
procedure $A'(x, \lambda, 1^t, 1^\ell, 1^k)$
    $n := |x|$
    $M := 0^{2n}$
    $s := c \cdot \left(\log^3 n + \log t + \log k + \log|\lambda|\right)$, where $c$ is the constant from Equation (64).
    $T := t^c \cdot k^c$

    for $\Pi \in \{0,1\}^{\leq s}$ do
        $r :=$ a uniformly random string in $\{0,1\}^T$.
        $y :=$ the output of $U(\Pi, x, r)$ after running $T$ steps.
        if $|y| < |M|$ and $\mathsf{Valid}(x, y, \lambda, 1^t, 1^\ell, 1^{k+s+2})$ then
            $M := y$
    Output $M$
```

It is easy to verify that $A'(x, \lambda, 1^t, 1^k, 1^\ell)$ runs in time $2^{O(\log^3 n)} \cdot \mathsf{poly}(|\lambda|, t, k, \ell)$. Next, we argue that if x is (t,k)-good, then the above algorithm outputs an $(1/\ell)$-rK^t_λ-witness of x with probability $1 - 2^{-k}$.

Note that if x is (t,k)-good, then as described in previous paragraphs there is a randomized program Π_{y_t} of length at most $s := c \cdot \left(\log^3 n + \log t + \log k + \log|\lambda|\right)$ such that $U(\Pi_{y_t}, x, r)$ outputs y_t within $T := t^c \cdot k^c$ steps with probability at least $1 - 2^{-k}/2$ over $r \sim \{0,1\}^T$. For such an x, our algorithm A' will successfully output an $(1/\ell)$-rK^t_λ-witness of x if both of the following are true.

1. The algorithm Valid succeeds (meaning that the condition stated in Claim 21 is satisfied) in all of the $m := \sum_{i=1}^{s} 2^i \leq 2^{s+1}$ executions, which happens with probability at least $1 - 2^m \cdot 2^{-k-s-2} \geq 1 - 2^{-k}/2$.
2. For $\Pi = \Pi_{y_t}$, $U(\Pi, x, r)$ outputs y_t within T steps, which happens with probability at least $1 - 2^{-k}/2$ over $r \sim \{0,1\}^T$.

To see this, if the first item is true, then the randomized program M output by the algorithm is always a "valid" one that outputs x within t steps with probability at least $\lambda - 1/\ell$. If the second item is true, we are guaranteed that that $|M| \leq |y_t| = \mathsf{rK}^t_\lambda(x)$, since $\mathsf{Valid}(x, y_t, \lambda, 1^t, 1^\ell, 1^{k+s+1}) = 1$ (for a successful execution of Valid). The correctness of the algorithm then follows by a union bound.

We now describe our final algorithm A in the theorem. Let τ be the quasi-polynomial in Lemma 47, and let $\mathsf{Approx\text{-}depth}$ be the algorithm from Lemma 47. Our final algorithm A works as follows.

On input $(x, \lambda, 1^t, 1^\ell, 1^k)$, we first check if

$$\mathsf{Approx\text{-}depth}\left(x, 1^{\lfloor \tau^{-1}(t') \rfloor}, 1^{p_{\mathsf{Sol}}(2t)}, 1^k\right) \leq d \cdot (\log t + \log^3 n) + \log k,$$

where d is the constant in Equation (62). If yes, we output $A'(x, \lambda, 1^t, 1^\ell, 1^k)$. Otherwise, we output $\perp$.

We argue that the algorithm A above satisfies the three conditions stated in the theorem. The first condition is easy to verify.

For the second condition, we consider two cases. Suppose x is not (t,k)-good, meaning that

$$\mathsf{rK}^{t'}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) > d \cdot (\log t + \log^3 n) + \log k.$$

Note that by Lemma 47, in this case $\mathsf{Approx\text{-}depth}\left(x, 1^{\lfloor \tau^{-1}(t') \rfloor}, 1^{p_{\mathsf{Sol}}(2t)}, 1^k\right)$ outputs, with probability at least $1 - 2^{-k}$, some s that satisfies

$$\begin{aligned} s &\geq \mathsf{rK}^{\tau(\lfloor \tau^{-1}(t') \rfloor)}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) \\ &\geq \mathsf{rK}^{t'}(x) - \mathsf{rK}^{p_{\mathsf{Sol}}(2t)}(x) \\ &> d \cdot (\log t + \log^3 n) + \log k. \end{aligned}$$

Therefore, our algorithm will output $\perp$ with probability at least $1 - 2^{-k}$.

Now suppose x is (t,k)-good. Then $A'(x, \lambda, 1^t, 1^\ell, 1^k)$ will output an $(1/\ell)$-rK^t-witness of x with probability at least $1 - 2^{-\ell}$, which suffices to imply that the probability that our algorithm outputs neither an $(1/\ell)$-optimal rK^t-witness of x nor $\perp$ is at most 2^{-k} in this case, which also yields the second condition in this case.

Finally, for the third condition, we will show that in the above algorithm the criteria using $\mathsf{Approx\text{-}depth}$ will fail (hence output $\perp$) with probability at most $1/k$ over $x \sim \mathcal{D}_n$. To show this, we claim the following.

▷ Claim 51. For every $t, k \in \mathbb{N}$ such that $t \geq \rho(n)$, with probability at least $1 - 1/k$ over $x \sim \mathcal{D}_n$, we have

$$\begin{aligned} \zeta :&= \mathsf{rK}^{\lfloor \tau^{-1}(t') \rfloor}(x) - \mathsf{rK}^{\tau(p_{\mathsf{Sol}}(2t))}(x) + \log \tau(\lfloor \tau^{-1}(t') \rfloor) + \log \tau(p_{\mathsf{Sol}}(2t)) + \log^3 \tau(n) \\ &\leq d \cdot (\log t + \log^3 n) + \log k. \end{aligned}$$

Proof of Claim 51. Recall the coding theorem for rK (Lemma 45). By letting ρ be a sufficiently large polynomial so that for all $t \geq \rho(n)\cdot\log(1/(1-\lambda)))$, it is satisfied that $\lfloor \tau^{-1}(t')\rfloor \geq p_{\mathsf{code}}(n)$, where p_{code} is the polynomial from Lemma 45, we get that for every $x \in \mathsf{Support}(\mathcal{D}_n)$,

$$\mathsf{rK}^{\lfloor \tau^{-1}(t')\rfloor}(x) \leq \mathsf{rK}^{p_{\mathsf{code}}(n)}(x) \leq \log\frac{1}{\mathcal{D}_n(x)} + \log p_{\mathsf{code}}(n). \tag{65}$$

On the other hand, by Lemma 9, with probability at least $1 - 1/k$ over $x \sim \mathcal{D}_n$, we have

$$\mathsf{K}(x) \geq \log\frac{1}{\mathcal{D}_n(x)} - b\log n - \log k,$$

where $b > 0$ is a constant. In particular, by Fact 6, this implies

$$\mathsf{rK}^{\tau(p_{\mathsf{Sol}}(2t))}(x) \geq \log\frac{1}{\mathcal{D}_n(x)} - b\log n - \log k. \tag{66}$$

Combining Equations (65) and (66), we get that with probability at least $1 - 1/k$ over $x \sim \mathcal{D}_n$,

$$\begin{aligned}\zeta :&= \mathsf{rK}^{\lfloor \tau^{-1}(t')\rfloor}(x) - \mathsf{rK}^{\tau(p_{\mathsf{Sol}}(2t))}(x) + \log\tau(\lfloor\tau^{-1}(t')\rfloor) + \log\tau(p_{\mathsf{Sol}}(2t)) + \log^3\tau(n)\\ &\leq \left(\log\frac{1}{\mathcal{D}_n(x)} + \log p_{\mathsf{code}}(n)\right) - \left(\log\frac{1}{\mathcal{D}_n(x)} - b\log n - \log k\right)\\ &\qquad + \log t' + \log\tau(p_{\mathsf{Sol}}(2t)) + \log^3\tau(n)\\ &= \log p_{\mathsf{code}}(n) + b\log n + \log k + \log t' + \log\tau(p_{\mathsf{Sol}}(2t)) + \log^3\tau(n)\\ &\leq d\cdot(\log t + \log^3 n) + \log k,\end{aligned}$$

where the last inequality holds by letting d be a sufficiently large constant. ◁

To see that the third condition follows from Claim 51, note that by Lemma 47, we have $\mathsf{Approx\text{-}depth}\left(x, 1^{\lfloor\tau^{-1}(t')\rfloor}, 1^{p_{\mathsf{Sol}}(2t)}, 1^k\right)$ outputs, with probability at least $1 - 2^{-k}$, some s such that $s \leq \zeta$. Then by Claim 51, we obtain that for at least $1 - 1/k$ fraction of the x sampled from $\mathcal{D}_n$, our algorithm will output something other than $\perp$ with probability at least $1 - 2^{-k}$, as desired. ◀

C Search-to-Decision Reductions for the GapMINKT Problem

Mazor and Pass [30, Theorem 1.1] have recently described a sub-exponential time search-to-decision reduction for a gap version of time-bounded Kolmogorov complexity. In this section, we describe some related results obtained via techniques from meta-complexity.

Let MINKT denote the set of strings $(x, 1^s, 1^t)$ such that $\mathsf{K}^t(x) \leq s$. We consider a gap version of the corresponding computational problem, defined as follows. For a polynomial p, we let $\mathsf{Gap}_p\mathsf{MINKT}$ denote the following promise problem:

- YES instances consist of strings $(x, 1^s, 1^t)$ such that $\mathsf{K}^t(x) \leq s$;
- NO instances consist of strings $(x, 1^s, 1^t)$ such that $\mathsf{K}^{p(t,|x|)} > s + \log p(t, |x|)$.

We say that an algorithm A solves $\mathsf{Search\text{-}Gap}_p\mathsf{MINKT}$ if given any $\mathsf{Gap}_p\mathsf{MINKT}$ YES instance $(x, 1^s, 1^t)$, A outputs a program Π of length at most $s + \log p(t, |x|)$ such that $U^{p(t,|x|)}(\Pi) = x$. In other words, the algorithm certifies that $(x, 1^s, 1^t)$ is not a NO instance of $\mathsf{Gap}_p\mathsf{MINKT}$. We can think of A as providing an approximate or near-optimal solution to the search problem for K^t, since there is a bounded overhead in the running time and in the description length of the provided solution.

▶ **Theorem 52.** *Assume that* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$*. If there is a polynomial* p *such that* $\mathsf{Gap}_p\mathsf{MINKT}$ *admits a polynomial-time algorithm, then there is a polynomial* q *and a polynomial-time algorithm that solves* $\mathsf{Search\text{-}Gap}_q\mathsf{MINKT}$*.*

Proof. We rely on the *efficiency* and *bounded advice complexity* (under $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$) of the reconstruction procedure of the k-wise direct product generator $\mathsf{DP}_k\colon \{0,1\}^n \times \{0,1\}^{nk} \to \{0,1\}^{nk+k}$, defined as follows:

$$\mathsf{DP}_k(x; z) := (z^1, \dots, z^k, \langle z^1, x\rangle, \dots, \langle z^k, x\rangle) ,$$

where $\langle z, x\rangle$ denotes the inner product of $z \in \{0,1\}^n$ and $x \in \{0,1\}^n$ over $\mathsf{GF}(2)$. We will need the following result.

▶ **Lemma 53** (Reconstruction Lemma for K^t [11])**.** *Assume that* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$*. There is a polynomial* q_1 *such that, for every* $n \in \mathbb{N}$*,* $x \in \{0,1\}^n$*, parameter* $k \in \mathbb{N}$*, and for every deterministic circuit* C *of size* ℓ *such that*

$$\left| \Pr_z[C(\mathsf{DP}_k(x; z)) = 1] - \Pr_w[C(w) = 1] \right| \geq 1/n,$$

where $z \sim \{0,1\}^{nk}$ *and* $w \in \{0,1\}^{nk+k}$*, it holds that*

$$\mathsf{K}^{q_1(n\cdot\ell)}(x \mid C) \leq k + \log q_1(n \cdot \ell).$$

Moreover, there is a deterministic algorithm B *that, given* $x \in \{0,1\}^n$*,* $k \in \mathbb{N}$*, and* C*, runs in polynomial time and outputs a string* y *of length at most* $k + \log q_1(n, \ell)$ *such that* $U^{q_1(n,\ell)}(y, C) = x$*.*

Sketch of the Proof of Lemma 53. The assumption that $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$ yields a pseudorandom generator G of seed length $O(\log m)$ that allows us to derandomize non-uniform algorithms of complexity at most m [20]. In the construction described below, we can use G to derandomize any internal procedure of the program that outputs x given C. We note that by fixing a good seed of G in such a derandomization, we will incur an overhead in the description length of the program for x of at most $\log \mathsf{poly}(n, \ell)$ bits, while the overhead in the running time of the program is $\mathsf{poly}(n, \ell)$. These overheads do not create an issue because we can take the polynomial q_1 to be of large enough degree. (Moreover, it would be enough to design a randomized algorithm B, since this algorithm can be derandomized in a standard way by trying all seeds of the generator and outputting the first valid program with the desired parameters.)

Using the circuit C as a distinguisher and Yao's equivalence between breaking a candidate generator and next-bit (un)predictability, it follows that there is an index $i \in [k]$ such that $\langle z^i, x\rangle$ can be predicted with probability at least $1/2 + \Omega(1/(n \cdot k))$, given $z^1, \dots, z^k, \langle z^1, x\rangle, \dots \langle z^{i-1}, x\rangle$ as input. Since $\langle z^i, x\rangle$ is the z^i-th bit of the Hadamard code of x, we can use the next-bit predictor and the list-decoding algorithm of the Hadamard code to recover with noticeable probability a list of strings of polynomial size that contains x. More precisely, this yields a randomized algorithm M (with access to C) that runs in time polynomial in n and ℓ such that, given a random choice of $z^1, \dots, z^k$ and the corresponding bits $\langle z^1, x\rangle, \dots \langle z^{i-1}, x\rangle$, outputs with probability at least $1/\mathsf{poly}(n, \ell)$ a list S of size $\mathsf{poly}(n, \ell)$ that contains x.

As explained above, using the generator G, a good choice of the random strings $z^1, \dots, z^k$ as well as of the internal randomness of M can be obtained from some seed $\sigma \in \{0,1\}^s$, where $s = O(\log \mathsf{poly}(n, \ell))$. Additionally, the $i - 1 \leq k$ "advice bits" $\langle z^1, x\rangle, \dots \langle z^{i-1}, x\rangle$

can be efficiently computed from x and $z^1, \dots z^{i-1}$. These advice bits are stored in the corresponding program y witnessing that $\mathsf{K}^{q_1(n \cdot \ell)}(x \mid C) \leq k + \log q_1(n \cdot \ell)$. Finally, the position of x in the list S can be encoded with $O(\log \mathsf{poly}(n, \ell))$ bits of advice and is also stored in y.

The search algorithm B from the "moreover" part of the result tries all seeds σ of the generator until a good seed is found, a condition that can be tested by running the corresponding program y_σ. The overall running time of B is $\mathsf{poly}(n, \ell)$, as desired. ◀

We now describe the search-do-decision reduction. Assume that $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. Let F be a polynomial-time algorithm that decides $\mathsf{Gap}_p\mathsf{MINKT}$. Note that we can assume without loss of generality that F is deterministic, since $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$ yields strong pseudorandom generators [20]. For a large enough polynomial q specified later in the proof, we describe a deterministic polynomial-time algorithm A that solves $\mathsf{Search}\text{-}\mathsf{Gap}_q\mathsf{MINKT}$, i.e., given any $\mathsf{Gap}_q\mathsf{MINKT}$ YES instance $(x, 1^s, 1^t)$, A outputs a $q(t, |x|)$-time-bounded description of x of length at most $s + \log q(t, |x|)$.

Let $\alpha \geq 1$ be a large enough constant. We assume that $s \leq n - 10\alpha \cdot \log n$, since otherwise a trivial description of x is a correct output for $\mathsf{Gap}_q\mathsf{MINKT}$ (taking q to be of large enough degree). We can also assume that $t \geq n$, as the polynomial q can depend on $n = |x|$. The search algorithm A sets $k = s + \alpha \cdot \log n$ in the execution of the algorithm B from Lemma 53, and let $C(v)$ be the (deterministic) circuit of size $\ell = \mathsf{poly}(t)$ obtained from F on inputs of the form $(v, 1^{s'}, 1^{t'})$, where $|v| = nk + k$, $s' = nk + k - (\alpha/4) \cdot \log n$, and $t' = q_2(t)$, for a polynomial q_2 of large enough degree.

For a given $\mathsf{Gap}_q\mathsf{MINKT}$ YES instance $(x, 1^s, 1^t)$, it is not hard to see that for every string $z \in \{0,1\}^{nk}$,

$$\mathsf{K}^{q_2(n)}(\mathsf{DP}_k(x; z)) \leq |z| + \mathsf{K}^t(x) + O(\log n) \leq nk + s + O(\log n) \leq nk + k - (\alpha/2) \cdot \log n,$$

where we used that α is large enough. On the other hand, for a random string $w \sim \{0,1\}^{nk+k}$, a simple counting argument gives that

$$\mathsf{K}(w) \geq nk + k - \log n$$

with probability at least $1/n$. Recall that $C(v)$ accepts every instance v such that $\mathsf{K}^{t'}(v) \leq s'$ and rejects every instance v such that $\mathsf{K}^{p(t', |v|)}(v) > s' + \log p(t', |v|)$. Consequently, due to our choices of s' and t' and using a large enough α, it is not hard to see that $C(v)$ satisfies the condition in the statement of Lemma 53.

Therefore, the algorithm B on inputs x, $k = s + \alpha \cdot \log n$, and C (as defined above), runs in time $\mathsf{poly}(x, k, |C|) = \mathsf{poly}(n, s, t)$ and outputs a string y of length at most

$$k + \log q_1(n, \ell) = s + \alpha \cdot \log n + \log q_1(n, \mathsf{poly}(t))$$

such that $U^{q_1(n,\ell)}(y, C) = x$. Since C can be efficiently computed from the code of F (which has description length $O(1)$) and parameters s and t (which can be described with $\log n + \log t$ bits), if we take q to be a large enough polynomial, A can produce from y a string Π of length at most $s + \log q(t, |x|)$ such that $U^{q(t,|x|))}(\Pi) = x$. This completes the proof of Theorem 52. ◀

▶ Remark 54 (Comparison with [30, Theorem 1.1]). On the one hand, the (black-box) search-to-decision reduction in [30, Theorem 1.1] is unconditional, while Theorem 52 relies on a standard derandomization assumption and is non-black-box. On the other hand, Theorem 52 provides a *polynomial*-time search to decision reduction for $\mathsf{GapMINKT}$, as opposed to the *sub-exponential* running time of [30, Theorem 1.1].

▶ Remark 55 (Exact versus Gap Search-to-Decision Reductions for K^t). We note that the assumption in Theorem 52 on the existence of a polynomial p such that $\mathsf{Gap}_p\mathsf{MINKT}$ admits a polynomial-time algorithm is implied by "$\mathsf{MINKT} \in \mathsf{AvgBPP}$" and $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$ (see [10, 9] or [12]). In other words, the assumption on the easiness of $\mathsf{Gap}_p\mathsf{MINKT}$ can be replaced by "$\mathsf{MINKT} \in \mathsf{AvgBPP}$" in Theorem 52. In particular, under the assumptions of Theorem 1 we can efficiently solve the exact search problem for K^t on average and the gap search problem for K^t in the worst case.

We observe that it is possible to derive a weaker unconditional consequence from Theorem 52 via a win-win argument. We will need the following simple result.

▶ **Proposition 56.** *Suppose that $\mathsf{E} \subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. Then, for every $\varepsilon > 0$, there exists infinitely many values of n and a circuit C_n of size at most $2^{\varepsilon \cdot n}$ such that, given a string $x \in \{0,1\}^n$ and $1 \leq t \leq 2^n$* (represented as an n-bit string)*, $C_n(x,t)$ outputs a minimum length program Π such that $U^t(\Pi) = x$.*

Proof. Under the assumption, for every $L \in \mathsf{E}$ and for every $\varepsilon > 0$, there is an infinite set $S \subseteq \mathbb{N}$ such that, for every $n \in S$, there is a circuit C_n of size at most $2^{\varepsilon \cdot n}$ that computes L_n, i.e., L restricted to inputs of length exactly n.

We consider a paddable language L (with a padding input parameter k) that contains all tuples $\langle x, w, i, s, t, 1^k \rangle$ such that:

(i) $|x| = n$ for some n, $|w| = n$, $|i| = \log n$, $|s| = \log n$, $|t| = n$, and k is arbitrary. We view i as an integer such that $1 \leq i \leq n$, and t as an integer such that $1 \leq t \leq 2^n$.

(ii) Let $w_{\leq i}$ be the i-th bit prefix w. There is a program Π that extends w_i and is of length at most s such that $U^t(\Pi) = x$.

We assume that the tuples in L employ an encoding such that the bit-length of $\langle x, w, i, s, t, 1^k \rangle$ as a string is precisely $4n + k$, whenever n is sufficiently large. This is easy to get, since $|x| + |w| + |i| + |s| + |t| = 3n + 2\log n$ for positive instances. The particular choice of encoding is not important as long as the tuple can be efficiently encoded and decoded.

Observe that $L \in \mathsf{E}$. Under the assumption of Proposition 56, for every $\delta > 0$ there are infinitely many input lengths m such that L on input length m admits a circuit D_m of size at most $2^{\delta \cdot m}$. Using the padding parameter k, it is not hard to see that we can use D_m to decide tuples $\langle x, w, i, s, t, 1^k \rangle$ with $|x| = m/5$. Finally, let $n = |x|$. Given D_m, by a standard binary search over prefixes of w, we can optimally solve the search problem for K^t on x in size $2^{\delta \cdot m} \cdot \mathsf{poly}(m) \leq 2^{6 \cdot \delta \cdot n}$. Since $\delta > 0$ can be taken arbitrarily small, the result follows. ◀

By combining Theorem 52 and Proposition 56, we get the following unconditional search-to-decision reduction. (Since we consider Boolean circuits in the next statement, which are devices that operate over fixed input lengths, we assume an upper bound on the input parameters as a function of n.)

▶ **Theorem 57.** *If there is a polynomial p such that $\mathsf{Gap}_p\mathsf{MINKT}$ admits a polynomial-time algorithm, then there is a polynomial q such that, for every $\varepsilon > 0$, there are infinitely many input lengths n such that $\mathsf{Search\text{-}Gap}_q\mathsf{MINKT}$ can be solved by a circuit of size at most $2^{\varepsilon \cdot n}$ on inputs $(x, 1^s, 1^t)$, where we assume that $x \in \{0,1\}^n$, $1 \leq s \leq n + \log n$, and $1 \leq t \leq 2^{o(n)}$.*

Proof. If $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$, the result immediately follows from Theorem 52. Otherwise, we get that $\mathsf{E} \subseteq \mathsf{i.o.SIZE}[2^{o(n)}]$. Let C_n be one of the circuits from Proposition 56. Then we can use C_n to solve the search problem of any $\mathsf{Gap}_q\mathsf{MINKT}$ YES instance $(x, 1^s, 1^t)$. This completes the proof. ◀

Note that, in contrast to the search-to-decision reduction from [30, Theorem 1.1], which provides a *uniform* algorithm for $\mathsf{Search\text{-}Gap}_q\mathsf{MINKT}$ with the sub-exponential-time $2^{\varepsilon \cdot s} \cdot \mathsf{poly}(n, t, s)$ (for every $\varepsilon > 0$), Theorem 57 only provides a non-uniform infinitely often sub-exponential-time $2^{\varepsilon \cdot n}$ algorithm (for every $\varepsilon > 0$), and so has similar sub-exponential in s efficiency only for $s \in \Omega(n)$.[9]

D Errorless Average-Case Search-to-Decision Reduction for K^t over the Uniform Distribution

In this section, we describe polynomial-time errorless average-case search-to-decision reduction over the uniform distribution for K^t. We get the following polynomial-time average-case search-to-decision reduction for K^t in the errorless setting over the uniform distribution. This complements a similar result in [22], which holds in the error-prone setting.

▶ **Theorem 58.** *If* $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ *holds, then there exist a polynomial* ρ *and a probabilistic polynomial-time algorithm* A *such that the following holds for all* $n, s, k \in \mathbb{N}$*, and all* $t \geq \rho(n)$*.*

1. *For all* $x \in \{0,1\}^n$*,*

$$\Pr_A\left[A(x, 1^t, 1^k) \text{ outputs either an } \mathsf{K}^t\text{-witness of } x \text{ or } \perp\right] \geq 1 - \frac{1}{2^k}.$$

2. *With probability at least* $1 - 1/k$ *over* $x \sim \mathcal{D}_n$*,*

$$\Pr_A\left[A(x, 1^t, 1^k) \text{ outputs } \perp\right] \leq \frac{1}{2^k}.$$

Proof. The proof follows a similar approach to that of Theorem 50.

Let $n \in \mathbb{N}$ and let $t \geq \rho(n)$ for some polynomial ρ specified later.

Consider any $x \in \{0,1\}^n$ and let y_t be a K^t-witness of x.

By the assumption that $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$ holds, it follows from Lemma 17 that there exist polynomials p_{Sol} such that

$$\begin{aligned}
\mathsf{pK}^{p_{\mathsf{Sol}}(2t)}(y_t \mid x) &\leq \mathsf{pK}^{2t}(x, y_t) - \mathsf{pK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) \\
&\leq \mathsf{K}^{2t}(x, y_t) - \mathsf{pK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) \\
&\leq |y_t| - \mathsf{pK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) \\
&= \mathsf{K}^t(x) - \mathsf{pK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t).
\end{aligned}$$

Using standard amplification techniques for probabilistic time-bounded Kolmogorov complexity (see, e.g., [6, Lemma 21]), we get

$$\mathsf{pK}_{1-2^{-k}}^{p_{\mathsf{Sol}}(2t) \cdot \mathsf{poly}(k)}(y_t \mid x) \leq \mathsf{K}^t(x) - \mathsf{pK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + O(\log k). \tag{67}$$

Let $d > 0$ be a constant determined later. We say that x is (t,k)*-good* if

$$\mathsf{pK}^{p_{\mathsf{Sol}}(2t)}(x) > n - d \cdot \log t - \log k.$$

[9] In Theorem 57 there is a dependence on n in the exponent of the circuit size, as opposed to a dependence on s in the running time as in [30, Theorem 1.1]. This is inherent in the non-uniform model when the parameter s is part of the input, since the circuit is fixed and must work for all values of s including $s = \Theta(n)$. In other words, in a uniform algorithm the running time can depend on a given input instance, but in a circuit its size is fixed for all inputs of a given input length.

Note that if x is (t,k)-*good*, then the quantity in Equation (67) becomes

$$\begin{aligned}\mathsf{pK}^{(t\cdot k)^d}_{1-2^{-k}}(y_t \mid x)| &\le \mathsf{K}^t(x) - \mathsf{pK}^{p_{\mathsf{Sol}}(2t)}(x) + \log p_{\mathsf{Sol}}(2t) + O(\log k)\\ &< (n+O(1)) - (n - d\cdot\log t - \log k) + \log p_{\mathsf{Sol}}(2t) + O(\log k)\\ &\le 2d\cdot(\log t + \log k),\end{aligned}$$

if we let d be a sufficiently large constant. This implies that for at least $1-2^{-k}$ fraction of the $w \in \{0,1\}^T$, where $T := (tk)^d$, there is a program Π_{y_t} of size at most $s := 2d\cdot(\log t + \log k)$ such that $U(\Pi, w)$ outputs y_t within T steps. Therefore, the following procedure A' will be able to find a K^t-witness of x with probability at least $1-2^{-k}$.

> On input $(x, 1^t, 1^k)$, we pick $w \sim \{0,1\}^T$, enumerate all $\Pi \in \{0,1\}^{\le s}$, run $U(\Pi, w)$ for T steps and obtain a list of candidate programs $\{y\}$ (which is guaranteed to contain y_t). We then return the shortest y that outputs x within t steps.

Let M be the randomized algorithm that approximates pK^t as in Lemma 16. By standard amplification techniques, we can amplify its success probability to be $1-2^{-k}$, by blowing up the running time by at most $\mathsf{poly}(k)$. Consider the following algorithm $\mathsf{Certify}$:

> On input $(x, 1^t, 1^k)$, let $t' := p_{\mathsf{Sol}}(2t)$ and let $\theta := n - d\cdot\log t - \log k$. We accept if and only if $M(x, 1^{t'}) \ge \theta$.

Our final algorithm A works as follows:

> On input $(x, 1^t, 1^k)$, we runs $\mathsf{Certify}(x, 1^t, 1^k)$, if it rejects, we output $\perp$; otherwise, we run $A'(x, 1^t, 1^k)$ and output whatever it outputs.

We show the first condition of Theorem 58. Note that on the one hand, if x is not (t,k)-good, then by the correctness of M, $\mathsf{Certify}(x, 1^t, 1^k)$ will reject with probability at least $1-2^{-k}$ over its internal randomness.

On the other hand, if x is indeed (t,k)-good, then our algorithm A' will return a K^t-witness of x with probability with probability at least $1-2^{-k}$ over its internal randomness.

To see the second condition, note that by a simple counting argument, with probability at least $1-1/k$ over $x \sim \{0,1\}^n$, it holds that

$$\begin{aligned}\mathsf{pK}^{\tau(p_{\mathsf{Sol}}(2t))}(x) &\ge \mathsf{K}(x) - O(\log \tau(p_{\mathsf{Sol}}(2t)))\\ &\ge n - O(\log \tau(p_{\mathsf{Sol}}(2t))) - \log k\\ &> n - d\cdot\log t - \log k,\end{aligned}$$

where the last inequality holds if we choose d to be a sufficiently large constant. Again, by the correctness of M, this implies that $\mathsf{Certify}(x, 1^t, 1^k)$ will accept at least $(1-1/k)$-fraction of $x \in \{0,1\}^n$ (with probability at least $1-2^{-k}$ over its internal randomness). ◀