Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 89 (2025)

Kolmogorov's Approach to P vs. NP: Chain Rules for Time-Bounded Kolmogorov Complexity

Valentine Kabanets* Antonina Kolokolova†

March 13, 2026

Abstract

Time-bounded conditional Kolmogorov complexity of a string x given y, $\mathsf{K}^t(x \mid y)$, is the length of a shortest program that, given y, prints x within t steps. The Chain Rule for conditional K^t with error e is the following hypothesis: there is a constant $c \in \mathbb{N}$ such that, for any strings $y, x_1, \ldots, x_\ell \in \{0,1\}^*$, for any $\ell \in \mathbb{N}$, and all sufficiently large time bounds t,

$$\mathsf{K}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{K}^{t^c}(x_i \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - e(N,t),$$

where $N = \sum_{i=1}^{\ell} |x_i|$. When $y = \epsilon$ (the empty string), we get the Chain Rule for K^t.

In the late 1960s, Kolmogorov suggested that disproving the Chain Rule for K^t may be a good approach to proving that $\mathsf{P} \neq \mathsf{NP}$. We make a step towards showing that the two may be equivalent. Namely, we pinpoint the worst-case complexity assumptions *equivalent* to Chain Rules for (conditional) K^t, and the probabilistic variant pK^t, where $\mathsf{pK}^t(x \mid y) \leq s$ iff $\mathsf{K}^t(x \mid y, r) \leq s$ for at least $2/3$ of random strings $r \in \{0,1\}^t$.

- Chain Rule for conditional K^t with error $e(N,t) \leq o(N)$ is *equivalent* to the conjunction of the following two statements:
  1. $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$, and
  2. $\mathsf{GapMcK^tP} \in \mathsf{promise\text{-}P}$, where $\mathsf{GapMcK^tP}$ is a promise problem to distinguish between inputs $(x, y, 1^s)$ with $\mathsf{K}^t(x \mid y) \leq s$ and those with $\mathsf{K}^{\mathsf{poly}(t)}(x \mid y) > s + o(|x|)$.
- Chain Rule for conditional pK^t with error $e(N,t) \leq o(N)$ is *equivalent* to $\mathsf{GapMcpK^tP} \in \mathsf{promise\text{-}BPP}$, for the analog of $\mathsf{GapMcK^tP}$ for conditional pK^t.
- We get analogous equivalences for the case of unconditional K^t and pK^t (i.e., for $y = \epsilon$).

These are the first exact complexity characterizations for natural versions of Chain Rules for time-bounded Kolmogorov complexity. Assuming $\mathsf{GapMcK^tP}$ is NP-hard (which is true under cryptographic assumptions [HIR23]), the equivalence above would simplify to "the Chain Rule for conditional K^t with error $e(N,t) \leq o(N)$ holds iff $\mathsf{NP} = \mathsf{P}$", which would completely validate Kolmogorov's intuition.

Among some other results, we present a natural $\mathsf{promise\text{-}BPP}$-complete problem based on the problem of approximating $\mathsf{pK}^t(x \mid y)$ for short inputs x with $|x| \leq \log t$, and give some algorithmic consequences if $\mathsf{GapMcpK^tP}$ were easy.

*School of Computing Science, Simon Fraser University, Burnaby, BC, Canada; kabanets@sfu.ca
†Department of Computer Science, Memorial University of Newfoundland, NL, Canada; kol@mun.ca

ISSN 1433-8092

Contents

1 Introduction

Kolmogorov complexity defines an algorithmic measure of information in a given string x, denoted $\mathsf{K}(x)$, as the length of a shortest program that prints x. The conditional version $\mathsf{K}(x \mid y)$ is the length of a shortest program that, given y, prints x.

One of the fundamental properties of the Kolmogorov complexity measure K, discovered by Kolmogorov and Levin (see [ZL70]), is the Chain Rule, saying that for any binary strings x, y, and z,

$$\mathsf{K}(x, y \mid z) \approx \mathsf{K}(x \mid z) + \mathsf{K}(y \mid z, x), \tag{1}$$

where $\approx$ hides an additive error term, at most logarithmic in the total length of all strings. That is, the most efficient way to describe a pair (x, y) given z is to describe one of them given z, and then describe the other one assuming the knowledge of z and the first one.

When $z = \epsilon$ (the empty string), this Chain Rule is also called *Symmetry of Information (SoI)* for K, since it implies

$$\mathsf{K}(x) - \mathsf{K}(x \mid y) \approx \mathsf{K}(y) - \mathsf{K}(y \mid x),$$

reminding of the classical symmetry of information equality $H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$, where $H(X)$ is Shannon's entropy of a random variable X.

SoI for K is universally regarded as one of the most beautiful and useful results in classical Kolmogorov complexity. (For completeness, we prove SoI for K in Section A of the appendix.)

Symmetry of Information for Time-Bounded Kolmogorov Complexity. Of particular interest to computational complexity is the variant of *time-bounded* Kolmogorov complexity measure K^t, where $\mathsf{K}^t(x)$ refers to the length of a shortest program that prints x within t time steps. An obvious question is whether the time-bounded measure K^t satisfies SoI. Since the upper bound $\mathsf{K}^{2t}(x, y) \lessapprox \mathsf{K}^t(x) + \mathsf{K}^t(y \mid x)$ is straightforward, SoI for K^t is defined as the following hypothesis:

$$\mathsf{K}^t(x, y) \geq \mathsf{K}^{p(t)}(x) + \mathsf{K}^{p(t)}(y \mid x) - O(\log t), \tag{SoI for K^t}$$

for some polynomial p, and any sufficiently large time bound t.

According to Levin [Lev03], Kolmogorov was interested in the SoI for K^t question, since he believed it might be useful in the quest to resolve open problems in computational complexity. In particular, Kolmogorov suggested in the late 1960s that disproving SoI for K^t might be a good approach to proving that $\mathsf{P} \neq \mathsf{NP}$. [1]

However, it was only in 1995 that Longpre and Watanabe [LW95] proved that $\mathsf{P} = \mathsf{NP}$ implies SoI for K^t. Thus, Kolmogorov's suggestion to disprove SoI for K^t in order to prove $\mathsf{P} \neq \mathsf{NP}$ was sound. Since then, it remains an open question if disproving SoI for K^t is also *necessary* for proving $\mathsf{P} \neq \mathsf{NP}$. More generally, the open question is to come up with a *natural complexity statement* (like $\mathsf{P} = \mathsf{NP}$) that would be *equivalent* to SoI for K^t.

[1] Kolmogorov's intuition was, perhaps, that his proof of the chain rule for the time-unbounded K used an algorithm doing a *brute-force search* over exponentially many possibilities, and it was not clear to him how such exhaustive search could be avoided.

Multi-String Chain Rules. Applying the Chain Rule in (1) inductively, we easily get the following version of the chain rule for multiple strings: For any $\ell \in \mathbb{N}$, and any $x_1, \ldots, x_\ell, y \in \{0,1\}^*$,

$$\mathsf{K}(x_1, \ldots, x_\ell \mid y) \approx \sum_{i=1}^{\ell} \mathsf{K}(x_i \mid y, x_1, \ldots, x_{i-1}), \tag{2}$$

where $\approx$ hides an additive error term at most $\ell \cdot O(\log(\sum_{i=1}^{\ell} |x_i| + |y|))$. (A way to interpret the universal quantification over ℓ and $x_1, \ldots, x_\ell$ above is: for all strings $x \in \{0,1\}^*$ and for any partitioning of x into ℓ substrings $x_1, \ldots, x_\ell$, for any $1 \leq \ell \leq |x|$, so that $x = x_1 \ldots x_\ell$.)

An analogous version for the time-bounded case is the following hypothesis:

$$\mathsf{K}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{K}^{p(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - e(N), \qquad \text{(Chain Rule for } \mathsf{cK}^t\text{)}$$

where p is some polynomial that is the *same* for any number ℓ of strings $x_1, \ldots, x_\ell$, y is any string, t is a sufficiently large polynomial time bound in the total string length $N + |y|$ for $N = \sum_{i=1}^{\ell} |x_i|$, and $e(N)$ is an error term.

When $y = \epsilon$ (the empty string), we will refer to the chain rule above as the Chain Rule for K^t, with error $e(N)$.

Note that the interesting setting of parameters for this Chain Rule is when ℓ is *super-constant* (for example, $\ell = o(N/\log t)$). For super-constant ℓ, unlike in the time-unbounded case of K, SoI for K^t (the chain rule for two strings) does not seem to imply the multi-string chain rule by induction on ℓ. [2] Thus, in the time-bounded setting, the multi-string Chain Rule appears to be *more powerful* than SoI.

This, potentially more powerful, version of the chain rule for conditional K^t is our main subject of study. For this chain rule, we do get an *exact complexity characterization*, as we explain next.

1.1 Results

Chain rule for (conditional) K^t. First we define the problem $\mathsf{Gap}_{\tau,o(n)}\mathsf{McK^tP}$. This is a promise-problem for conditional $\mathsf{K}^t(x \mid y)$, where one needs to distinguish between inputs $(x, y, 1^s)$ with $\mathsf{K}^t(x \mid y) \leq s$ and those with $\mathsf{K}^{\tau(t)}(x \mid y) > s + o(|x|)$, for some polynomial τ. For the case of K^t, the promise-problem $\mathsf{Gap}_{\tau,o(n)}\mathsf{MK^tP}$ is defined analogously (with $y = \epsilon$, the empty string). The Chain Rule for K^t is related to the complexity of $\mathsf{Gap}_{\tau,o(n)}\mathsf{MK^tP}$; similarly, the Chain Rule for conditional K^t is related to the complexity of $\mathsf{Gap}_{\tau,o(n)}\mathsf{McK^tP}$. We state both of these equivalences in the single statement below.

Theorem 1.1 (Case of (conditional) K^t, informal)**.** *Chain Rule for (conditional)* K^t *with multiple strings, with error* $e(N) \leq o(N)$*, is* equivalent *to the conjunction*

$$\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}] \;\;\&\;\; \mathsf{Gap}_{\tau,o(n)}\mathsf{M(c)K^tP} \in \mathsf{promise\text{-}P}.$$

[2] The reason is a polynomial blowup in the time bounds for K^t on the right-hand side of SoI after each inductive step, resulting in super-polynomial in t time bounds on the right-hand side of the resulting chain rule when ℓ is super-constant.

The problem $\mathsf{Gap}_{\tau,o(n)}\mathsf{McK^tP}$ is likely to be NP-hard. For example, [HIR23] show that this problem is NP-hard under randomized polynomial-time reductions, assuming the existence of subexponentially secure witness encryption[3] for NP. (There is also some evidence that $\mathsf{Gap}_{\tau,o(n)}\mathsf{MK}^t\mathsf{P}$ is NP-hard [Ila23].)

If we suppose that $\mathsf{Gap}_{\tau,o(n)}\mathsf{McK^tP}$ is indeed NP-hard, our equivalence in Theorem 1.1 would simplify to the following: "Chain Rule for conditional K^t with multiple strings, with error $e(N) \leq o(N)$, is *equivalent* to $\mathsf{P} = \mathsf{NP}$". This would show that disproving the Chain Rule for K^t is also *necessary* for proving $\mathsf{P} \neq \mathsf{NP}$, validating Kolmogorov's intuition to the fullest!

Chain rule for (conditional) pK^t. In our equivalence for the chain rule for K^t above, we have a derandomization assumption of exponential circuit complexity for a problem in exponential time. This circuit lower bound assumption may be dropped if we consider a probabilistic variant of time-bounded Kolmogorov complexity pK^t instead of K^t. This probabilistic measure can be defined as follows: $\mathsf{pK}^t(x \mid y) \leq s$ if $\mathsf{K}^t(x \mid y, r) \leq s$ for at least $2/3$ of uniformly random strings $r \in \{0,1\}^t$.

We need the problem $\mathsf{Gap}_{\tau,o(n)}\mathsf{McpK^tP}$, which is a promise problem for conditional $\mathsf{pK}^t(x \mid y)$, defined analogously to $\mathsf{Gap}_{\tau,o(n)}\mathsf{McK^tP}$ above. For the case of unconditional pK^t, the problem $\mathsf{Gap}_{\tau,o(n)}\mathsf{MpK^tP}$ is defined similarly.

Theorem 1.2 (Case of (conditional) pK^t, informal)**.** *Chain Rule for (conditional)* pK^t *with multiple strings, with error* $e(N) \leq o(N)$*, is* equivalent *to* $\mathsf{Gap}_{\tau,o(n)}\mathsf{M(c)pK^tP} \in \mathsf{promise\text{-}BPP}$.

Similarly to the case of K^t, if we assume that, say, $\mathsf{Gap}_{\tau,o(n)}\mathsf{McpK^tP}$ is NP-hard (under randomized reductions), then the equivalence in Theorem 1.2 would simplify to: "Chain Rule for conditional pK^t with multiple strings, with error $e(N) \leq o(N)$, is *equivalent* to $\mathsf{NP} \subseteq \mathsf{BPP}$".

Theorems 1.1 and 1.2 are formally stated and proved in Section 7.

The complexity of computing pK^t. While it is likely that the problem $\mathsf{Gap}_{\tau,o(n)}\mathsf{McpK^tP}$ considered above may be NP-hard, we do not have any proof yet. A natural approach to exploring the computational power of a problem is to assume that it is easy and see if any interesting algorithmic consequences would follow. In that spirit, we show the following.

Theorem 1.3 (Consequences of easiness of $\mathsf{GapMcpK^tP}$, informal)**.** *If, for some polynomial* τ*,* $\mathsf{Gap}_{\tau,o(n)}\mathsf{McpK^tP} \in \mathsf{promise\text{-}P}$*, then*

1. $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$*, and*
2. *there is a deterministic polynomial-time algorithm for a version of Range Avoidance, where one is given a circuit* $C\colon \{0,1\}^n \to \{0,1\}^{2n}$*, and needs to find a* $y \in \{0,1\}^{2n}$ *such that* $C(x) \neq y$ *for all* $x \in \{0,1\}^n$.

Recall that deciding if $\mathsf{pK}^t(x \mid y) \leq s$ is equivalent to deciding if

$$\Pr_{r \in \{0,1\}^t}\left[\mathsf{K}^t(x \mid r, y) \leq s\right] \geq 2/3. \tag{3}$$

We consider a variant of this problem where (i) the input x is very short, $|x| \leq \log t$, and (ii) we just need to approximate the probability in (3) up to an additive error $1/8$. It is not hard to see

[3] We note that secure witness encryption for NP, introduced in [Gar+13], may exist in either the world where there are one-way functions, or the world where $\mathsf{P} = \mathsf{NP}$.

that under these two restrictions, the resulting problem to approximate $\Pr_{r\in\{0,1\}^t}\left[\mathsf{K}^t(x \mid r, y) \le s\right]$ is in promise-BPP. (For any such short x, we can compute $\mathsf{K}^t(x \mid r, y)$ in deterministic $\mathsf{poly}(t)$ time by brute force. By randomly sampling enough r's, we can approximate the required probability in randomized time $\mathsf{poly}(t)$.) It turns out that this problem is promise-BPP-complete (under polynomial-time Turing reductions).

Theorem 1.4 (promise-BPP-complete problem)**.** *The problem to estimate, for given $x, y \in \{0,1\}^*$ and $s, t \in \mathbb{N}$ such that $|x| \le \log t$, the probability*

$$\Pr_{r\in\{0,1\}^t}\left[\mathsf{K}^t(x \mid r, y) \le s\right],$$

to within an additive error at most $1/8$, is promise-BPP*-complete.*

The proof of Theorem 1.4 also yields a simple alternative proof of the recent result by [LPT24] showing that if one could derandomize Yao's "distinguisher-to-predictor" transformation, then promise-BPP = promise-P.

Theorems 1.3 and 1.4 are formally stated and proved in Section 8.

1.2 Techniques

To prove Theorems 1.1 and 1.2, we use the concept of a *natural property* for (conditional) K^t and pK^t, respectively. A natural property for conditional K^t on n-bit strings with *usefulness* $s(n,t)$ is a predicate $\mathcal{P}$ such that: for all $x \in \{0,1\}^n$ and $y \in \{0,1\}^*$, if $\mathsf{K}^t(x \mid y) \le s(n,t)$, then $\mathcal{P}(x, y, 1^t) = 1$; and for all $y \in \{0,1\}^*$, $\mathcal{P}(x, y, 1^t) = 0$ for at least $1/2$ of uniformly random $x \in \{0,1\}^n$. Thus, in a certain sense, a natural property distinguishes $s(n,t)$-easy n-bit strings (given y) from uniformly random ones.

A natural property for conditional pK^t is defined analogously. Also, natural properties for unconditional K^t and pK^t are defined similarly to the conditional case above, with $y = \epsilon$.

We first give "worst-case to average-case reductions" showing that, under the assumption that $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$, computing a natural property for (conditional) K^t is equivalent to computing a corresponding gap version of $\mathsf{M(c)K^tP}$. Similarly, for pK^t, we show that computing a natural property for (conditional) pK^t is equivalent to computing a corresponding gap version of $\mathsf{M(c)pK^tP}$; no derandomization is needed in this case.

We then show that a chain rule for (conditional) K^t (or pK^t) is equivalent to the existence of an efficiently computable natural property for (conditional) K^t (or pK^t); and, in the case of the chain rule for (conditional) K^t, also the derandomization assumption that $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$.

This implies the equivalence between chain rules and the worst-case complexity of the corresponding gap problems for K^t (or pK^t).

We give more details next.

Chain Rule from Natural Property. Here we sketch a proof that a natural property for (conditional) pK^t implies a chain rule for (conditional) pK^t; the case of the chain rule for (conditional) K^t is proved similarly (using an extra derandomization assumption that $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$).

The proof of the Chain Rule for pK^t from a natural property for pK^t is a generalization of the proof argument from [Hir22; GK22; Gol+22] [4]. It uses tools from pseudorandomness, in particular,

[4] A similar result has been recently proved by [HN25].

a Nisan-Wigderson-style [NW94] hardness-based generator based on weak designs of [RRV02], with efficient reconstruction; this generator has been analyzed and used in the context of time-bounded Kolmogorov complexity by [BLM05; Hir22]. (The previous works on chain rules [Hir22; GK22; Gol+22; HN25] use a simpler generator based on the direct product of Hadamard codes, DP^x_k, which maps a seed $z_1, \ldots, z_k \in \{0,1\}^n$ to the inner products modulo 2 of x with each z_i, for $1 \le i \le k$. This Direct Product generator, however, has a seed of length nk, which can be as large as n^2. This was not an issue in the previous works, but is a problem for us since we work in the regime of additive errors $o(n)$. That is why we use the more seed-efficient generator G^x_k, that has the seed length $O(\log^3 n)$.)

In more detail, for $x \in \{0,1\}^n$ and a parameter $0 \le k \le 2n$, there is a polynomial-time computable generator G^x_k takes a seed $z \in \{0,1\}^{o(n)}$ and outputs a k-bit string satisfying the following: any efficient algorithm that can distinguish the outputs of $G^x_k(z)$ from uniform yields an algorithm for efficiently reconstructing x, given about k bits of advice. In other words, G^x_k "encodes" any given string $x \in \{0,1\}^n$ into a distribution over k-bit strings (over all seeds z) with the following property: if this distribution can be distinguished from the uniform distribution by some efficient algorithm (distinguisher) D (with possibly some advice $\alpha \in \{0,1\}^*$), then $\mathsf{pK}^t(x \mid \alpha) \lessapprox k$. It follows that if we set k to be just slightly less than $\mathsf{pK}^t(x \mid \alpha)$, then DP^x_k will be a pseudorandom generator.

We first give an oversimplified proof argument, and then explain the extra details that need to be added. To prove the conditional pK^t Chain Rule for strings $x_1, \ldots, x_\ell, y$, we consider the concatenation of ℓ generators

$$G(z) := z \circ G^{x_1}_{k_1}(z) \circ \cdots \circ G^{x_\ell}_{k_\ell}(z)$$

on the same seed z, where each $k_i \approx \mathsf{pK}^{t'}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1})$, for some $t' = \mathsf{poly}(t)$.[5] By a hybrid argument[6], this choice of k_i's ensures that the concatenation of ℓ generators is a pseudorandom generator against efficient distinguishers with advice y. This means that a natural property for $\mathsf{pK}^t(- \mid y)$ is fooled by the generator. It follows that, for some choice of the seed z, we have by the usefulness of the natural property that

$$\begin{aligned}\mathsf{pK}^t(z \circ G^{x_1}_{k_1}(z) \circ \cdots \circ G^{x_\ell}_{k_\ell}(z) \mid y) &\gtrapprox \sum_{i=1}^{\ell} k_i \\ &\approx \sum_{i=1}^{\ell} \mathsf{pK}^{t'}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}).\end{aligned}$$

On the other hand, for every choice of z we have by the definition of G^x_k that

$$\mathsf{pK}^t(z \circ G^{x_1}_{k_1}(z) \circ \cdots \circ G^{x_\ell}_{k_\ell}(z) \mid y) \lessapprox \mathsf{pK}^t(x_1, \ldots, x_\ell \mid y).$$

Combining the two inequalities above, we get the following "weak" version of the required chain rule:

$$\mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) \gtrapprox \sum_{i=1}^{\ell} \mathsf{pK}^{t'}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}).$$

[5] Our actual generator is a padded version of $G(z)$, where for each $1 \le i \le \ell$, we pad up $G^{x_i}_{k_i}(z)$ with a uniformly random string of length $2|x_i| - k_i$; this allows us to calculate the required hybrid distributions with small additional advice only.

[6] Intuitively, to be able to sample efficiently from a hybrid distribution with the first i positions being $G^{x_1}_{k_1}(z) \circ \cdots \circ G^{x_i}_{k_i}(z)$, we need to know both $x_1, \ldots, x_i$ and $k_1, \ldots, k_i$.

For the case of $\ell \in O(1)$, the above argument yields the required chain rule, because we can describe constantly many k_i's with additional advice of $O(\log N)$ bits, where $N = \sum_{i=1}^{\ell} |x_i|$. However, for $\ell \in \omega(1)$, the extra advice would be at least $\sum_{i=1}^{\ell} O(i \cdot \log N) \geq \Omega(\ell^2 \log N)$ bits, which is too much for us to afford (e.g., this advice is bigger than N when $\ell \approx N/\log N$). So we need to compute the parameters $k_1, \ldots, k_\ell$ efficiently.

There are two ways of how one might try doing this. If we have an efficient (deterministic) algorithm for approximately computing conditional pK^t, we can compute $k_1, \ldots, k_\ell$ in this order so that $k_i \approx \mathsf{pK}^{t'}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1})$. If we assume a natural property for *conditional* pK^t, we do have an efficient *randomized* approximation algorithm for conditional pK^t. There is still a complication that such an algorithm for computing k_i's is only randomized, and so k_i's are not defined uniquely. It is however possible to solve this problem, using some averaging arguments. (For the simpler case of proving the chain rule for conditional K^t, we do follow this approach, using a deterministic approximation algorithm for conditional K^t; see Theorem 4.1 for details.)

Another approach (which we actually take for the case of pK^t) is based on the observation from [HN25] that one can define $k_1, \ldots, k_\ell$ sequentially as maximally possible values so that the natural property algorithm cannot distinguish between any two successive hybrid distributions; this can be tested efficiently with random sampling. The latter implies that the natural property is fooled by the concatenated generator, which implies that $\mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) \gtrapprox \sum_{i=1}^{\ell} k_i$. Using the reconstruction property of the generator G and the fact that k_i's were defined as *maximum* possible values to have indistinguishability between two successive hybrid distributions, we can show that each $k_i \gtrapprox \mathsf{pK}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1})$. This again implies the "weak version" of the chain rule for pK^t.

To eliminate k_i's from the advice, we follow [HN25] and take advantage of the fact that "good" k_i's are computed with very high probability. This can be shown to imply that the "weak version" of the chain rule for pK^t also (approximately) holds in *expectation* over the randomness r used to determine k_i's, i.e.,

$$\mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) \gtrapprox \mathbf{Exp}_r \left[\sum_{i=1}^{\ell} \mathsf{pK}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, r) \right].$$

The linearity of expectation allows us to replace the right-hand side of the inequality above with the sum of expectations of $\mathsf{pK}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, r)$ over uniformly random r's. Finally, by Markov's inequality and the definition of pK^t, one can lowerbound each $\mathbf{Exp}_r[\mathsf{pK}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, r)]$ by $\mathsf{pK}^{p'(t)}(x_i \mid y, x_1, \ldots, x_{i-1})$, for a slightly larger polynomial time bound $p'(t)$. This produces the required chain rule for pK^t. (See Theorem 4.3 for details.)

The case of the chain rule for (conditional) K^t can be argued along the same lines, but is simpler, as we make use of the derandomization assumption (exponential circuit lower bounds for E) to compute the required parameters $k_1, \ldots, k_\ell$ efficiently *deterministically*. (See Theorem 4.2 for details.)

Natural Property from Chain Rule. To derive a natural property for conditional pK^t from the Chain Rule for conditional pK^t, we proceed as follows. Given strings x, y, partition $x \in \{0,1\}^n$ into $\ell = n/(c \log t)$ strings $x_1, \ldots, x_\ell$ of length $c \log t$ each. If $\mathsf{pK}^t(x \mid y) \leq s(n)$, for some $s(n) < n$,

then by the Chain Rule, so is the sum of conditional Kolmogorov complexities

$$\sum_{i=1}^{\ell} \mathsf{pK}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}),$$

where for simplicity of the presentation we ignore some additive error terms. Hence, by averaging, there is at least one $1 \leq i \leq \ell$ such that

$$\mathsf{pK}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) \leq s(n)/\ell. \tag{4}$$

Since each x_i is of length $O(\log t)$, we can approximate the conditional time-bounded Kolmogorov complexity of x_i in Eq. (4) in randomized time $\mathsf{poly}(n, t)$.

The resulting randomized algorithm will accept (with high probability) all "easy" strings x. By a counting argument, we can also show that this algorithm is likely to reject at least $1/2$ of uniformly random strings x. Thus we get a natural property for pK^t, which is computable by a randomized polynomial-time algorithm. See Section 5 for details.

Natural Property for Conditional K^t vs. $\mathsf{GapMcK^tP}$. Using the idea of the "worst-case to average-case" reduction of [Hir18], combined with the use of DP generators as in [Hir20c; Hir20b], we can show an equivalence between the existence of an efficiently computable natural property for (conditional) K^t (with usefulness $n - o(n)$) and the existence of an efficient algorithm for solving $\mathsf{Gap}_{\tau, o(n)}\mathsf{M(c)K^tP}$, for some polynomial τ (under the circuit lower bound assumption that $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$). Such an equivalence allows us to relate the chain rules to the existence of efficient algorithms for solving $\mathsf{GapM(c)K^tP}$, yielding Theorem 1.1. (A similar argument yields also Theorem 1.2, without using the circuit lower bound assumption.) We sketch the main idea next; see Section 3.1 for details.

The main idea used in the proof that a natural property implies a chain rule (sketched above) is that a given string x can be compressed to about k bits if the generator G^x_k can be broken by some $\mathsf{poly}(t)$-time distinguisher.

Since the conditional K^t complexity of the output of the generator $z \circ G^x_k(z)$ (given an auxiliary string y) is always at most approximately $\mathsf{K}^t(x \mid y) + |z|$, a natural property for conditional K^t (with usefulness at least $\mathsf{K}^t(x \mid y) + |z|$) will always accept the outputs $G^x_k(z)$, for all z and y, no matter what k is.

In the other direction, if this natural property accepts $z \circ G^x_k(z)$, given y, with high probability (say, at least 0.6) over random z, then it distinguishes the outputs of $G^x_k(z)$ from the uniform distribution (because a natural property accepts at most $1/2$ of random strings). Thus it breaks the generator G^x_k. It follows (by the reconstruction property of G) that $\mathsf{K}^{\mathsf{poly}(t)}(x \mid y) \leq k$.

These arguments can be used to imply that, for $k \approx s$, estimating the acceptance probability of the natural property on $(z \circ G^x_k(z), y)$, over random z, allows one to distinguish between pairs of strings (x, y) with $\mathsf{K}^t(x \mid y) \leq s$ and those with $\mathsf{K}^{\mathsf{poly}(t)}(x \mid y) \gg s$: the former pairs of strings will be accepted with probability 1, while the latter ones with probability at most 0.6.

Incompressible Strings from Chain Rules. Chain Rule for K^t implies an efficient algorithm to construct, for any $n, t \in \mathbb{N}$, for $n \leq t \leq 2^{o(n)}$, an *incompressible string* $z \in \{0,1\}^n$ with $\mathsf{K}^t(z) \geq n/2$, in time $\mathsf{poly}(n, t)$.

To find the required incompressible $z \in \{0,1\}^n$, for given $n, t \in \mathbb{N}$, proceed as follows.

Set $t' = p(t)$, for the polynomial p from the Chain Rule for K^t. For a constant $c > 0$ to be determined, set $m = c \log t$ and $\ell = n/m$. Find, one after the other, by brute force, strings $w_1, w_2, \ldots, w_\ell \in \{0,1\}^m$ so that, for each $1 \le i \le \ell$, $\mathsf{K}^{t'}(w_i \mid w_1, \ldots, w_{i-1}) \ge m$. Output $z = w_1 w_2 \ldots w_\ell \in \{0,1\}^n$.

For the analysis, observe that such strings $w_1, \ldots, w_\ell$ exist by a counting argument, and each can be found by brute force in time $2^{O(m)} \cdot \mathsf{poly}(t') \le \mathsf{poly}(t)$. By the Chain Rule for K^t,

$$\mathsf{K}^t(z_1, \ldots, z_\ell) \ge \ell \cdot m - \ell \cdot O(\log t) = (n/c \log t) \cdot (c \log t - O(\log t)),$$

which is at least $n/2$, for a sufficiently large constant $c > 0$. Hence $z = w_1 \ldots w_\ell$ is the required incompressible string, and it is constructed in time $\ell \cdot \mathsf{poly}(t) \le \mathsf{poly}(n, t)$.

The constructed incompressible string z (for large enough polynomial t) can be shown to require large circuit size as well. Thus we get that a chain rule for conditional K^t also yields the circuit lower bound $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$, completing the proof of the forward direction of Theorem 1.1. See Section 6 for details.

Similar reasoning also proves Item (1) of Theorem 1.3. The idea is that the assumption of Theorem 1.3 yields a chain rule for conditional pK^t, by Theorem 1.2. The latter can be used to construct an incompressible string, relative to the conditional pK^t complexity measure, piece by piece, as we constructed the string $z = w_1 \ldots w_\ell$ above. Here, when constructing each w_i of high conditional pK^t complexity, we use an assumed deterministic polynomial-time algorithm $\mathcal{B}$ for $\mathsf{GapMcpK^tP}$ to choose the lexicographically first log-length string rejected by $\mathcal{B}$.

The proof of Item (2) of Theorem 1.3 also follows since there is a close connection (in fact, equivalence) between solving the Range Avoidance problem and efficiently constructing a string x of near-maximum conditional Kolmogorov complexity. The idea is that every string $x \in \{0,1\}^m$ in the range of a given circuit $C\colon \{0,1\}^n \to \{0,1\}^{2n}$ has $\mathsf{K}^t(x \mid C) \le n < 2n$, for some $t = \mathsf{poly}(|C|)$. Thus any string $z \in \{0,1\}^{2n}$ with $\mathsf{K}^t(z \mid C) \approx 2n$ will be a solution to the Range Avoidance problem for the circuit C. See Section 8.4 for details.

Promise-BPP completeness. To prove Theorem 1.4, consider the generator

$$G^{x_1,\ldots,x_n}(z) := \langle x_1, z\rangle \circ \cdots \circ \langle x_n, z\rangle,$$

where $\langle a, b\rangle$ denotes the inner product modulo 2 of strings a and b, and each $x_i \in \{0,1\}^m$ for $m = c \log n$ for some large constant $c > 1$, and also $z \in \{0,1\}^m$. (We use a concatenation of Direct-Product generators, DP^x_k, for $k = 1$, where the DP^x_k is another generator that has an efficient reconstruction procedure, similar to that for the generator G^x_k we used earlier.)

We will explain how to choose the strings $x_1, \ldots, x_n$ to make this generator "good". We want to use this $G^{x_1,\ldots,x_n}(z)$ to approximate the acceptance probability of any given circuit $C\colon \{0,1\}^n \to \{0,1\}$, to within an additive error $1/8$. This approximation problem is known to be promise-BPP-complete.

Suppose that $G^{x_1,\ldots,x_n}(z)$ fails to approximate the acceptance probability of some $C\colon \{0,1\}^n \to \{0,1\}$ to within additive error $1/8$. Then this circuit C is a distinguisher between $G^{x_1,\ldots,x_n}(z)$ and the uniform distribution over $\{0,1\}^n$. Using an argument similar to that in the "Chain Rule from Natural Property" paragraph above, we can show that for some $1 \le i \le n$,

$$\mathsf{pK}^{\mathsf{poly}(|C|)}(x_i \mid C, x_1, \ldots, x_{i-1}) < c' \log n,$$

for some constant $c' > 1$, independent of the constant c.

It follows that if we are able to select $x_1, \ldots, x_n$ so that, for each $1 \leq i \leq n$,

$$\mathsf{pK}^{\mathsf{poly}(|C|)}(x_i \mid C, x_1, \ldots, x_{i-1}) > c' \log n,$$

the resulting generator $G^{x_1,\ldots,x_n}(z)$ will "fool" this circuit C. (Such generators fooling a specific given circuit C are called *targeted* generators by [Gol11].)

Since $\mathsf{pK}^t(x \mid w) \leq s$ iff $\mathsf{K}^t(x \mid w, r) \leq s$ for at least $2/3$ of random r's, we can show that access to an oracle that approximates the probability

$$\Pr_{r \in \{0,1\}^t} \left[\mathsf{K}^t(x \mid w, r) \leq s\right]$$

allows us to construct the required strings $x_1, \ldots, x_n$, one by one, by trying, for each $1 \leq i \leq n$, all possible candidate strings for x_i in $\{0,1\}^m$. See Section 8.1 and Section 8.2 for more details. For the proof that derandomizing Yao's "distinguisher-to-predictor" construction would imply $\mathsf{promise\text{-}BPP} = \mathsf{promise\text{-}P}$, see Section 8.3.

1.3 Related Work

SoI for Time-Bounded Kolmogorov Complexity. Some of the first formal connections between SoI for K^t and computational complexity and cryptography were discovered in the 1990s. It was shown that SoI for K^t implies that there are no one-way functions [LM93; LW95], and that SoI for K^t is implied by the assumption $\mathsf{P} = \mathsf{NP}$ [LW95].

The gap between the necessary and sufficient conditions for SoI for K^t has been narrowed recently. Hirahara [Hir22] and, independently, Goldberg and Kabanets [GK22] improved the result of [LW95] to show that SoI for K^t is implied by $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, an average-case (errorless) version of the assumption that $\mathsf{NP} = \mathsf{P}$. The argument was extended by [Gol+22] to the case of SoI for pK^t, under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$.

Thus, SoI for K^t is sandwiched between two average-case assumptions: that any one-way function candidate can be efficiently inverted on average, and that NP is easy on average. Hirahara [Hir22] makes further progress toward closing the gap. He shows that, assuming E requires exponential-size circuits, SoI for K^t is sandwiched between the assumptions that there exists an *errorless* average-case heuristic scheme and that there exists an *error-prone* average-case heuristic scheme, both for computing K^t. Still, exact complexity-theoretic characterization of SoI for K^t is missing.

Lee and Romashchenko [LR05] proved that SoI for K^t implies that $\mathsf{EXP} \neq \mathsf{BPP}$. Perifel [Per07] showed that a certain version of SoI for conditional K^t implies that $\mathsf{EXP} \not\subset \mathsf{P/poly}$. The latter paper seems to be the first to study the Chain Rule hypothesis for conditional K^t, showing how to derive a weak version of the Chain Rule for K^t from a variant SoI hypothesis for K^t.

Unconditionally, Ronneburger [Ron04] proved that SoI does *not* hold for Levin's version of the time-bounded Kolmogorov complexity measure Kt.

Recently, SoI for K^t has been an important tool for, among others, worst-case to average-case reductions for problems in the polynomial-time hierarchy, computational learning, meta-complexity, and cryptography [Hir20c; Hir21; HN21; Hir22; GK22; CHV22; Gol+22; Hir+23; Ila23].

For the probabilistic version of time-bounded Kolmogorov complexity pK^t, [Hir+23] shows that a certain *average-case* version of SoI for pK^t (over polynomial-time samplable distributions) is

equivalent to the non-existence of one-way functions. Note that this result is also an equivalence between a variant of SoI and a complexity (cryptography) assumption. The difference from the equivalences proved in our paper, in particular, our Theorem 1.2, is that we consider the *worst-case* version of a chain rule ("multi-string version of SoI") for conditional pK^t, and show it is equivalent to the *worst-case* complexity assumption about approximating conditional pK^t.

A very recent work by [HN25] shows how to prove a multi-string chain rule for pK^t, from an assumption that a certain variant of approximately computing K^t is in promise-BPP. They do it also by generalizing the previous proofs of SoI for pK^t [Hir22; GK22], using the properties of the Direct Product generator DP^x_k. They observe that one needs to compute the parameters $k_1, \ldots, k_\ell$ for the corresponding instantiations of $\mathsf{DP}^{x_1}_{k_1}, \ldots, \mathsf{DP}^{x_\ell}_{k_\ell}$, and propose a nice way of doing it via the "parameter-tuning" approach, which we also adopt in our proofs of the chain rules for K^t and pK^t from corresponding natural properties.

Complexity of computing Time-Bounded Kolmogorov Complexity measures. Hirahara [Hir18] shows worst-case to average-case reduction for GapMINKT. For the conditional Kolmogorov complexity $\mathsf{K}^t(x \mid y)$, the corresponding minimization problem $\mathsf{McK}^t\mathsf{P}$ is known to be NP-hard (under randomized reductions) in the *sublinear* time-bound regime when $t \ll |y|$ [Hir22; LP22b]; a sublinear-time version of computing conditional $\mathsf{pK}^t(x \mid y)$ is also NP-hard [LS24]. The problem $\mathsf{McK}^t\mathsf{P}$ is shown to be NP-hard also for $t \geq |y|$, but only under the additional cryptographic assumption of the existence of secure witness encryption [HIR23]. Hirahara [Hir20c] shows that conditional $\mathsf{K}^{t,\mathsf{SAT}}$ (where the decoding Turing machine also has oracle access to SAT) is NP-hard to compute. For a random oracle $\mathcal{O}$, the oracle version $\mathsf{MK}^{t,\mathcal{O}}\mathsf{P}$ is also known to be NP-hard [Ila23]; see [Ila23] and the references therein for more information about the ongoing quest to prove NP-hardness of various meta-complexity problems such as MCSP (Minimum Circuit Size Problem) and $\mathsf{MK}^t\mathsf{P}$ (Minimum K^t complexity Problem).

While the worst-case complexity of $\mathsf{MK}^t\mathsf{P}$ is yet unknown, its average-case complexity (in the error-prone setting of average-case complexity) has been characterized by Liu and Pass [LP20] who showed that $\mathsf{MK}^t\mathsf{P}$ is hard on average if and only if one-way functions exist; this equivalence was later extended also to the case of $\mathsf{McK}^t\mathsf{P}$ [LP22b]. More equivalences between one-way functions and meta-complexity are given in [IRS22; Hir23; LS24].

Natural properties. The concept of a *natural property* for circuit complexity was introduced by Razborov and Rudich [RR97] in the context of trying to understand the limitations of current proof techniques for showing strong circuit lower bounds. Roughly, a natural property for circuit size $S(n)$ is a predicate on inputs of size $N = 2^n$ that accepts all truth tables of n-input Boolean functions of circuit complexity at most $S(n)$ (all "easy" strings), and rejects at least $1/2$ of random N-bit strings; the first condition (of accepting "easy" strings) is called *usefulness*. It was shown in [RR97] that an efficiently computable natural property (of appropriate usefulness) yields an efficient algorithm for inverting well on average any given candidate one-way function, and so the existence of one-way functions would imply the non-existence of efficiently computable natural properties.

The concept became highly influential, and inspired a lot of research on circuit complexity, derandomization, computational learning, proof complexity, and meta-complexity, to name just a few. For instance, in the context of computational learning, the power of natural properties was explored in [Car+16; Car+17; OS17; Bin+22; Kar24]; in [GK23], stronger learning results were obtained from the natural properties for time-bounded Kolmogorov complexity such as K^t and

KT. Natural properties were shown to imply various circuit lower bounds in, e.g., [KC00; IKW02; IKV23]. In [Hir+24], an assumed natural property for K^t and the additional assumption that $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$, give efficient average-case algorithms for finding programs of size $\mathsf{K}^t(x)$ that generate x within time t (i.e., K^t *witnesses*), for strings x coming from any efficiently samplable distribution.

Range Avoidance Problem. The Range Avoidance Problem is related to the dual weak pigeonhole principle studied in the context of bounded arithmetic and propositional proof complexity [Kra01; Kra04; Jer04; Jeř07]. Motivated to identify natural search problems in the polynomial-time hierarchy, [Kle+21] studied the Range Avoidance Problem, for the case of circuits $C\colon \{0,1\}^n \to \{0,1\}^m$ with $m \geq n+1$, under the name 1-Empty. Korten [Kor21] showed that a polynomial-time algorithm for the Range Avoidance Problem would imply polynomial-time constructions for many important combinatorial objects (e.g., Ramsey graphs, extractors, rigid matrices, etc.). The Range Avoidance Problem for restricted circuit classes (and for different stretch regimes) was studied by [RSW22; GLW22; Gaj+23]. It is shown in [ILW23] that, under certain cryptographic assumptions, a polynomial-time algorithm for the Range Avoidance Problem (even for a polynomial stretch regime) would imply that $\mathsf{NP} = \mathsf{coNP}$; this result provides some evidence that the Range Avoidance problem may be intractable.

2 Preliminaries

2.1 Kolmogorov Complexity

Let U be a Turing machine. For $t \in \mathbb{N}$ and $x, y \in \{0,1\}^*$, we define *t-time-bounded Kolmogorov complexity of x* given y (with respect to U) as

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

We assume that the string y is given on a separate input tape. As usual, we fix U to be a time-optimal machine [LV19], 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 use $\mathsf{K}(x \mid y)$ to denote the (time-unbounded) Kolmogorov complexity of x given y. When $y = \epsilon$ (i.e., y is the empty string), we drop the conditioning on y, getting the definition of the (time-bounded) Kolmogorov complexity of x.

Next we recall the definitions of the randomized time-bounded Kolmogorov complexity pK^t; see [LO22] for more information on randomized time-bounded Kolmogorov complexity variants and their applications.

For $\lambda \in [0,1]$ and $t \in \mathbb{N}$, we define *t-time-bounded probabilistic Kolmogorov complexity of x* given y as

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

Equivalently,

$$\mathsf{pK}^t_\lambda(x \mid y) \leq s \iff \Pr_{r \in \{0,1\}^t}\big[\mathsf{K}^t(x \mid y, r) \leq s\big] \geq \lambda. \tag{5}$$

We assume that the random string r is given on a separate input tape. For simplicity, we omit λ when $\lambda = 2/3$.

We have the following "success amplification" lemma for pK^t.

Lemma 2.1 (Success Amplification for pK^t [Gol+22])**.** *For any strings $x, y \in \{0,1\}^*$, time bound $t \in \mathbb{N}$, and $0 < \alpha < \beta < 1$, we have*

$$\mathsf{pK}^{O(qt/\alpha)}_{\beta}(x \mid y) \leq \mathsf{pK}^t_{\alpha}(x \mid y) + O(\log(q/\alpha)),$$

where $q = -\ln(1-\beta)$.

Lemma 2.2 (Probabilistic Incompressibility [Gol+22])**.** *For any string $y \in \{0,1\}^*$, time bound $t \in \mathbb{N}$ (including $t = \infty$), $0 < \lambda \leq 1$, and $k \in \mathbb{N}$, we have*

$$\mathbf{Pr}_{x \in \{0,1\}^n}\left[\mathsf{pK}^t_{\lambda}(x \mid y) \leq n - k\right] \leq \frac{(2/\lambda)}{2^k}.$$

Remark 2.3. *By definition, the problem of deciding if $\mathsf{K}^t(x \mid y) \leq s$, for inputs $x, y \in \{0,1\}^*, 1^t, 1^s$, is in NP. The case of pK^t is more complicated, if one insists on deciding if $\mathsf{pK}^t(x \mid y) \leq s$ (which seems to require the power of exact counting). However, the following more relaxed promise-version is easily seen to be in $\mathsf{promise\text{-}AM}$:*

- $\Pi_{\text{YES}} = \{(x, y, 1^t, 1^s) \mid \mathsf{pK}^t_{2/3}(x \mid y) \leq s\}$, *and*
- $\Pi_{\text{NO}} = \{(x, y, 1^t, 1^s) \mid \mathsf{pK}^t_{1/3}(x \mid y) > s\}$.

By Lemma 2.1, we also get that the following gap-problem is in $\mathsf{promise\text{-}AM}$:

- $\Pi_{\text{YES}} = \{(x, y, 1^t, 1^s) \mid \mathsf{pK}^t(x \mid y) \leq s\}$, *and*
- $\Pi_{\text{NO}} = \{(x, y, 1^t, 1^s) \mid \mathsf{pK}^{ct}(x \mid y) > s + c\}$,

where $c \geq 1$ is a fixed constant.

2.2 Generators

Let $\mathcal{U}_n$ denote the uniform distribution over n-bit strings. For a generator $G\colon \{0,1\}^\ell \to \{0,1\}^n$, a (randomized) algorithm $D\colon \{0,1\}^n \to \{0,1\}$, and $0 < \varepsilon \leq 1$, we say that D *ε-distinguishes* $G(\mathcal{U}_\ell)$ from $\mathcal{U}_n$ if

$$\left|\mathbf{Pr}_{z \in \{0,1\}^\ell, D}[D(G(z)) = 1] - \mathbf{Pr}_{y \in \{0,1\}^n, D}[D(y) = 1]\right| \geq \varepsilon.$$

Otherwise, we say that the generator G *ε-fools* D. We call a generator G *an ε-pseudorandom generator (PRG)* for a class $\mathcal{C}$ of Boolean functions, if G ε-fools every $D \in \mathcal{C}$.

Theorem 2.4 ([NW94; IW97])**.** *Assume $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$. Then, for any size parameter $s(n)$ and error ε, there is an explicit construction of a generator $\mathsf{IW}\colon \{0,1\}^\ell \to \{0,1\}^n$ that is an ε-PRG for the class of n-input Boolean circuits of size at most $s(n)$, with the seed length $\ell \leq O(\log(s(n)/\varepsilon))$, that can be evaluated on any given seed in time $\mathsf{poly}(s(n)/\varepsilon)$.*

We will also need a generator G based on Trevisan's extractor [Tre01]. This is a variant of the Nisan-Wigderson generator [NW94] (using weak designs of [RRV02]) combined with a certain list-decodable error-correcting code. It was first used in [BLM05] for K^t reconstruction, and later by [Hir22] (see also [Hir20a]) for randomized rK^t reconstruction. We will use the formulation of the properties of the generator G as in [BLM05; Hir22].

Lemma 2.5 ([BLM05; Hir22])**.** *There exist deterministic polynomial-time algorithms G and A, and a randomized polynomial-time oracle-algorithm $R^{(\cdot)}$ such that, for all sufficiently large $n, k \in \mathbb{N}$, with $k \leq 2n$, and $d = O(\log^3 n)$, we have*

- *(generator) $G\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^k$,*
- *(advice) $A\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^k$,*
- *(reconstruction) $R^{(\cdot)}\colon \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$,*

and the following reconstruction property holds. For every $x \in \{0,1\}^n$, and any (possibly) randomized algorithm $D\colon \{0,1\}^{k+d} \to \{0,1\}$ such that D ε-distinguishes $z \circ G(x,z)$, for $z \sim \mathcal{U}_d$, from $\mathcal{U}_{k+d}$, we have that

$$\mathbf{Pr}_{z \in \{0,1\}^d, R, D}\left[R^D(A(x,z),z) = x\right] \geq \frac{1}{q(k/\varepsilon)},$$

for some fixed polynomial q.

Definition 2.6 (Generator G^x_k)**.** *For a given $x \in \{0,1\}^n$, and any $k \leq 2n$, we define $G^x_k\colon \{0,1\}^d \to \{0,1\}^k$ as $G^x_k(z) = G(x,z)$, for G from Lemma 2.5, where $d = O(\log^3 n)$. We define the* strong *version of G, denoted $\mathsf{s}\text{-}G^x_k\colon \{0,1\}^d \to \{0,1\}^{d+k}$, to output also the seed $z \in \{0,1\}^d$ used, i.e., $\mathsf{s}\text{-}G^x_k(z) = z \circ G^x_k(z)$.*

Lemma 2.7 (K^t Reconstruction using G^x_k)**.** *Assume $\mathsf{E} \not\subseteq \mathsf{io}\text{-}\mathsf{SIZE}[2^{o(n)}]$. For $\varepsilon > 0$, $x \in \{0,1\}^n$, $k \in \mathbb{N}$ satisfying $k \leq 2n$, and $d = O(\log^3 n)$, let D be a randomized algorithm that takes an advice string β and runs in time t_D such that D ε-distinguishes $\mathsf{s}\text{-}G^x_k(\mathcal{U}_d)$ from $\mathcal{U}_{k+d}$. Then there is a polynomial p_{G} such that*

$$\mathsf{K}^{p_{\mathsf{G}}(nt_D/\varepsilon)}(x \mid \beta) \leq k + \log p_{\mathsf{G}}(nt_D/\varepsilon).$$

Proof. By Lemma 2.5, with probability at least $\alpha := 1/q(n/\varepsilon)$ over the choice of internal randomness of D and R, and random seed $z \in \{0,1\}^d$, the polynomial-time algorithm $R^D(A(x,z),z;r)$ outputs x, where r is the random string used as internal randomness of R and D. By "hard-wiring" x, we get a polynomial-size circuit C that takes as input random strings z, r, and accepts iff $R^D(A(x,z),z;r)$ outputs x.

By Theorem 2.4, there is an $(\alpha/2)$-PRG IW with seed length $O(\log(\mathsf{poly}(t_D/\alpha)))$ for C. Hence, there is a seed σ of IW such that $R^D(A(x,z_\sigma),z_\sigma;r_\sigma)$ outputs x, when $z_\sigma \circ r_\sigma = \mathsf{IW}(\sigma)$. So, given $\sigma \in \{0,1\}^{O(\log(\mathsf{poly}(t_D/\alpha)))}$ and $a = A(x,z_\sigma) \in \{0,1\}^k$, we can reconstruct x in deterministic time $\mathsf{poly}(t_D/\alpha)$ (which consists of the runtime of $\mathsf{IW}(\sigma)$ plus the runtime of $R^D(a,z_\sigma;r_\sigma)$. □

Lemma 2.8 (pK^t Reconstruction using G^x_k)**.** *For $\varepsilon > 0$, $x \in \{0,1\}^n$, and $k \in \mathbb{N}$ satisfying $k \leq 2n$, and $d = O(\log^3 n)$, let D be a randomized algorithm that takes an advice string β and runs in time t_D such that D ε-distinguishes $\mathsf{s}\text{-}G^x_k(\mathcal{U}_d)$ from $\mathcal{U}_{d+k}$. Then there is a polynomial p_{G} such that*

$$\mathsf{pK}^{p_{\mathsf{G}}(nt_D/\varepsilon)}(x \mid \beta) \leq k + O(\log(n/\varepsilon)).$$

Proof. By Lemma 2.5, with probability at least $\alpha := 1/q(n/\varepsilon)$ over the choice of internal randomness of D and R, and random seed $z \in \{0,1\}^d$, the algorithm $R^D(a,z)$ outputs x, when $a = A(x,z) \in \{0,1\}^k$. It follows that, for some polynomial p (the runtime of the oracle algorithm $R^{(\cdot)}$), we have that $\mathsf{pK}^{p(t_D)}_\alpha(x \mid \beta) \leq k$.

By Lemma 2.1, we conclude that $\mathsf{pK}^{O(p(t_D)\cdot q(n/\varepsilon))}(x \mid \beta) \leq k + O(\log q(n/\varepsilon))$. For $p_G := \max\{p,q\}$, the lemma follows. □

2.3 Natural Properties

We recall the definition of a natural property for circuit size. For a truth table $x \in \{0,1\}^N$, for $N = 2^n$, we denote by $\mathsf{size}(x)$ the size of a smallest Boolean circuit that computes the n-input Boolean function with the truth table x.

Definition 2.9 (Natural Property for Circuit Size [RR97])**.** *A* natural property for circuit size *for truth tables of length $N = 2^n$ with* usefulness *$s(N) < N$ is a predicate $\mathcal{R}\colon \{0,1\}^N \to \{0,1\}$ such that*

1. *for all $x \in \{0,1\}^N$, if $\mathsf{size}(x) \leq s(N)$, then $\mathcal{R}(x) = 1$, and*
2. *$\mathbf{Pr}_{x\in\{0,1\}^N}[\mathcal{R}(x) = 0] \geq 1/2$.*

Such a natural property is called BPP-computable *if there is a randomized polynomial-time algorithm $\mathcal{A}$ such that*

1. *for all $x \in \{0,1\}^N$, if $\mathsf{size}(x) \leq s(N)$, then $\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(x) = 1] \geq 0.9$, and*
2. *$\mathbf{Pr}_{x\in\{0,1\}^N}[\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(x) = 0] \geq 0.9] \geq 1/2$.*

We need the following result of [IKV23]. We use a common definition of the class BPP with advice, where we say that $L \in \mathsf{BPP}/a(n)$, for some advice size $a(n)$ for input length n, if there is a probabilistic polynomial-time Turing machine M such that, for inputs x of length n and for some *good* advice $\alpha \in \{0,1\}^{a(n)}$, $M(x,\alpha)$ either accepts or rejects every given $x \in \{0,1\}^n$ with probability at least $2/3$ over its internal randomness. Note that there is no such acceptance probability guarantee for any different advice string $\alpha' \neq \alpha$.

Theorem 2.10 ([IKV23])**.** *Let $\mathcal{R}$ be a natural property for circuit size for truth tables of length $N = 2^n$ with usefulness $s(N) \geq N^\varepsilon$, for some $0 < \varepsilon < 1$. Then*

1. *$\mathsf{ZPEXP}^{\mathcal{R}} \not\subset \mathsf{P/poly}$, and*
2. *$\mathsf{ZPP}^{\mathcal{R}}/1 \not\subset \mathsf{SIZE}[n^k]$ and $\mathsf{promise\text{-}ZPP}^{\mathcal{R}} \not\subset \mathsf{SIZE}[n^k]$ for all $k \in \mathbb{N}$.*

Replacing a natural property oracle $\mathcal{R}$ with an efficient randomized version, we immediately get from Theorem 2.10 the following.

Corollary 2.11 (implicit in [IKV23])**.** *Suppose there is a* BPP*-computable natural property for circuit size for truth tables of length $N = 2^n$ with usefulness $s(N) \geq N^\varepsilon$, for some $0 < \varepsilon < 1$. Then*

1. *$\mathsf{BPEXP} \not\subset \mathsf{P/poly}$, and*
2. *$\mathsf{BPP}/1 \not\subset \mathsf{SIZE}[n^k]$ and $\mathsf{promise\text{-}BPP} \not\subset \mathsf{SIZE}[n^k]$ for all $k \in \mathbb{N}$.*

Proof. A natural oracle $\mathcal{R}$ is only used as a distinguisher between a distribution over "easy" strings and the uniform distribution. A BPP-computable natural property gives rise to a randomized such distinguisher, which is not required to satisfy the BPP-promise (of rejecting or accepting with high probability) on all inputs. The base ZPEXP algorithm will run a randomized algorithm for a BPP-computable natural property to simulate its oracle access to $\mathcal{R}$, becoming a BPEXP algorithm in item (1), and a BPP/1 or promise-BPP algorithm in item (2). □

We define a natural property for K^t (and for pK^t) by analogy with the natural property for circuit size above.

Definition 2.12 (Natural Properties for K^t and pK^t)**.** *A* P-computable natural property *for $\mu \in \{\mathsf{K}^t, \mathsf{pK}^t\}$ on n-bit inputs, with* usefulness $s(n,t)$*, for some $s(n,t) < n$, is a polynomial-time algorithm $\mathcal{A}$ satisfying the following: For some polynomial p, we have for all sufficiently large $n \in \mathbb{N}$ and $t \geq p(n)$ that*

1. *for all $x \in \{0,1\}^n$, if $\mu(x) \leq s(n,t)$, then $\mathcal{A}(x, 1^t) = 1$, and*
2. $\mathbf{Pr}_{x \in \{0,1\}^n}\left[\mathcal{A}(x, 1^t) = 0\right] \geq 1/2$.

Such a natural property is said to be BPP-computable *if there is a randomized polynomial-time algorithm $\mathcal{A}$ such that, for some polynomial p, we have for all large $n \in \mathbb{N}$ and $t \geq p(n)$ that*

1. *for all $x \in \{0,1\}^n$, if $\mu(x) \leq s(n,t)$, then $\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(x, 1^t) = 1] \geq 0.9$, and*
2. $\mathbf{Pr}_{x \in \{0,1\}^n}\left[\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(x, 1^t) = 0] \geq 0.9\right] \geq 1/2$.

We also define a natural property for *conditional* K^t and pK^t as follows.

Definition 2.13 (Natural Property for Conditional K^t and pK^t)**.** *For $\mu \in \{\mathsf{K}^t, \mathsf{pK}^t\}$, a* natural property for conditional μ on n-bit strings with usefulness $s(n,t)$ *is a predicate $\mathcal{P}\colon \{0,1\}^* \times \{0,1\}^* \times 1^* \to \{0,1\}$ satisfying the following: For some polynomial p, we have for all large $n, m \in \mathbb{N}$ and $t \geq p(n+m)$ that*

1. *for all $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, if $\mu(x \mid y) \leq s(n,t)$, then $\mathcal{P}(x, y, 1^t) = 1$, and*
2. *for all $y \in \{0,1\}^m$, $\mathbf{Pr}_{x \in \{0,1\}^n}[\mathcal{P}(x, y, 1^t) = 0] \geq 1/2$.*

A BPP*-computable natural property for conditional μ is defined similarly to that for unconditional μ in Definition 2.12 above.*

Below we consider natural properties with usefulness $n - \delta(n,t)$, for monotone non-decreasing functions $\delta\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. When $\delta(n,t)$ depends only on one of its inputs, we omit the mention of the other input, and think of δ as a function of one input only.

2.4 Range Avoidance

Definition 2.14 (Avoid)**.** *The Range Avoidance Problem [Kle+21; Kor21] is the following search problem: Given a circuit $C\colon \{0,1\}^n \to \{0,1\}^m$, for some $m > n$, find a string $z \in \{0,1\}^m$ such that, for all $x \in \{0,1\}^n$, $C(x) \neq z$. Let* Avoid *be the Range Avoidance problem for circuits $C\colon \{0,1\}^n \to \{0,1\}^m$ with $m > (1+\varepsilon)n$ for some (arbitrarily small) constant $\varepsilon > 0$ (e.g., think $m > (1.01) \cdot n$).*

Remark 2.15. *The complexity of the Range Avoidance Problem appears to be sensitive to the assumed stretch of circuits $C\colon \{0,1\}^n \to \{0,1\}^m$. [Kle+21; Kor21] considered the version with $m \geq n+1$, and showed that $m = n+1$ and $m = 2n$ are equivalent under P^{NP}-reductions. Such an equivalence is not known under P-reductions. A similar issue arises also in the context of bounded arithmetic where one seems to need different variants of the dual weak pigeonhole principle axiom for different systems of randomized polynomial-time reasoning (cf. [Jeř07]). In the present paper, we only consider the version of Range Avoidance with at least linear stretch.*

For every $y \in \{C(x) \mid x \in \{0,1\}^n\}$, we can reconstruct y if we know some pre-image $x \in \{0,1\}^n$ such that $C(x) = y$. Therefore, $\mathsf{K}^t(y \mid C) \leq n$, for $t = p(|C|)$, where p is the polynomial bound on the time required to evaluate a given circuit C on a given input. Hence, to solve Avoid for a given circuit C, it suffices to find a string of high conditional Kolmogorov complexity $\mathsf{K}^t(- \mid C)$. In fact, as observed in [RSW22], the two tasks are polynomial-time equivalent: one can solve Avoid in polynomial time iff one can find "incompressible" strings for conditional K^t.[7]

Actually, finding a string of high conditional pK^t complexity allows us to solve a natural generalization of the Range Avoidance problem, which we term Collective Range Avoidance (CRA).

Definition 2.16 (Collective Range Avoidance (CRA))**.** *The* 1/3-Collective Range Avoidance (CRA) *problem is the following search problem: Given a circuit $C\colon \{0,1\}^\ell \times \{0,1\}^n \to \{0,1\}^m$ for $m > (1+\varepsilon)n$ for some constant $\varepsilon > 0$, find a string $z \in \{0,1\}^m$ such that, for at least $1/3$ of strings $r \in \{0,1\}^\ell$, the string z is not in the range of the circuit $C_r\colon \{0,1\}^n \to \{0,1\}^m$ where $C_r(x) = C(r,x)$.*

Note that the usual Range Avoidance problem is a special case of the CRA problem defined above when $\ell = 0$.

Consider any string $y \in \{0,1\}^m$ that is in the range of at least $2/3$ of circuits $C_r\colon \{0,1\}^n \to \{0,1\}^m$, over uniformly random $r \in \{0,1\}^\ell$. Then, with probability at least $2/3$ over $r \in \{0,1\}^\ell$, there is a string $x_r \in \{0,1\}^n$ such that $C(r, x_r) = y$. This implies that $\mathsf{pK}^t(y \mid C) \leq n$, for some $t = \mathsf{poly}(|C|)$ (the time needed to evaluate the circuit C on a given input). Thus, to solve 1/3-CRA problem for a given circuit $C\colon \{0,1\}^\ell \times \{0,1\}^n \to \{0,1\}^m$, it suffices to find a string $z \in \{0,1\}^m$ with $\mathsf{pK}^t(z \mid C) > n$, for some polynomial $t = \mathsf{poly}(|C|)$.

For completeness, we state the following observation; the proof similar to the one in [RSW22] for the case of unary Range Avoidance.

Lemma 2.17. *The search problem* (1/3)-CRA *is in polynomial time if and only if, for every constant $0 < \gamma < 1$, there is a polynomial-time algorithm that, given 1^m, 1^t, and $y \in \{0,1\}^\ell$, finds a string $z \in \{0,1\}^m$ with $\mathsf{pK}^t(z \mid y) \geq (1-\gamma) \cdot m$.*

Proof. In the forward direction, given 1^m, 1^t, $y \in \{0,1\}^\ell$, and $0 < \gamma < 1$, set $n = \lceil \alpha \cdot m \rceil$, for some constant $0 < \alpha < 1$ to be determined, define a circuit $C_y\colon \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^m$, which simulates the following Turing machine:

> On inputs $r \in \{0,1\}^t$ and $x \in \{0,1\}^n$, decompose $x = uv$ where $|u| = \log n$. Let $0 \leq d < n$ be the integer encoded by the binary string u. Let w be the last $\min\{d, n - \log n\}$ bits of x. Run the universal Turing machine $U(w, r, y)$ for at most t steps, and output the binary string it outputs, padded or trimmed to be exactly of length m.

[7] Actually, [RSW22] observes this equivalence for the case of unary Avoid and K^t, but it immediately generalizes to the case of Avoid and conditional K^t.

Note that C_y is of size $\mathsf{poly}(n, \ell, t)$. Applying an assumed polynomial-time algorithm for 1/3-CRA to C_y, we get a string $z \in \{0,1\}^m$ such that, for all sufficiently large $m \in \mathbb{N}$,

$$\begin{aligned} \mathsf{pK}^t(z \mid y) &> n - \log n \\ &\geq \alpha \cdot m - \log m \\ &\geq (1-\gamma) \cdot m, \end{aligned}$$

if we set $\alpha = 1 - (\gamma/2)$.

In the reverse direction, For a given circuit $C\colon \{0,1\}^\ell \times \{0,1\}^n \to \{0,1\}^m$, with $m > (1+\varepsilon)n$ for some constant $\varepsilon > 0$, consider any string $z \in \{0,1\}^m$ with

$$\mathsf{pK}^t(z \mid C) \geq (1-\nu)m,$$

where $t = p(|C|)$ for some polynomial p, and $\nu = \varepsilon/(1+\varepsilon)$. By our choice of parameters, we get that $\mathsf{pK}^t(z \mid C) > n$. Hence, by our earlier observation, the string z is a solution to 1/3-CRA, and by our assumption, such a z can be found in polynomial time. □

3 Natural Properties vs. Worst-Case Approximation of K^t

3.1 Approximating Time-Bounded Kolmogorov Complexity

$\mathsf{MK}^t\mathsf{P}$ (also denoted as MINKT) is the problem to determine if $\mathsf{K}^t(x) \leq s$, for given positive integers t and s. We recall the definition of the gap version of $\mathsf{MK}^t\mathsf{P}$.

Definition 3.1 ([Ko91; Hir20b])**.** *For a polynomial τ and a function $\delta\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\mathsf{Gap}_{\tau,\delta}\mathsf{MK}^t\mathsf{P}$ is the following promise problem:*

$$\begin{aligned} \Pi_{\text{YES}} &= \{(x, 1^s, 1^t) \mid \mathsf{K}^t(x) \leq s\}, \\ \Pi_{\text{NO}} &= \{(x, 1^s, 1^t) \mid \mathsf{K}^{\tau(t)}(x) > s + \delta(|x|, t)\}. \end{aligned}$$

Next we consider the Minimum Conditional Time-Bounded Kolmogorov Complexity Problem, denoted $\mathsf{McK}^\mathsf{t}\mathsf{P}$, where

$$\mathsf{McK}^\mathsf{t}\mathsf{P} = \{(x, y, 1^s, 1^t) \mid \mathsf{K}^t(x \mid y) \leq s\}.$$

Definition 3.2. *For a polynomial τ and a function $\delta\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, define $\mathsf{Gap}_{\tau,\delta}\mathsf{McK}^\mathsf{t}\mathsf{P}$ as the following promise problem:*

$$\begin{aligned} \Pi_{\text{YES}} &= \{(x, y, 1^s, 1^t) \mid \mathsf{K}^t(x \mid y) \leq s\}, \\ \Pi_{\text{NO}} &= \{(x, y, 1^s, 1^t) \mid \mathsf{K}^{\tau(t)}(x \mid y) > s + \delta(|x|, t)\}. \end{aligned}$$

Lemma 3.3. *$\mathsf{Gap}_{\tau,\delta}\mathsf{McK}^\mathsf{t}\mathsf{P} \in \mathsf{promise\text{-}P}$ iff there is a polynomial-time algorithm $\tilde{K}$ such that, for all $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ and every large enough integer t,*

$$\mathsf{K}^{\tau(t)}(x \mid y) - \delta(n,t) \leq \tilde{K}(x, y, 1^t) \leq \mathsf{K}^t(x \mid y). \tag{6}$$

Similarly, $\mathsf{Gap}_{\tau,\delta}\mathsf{MK}^t\mathsf{P} \in \mathsf{promise\text{-}P}$ iff there is a polynomial-time algorithm $\tilde{K}$ such that, for all $x \in \{0,1\}^n$ and every large enough integer t,

$$\mathsf{K}^{\tau(t)}(x) - \delta(n,t) \leq \tilde{K}(x, 1^t) \leq \mathsf{K}^t(x). \tag{7}$$

Proof. In the forward direction, define $\tilde{K}(x, y, 1^t)$ to be the smallest integer $0 \leq s \leq 2|x|$ such that the assumed polynomial-time algorithm for $\mathsf{promise\text{-}Gap}_{\tau,\delta}\mathsf{McK^tP}$ outputs "yes" on input $(x, y, 1^s, 1^t)$. Since for this s, we have $(x, y, 1^s, 1^t) \notin \Pi_{\text{NO}}$, we conclude the required lower bound on $\tilde{K}(x, y, 1^t)$ in (6). For the required upper bound on $\tilde{K}(x, y, 1^t)$ in (6), observe that we will get a "yes" for some $s \leq \mathsf{K}^t(x \mid y)$, because we definitely get a "yes" when $s = \mathsf{K}^t(x \mid y)$.

In the reverse direction, define an algorithm for $\mathsf{Gap}_{\tau,\delta}\mathsf{McK^tP}$ as follows: "On input $(x, y, 1^s, 1^t)$, accept iff $\tilde{K}(x, y, 1^t) \leq s$." By the definition of $\tilde{K}$, this algorithm will accept all instances in Π_{YES}, and reject all instances in Π_{NO}.

The proof for the case of unconditional K^t complexity is identical to the proof above. □

3.2 Average-Case vs. Worst-Case of Approximating K^t

A natural property for (conditional) K^t is closely related to the worst-case algorithm for the gap-version of the Minimum (conditional) K^t complexity Problem.

Lemma 3.4 (Average-to-Worst-Case Reductions for (conditional) K^t)**.**

- *If, for some polynomial τ and a function $\delta\colon \mathbb{N} \to \mathbb{N}$, $\mathsf{Gap}_{\tau,\delta}\mathsf{McK^tP} \in \mathsf{promise\text{-}P}$, then there is a P-computable natural property for conditional $\mathsf{K}^t(x \mid y)$ on n-bit strings x with usefulness $s(n, t) = n - \delta(n, t) - 3$.*
- *Similarly, if $\mathsf{Gap}_{\tau,\delta}\mathsf{MK^tP} \in \mathsf{promise\text{-}P}$, then there is a P-computable natural property for $\mathsf{K}^t(x)$ on n-bit strings x with usefulness $s(n, t) = n - \delta(n, t) - 3$.*

Proof. Let $\mathcal{K}$ be a polynomial-time algorithm for $\mathsf{Gap}_{\tau,\delta}\mathsf{McK^tP}$. For $s(n, t) = n - \delta(n, t) - 2$, define an algorithm $\mathcal{A}$ as follows: "On input $(x, y, 1^t)$, accept iff $\mathcal{K}(x, y, 1^{s(|x|,t)}, 1^t)$ accepts."

For correctness, observe that all $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ with $\mathsf{K}^t(x \mid y) \leq s(n, t)$ are accepted because $(x, y, 1^{s(|x|,t)}, 1^t) \in \Pi_{\text{YES}}$. For every y, we have by Lemma 2.2, for at least $1/2$ of random $x \in \{0,1\}^n$, $\mathsf{pK}(x \mid y) > n - 3$. Hence, for these random x, $\mathsf{K}^{\tau(t)}(x \mid y) \geq \mathsf{pK}(x \mid y) > n - 3 = s(n, t) + \delta(n, t)$, and so $(x, y, 1^{s(|x|,t)}, 1^t) \in \Pi_{\text{NO}}$.

The proof for the case of unconditional K^t is identical to the one above, with y set to be the empty string ϵ. □

Lemma 3.5 (Worst-to-Average-Case Reductions for (conditional) K^t)**.** *Assume $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$.*

- *If there is a P-computable natural property for conditional $\mathsf{K}^t(x \mid y)$ on n-bit strings x with usefulness $s(n, t) = n - \delta(n, t)$, then for some polynomial τ, we have $\mathsf{Gap}_{\tau,\delta'}\mathsf{McK^tP} \in \mathsf{promise\text{-}P}$, where $\delta'(n, t) = \delta(3n, 2t) + O(\log(nt))$.*
- *Similarly, if there is a P-computable natural property for $\mathsf{K}^t(x)$ on n-bit strings x with usefulness $s(n, t) = n - \delta(n, t)$, then for some polynomial τ, we have $\mathsf{Gap}_{\tau,\delta'}\mathsf{MK^tP} \in \mathsf{promise\text{-}P}$, where $\delta'(n, t) = \delta(3n, 2t) + O(\log(nt))$.*

Proof. We use the ideas from [Hir18; Hir20b; Hir20c], and the generator G^x_k from Definition 2.6. We consider the case of conditional K^t complexity first; the case of unconditional K^t is proved by the same argument.

Let $\mathcal{A}(x, y, 1^t)$ be a polynomial-time $p(n, t)$ algorithm for the assumed natural property for conditional $\mathsf{K}^t(x \mid y)$ on n-bit strings x. For given $x \in \{0,1\}^n, y \in \{0,1\}^*$, consider the generator

$$\mathsf{s}\text{-}G^x_k\colon \{0,1\}^d \to \{0,1\}^{d+k}$$

from Definition 2.6, for some $0 \le k \le 2n$ to be determined, and for $d = c\log^3 n$ for some constant $c > 0$. We have, for any t larger than some polynomial in n, and for any seed $z \in \{0,1\}^d$, that

$$\mathsf{K}^{2t}(\mathsf{s}\text{-}G_k^x(z) \mid y) \le \mathsf{K}^t(x \mid y) + |z| + c' \log\log n, \tag{8}$$

for some constant $c' > 0$ (where the $c' \log\log n$ bits are used to define the length of z). For any given $\sigma \in \mathbb{N}$, if $\mathsf{K}^t(x \mid y) \le \sigma$, then for $k = \sigma + c' \log\log n + \delta(3n, 2t)$, we have by (8) that

$$\begin{aligned}\mathsf{K}^{2t}(\mathsf{s}\text{-}G_k^x(z) \mid y) &\le \sigma + |z| + c' \log\log n \\ &\le (k+d) - \delta(k+d, 2t) \\ &= s(k+d, 2t).\end{aligned}$$

So, $\mathcal{A}(\mathsf{s}\text{-}G_k^x(z), y, 1^{2t})$ accepts with probability 1 over $z \in \{0,1\}^d$ (by the definition of a natural property on $(k+d)$-bit strings, with usefulness $s(k+d, 2t)$).

On the other hand, for every y, we have that $\mathcal{A}(w, y, 1^{2t})$ rejects at least $1/2$ of random $w \in \{0,1\}^{k+d}$. If it were the case that $\mathcal{A}(\mathsf{s}\text{-}G_k^x(z), y, 1^{2t})$ accepts with probability at least 0.6 over $z \in \{0,1\}^d$, then $\mathcal{A}(-, y, 1^{2t})$ would be a (0.1)-distinguisher between $\mathsf{s}\text{-}G_k^x$ and $\mathcal{U}_{d+k}$, using advice y. Hence, by Lemma 2.7, we would get that

$$\begin{aligned}\mathsf{K}^{p(t)}(x \mid y) &\le k + O(\log(nt)) \\ &= \sigma + c' \log\log n + \delta(3n, 2t) + O(\log(nt)) \\ &= \sigma + \delta(3n, 2t) + O(\log(nt)) \\ &= \sigma + \delta'(n, t),\end{aligned}$$

for some polynomial p. This means that $(x, y, 1^\sigma, 1^t)$ is not in Π_{NO} of $\mathsf{Gap}_{\tau,\delta'}\mathsf{McK^tP}$ for $\tau = p$. In other words, for every instance $(x, y, 1^\sigma, 1^t)$ in Π_{NO} of $\mathsf{Gap}_{\tau,\delta'}\mathsf{McK^tP}$, $\mathcal{A}(\mathsf{s}\text{-}G_k^x(z), y, 1^{2t})$ accepts with probability less than 0.6 over random $z \in \{0,1\}^d$.

We conclude that the following algorithm correctly solves $\mathsf{Gap}_{\tau,\delta'}\mathsf{McK^tP}$:

> On input $(x, y, 1^\sigma, 1^t)$, set $k = \sigma + c' \log\log n + \delta(3n, 2t)$. Estimate, to within an additive error 0.1, the probability over $z \in \{0,1\}^d$ that $\mathcal{A}(\mathsf{s}\text{-}G_k^x(z), y, 1^{2t})$ accepts. If the estimated probability is at least 0.9, then accept; otherwise, reject.

Using the PRG of Theorem 2.4, we can make the algorithm above run in deterministic polynomial time, which concludes the proof for the case of $\mathsf{K}^t(x \mid y)$. The proof for the case of unconditional $\mathsf{K}^t(x)$ is the same as above, with the auxiliary string y set to the empty string ϵ everywhere. □

Combining Lemma 3.4 and Lemma 3.5, we immediately get the following. Below we combine into a single statement (1) an equivalence between an efficient natural property for conditional K^t and an efficient algorithm for $\mathsf{Gap}_{\tau,\delta}\mathsf{McK^tP}$, with (2) a similar equivalence between an efficient natural property for K^t and an efficient algorithm for $\mathsf{Gap}_{\tau,\delta}\mathsf{MK^tP}$. The reader should either drop or keep the word *conditional* (c) on both sides of the stated equivalence.

Corollary 3.6. *Assume* $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$. *The following are equivalent:*

- *For any $C > 1$, there is a polynomial τ such that $\mathsf{Gap}_{\tau,\delta}\mathsf{M(c)K^tP} \in \mathsf{promise\text{-}P}$, for $\delta(n,t) \le n/C$.*
- *For any $D > 1$, there is a P-computable natural property for (conditional) K^t on n-bit strings x with usefulness $s(n,t) = n - n/D$.*

3.3 Case of pK^t

We similarly define the gap version of the minimum (conditional) pK^t measure.

Definition 3.7. *For a polynomial τ and a function $\delta\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$, define $\mathsf{Gap}_{\tau,\delta}\mathsf{McpK^tP}$ as the following promise problem:*

$$\Pi_{\text{YES}} = \{(x, y, 1^s, 1^t) \mid \mathsf{pK}^t(x \mid y) \le s\},$$
$$\Pi_{\text{NO}} = \{(x, y, 1^s, 1^t) \mid \mathsf{pK}^{\tau(t)}(x \mid y) > s + \delta(|x|, t)\}.$$

For the case of unconditional pK^t, we define $\mathsf{Gap}_{\tau,\delta}\mathsf{MpK^tP}$ in the similar way (with $y = \epsilon$).

Note that, by Remark 2.3, $\mathsf{Gap}_{\tau,\delta}\mathsf{McpK^tP} \in \mathsf{promise\text{-}AM}$. We get the following analog of Corollary 3.6.

Corollary 3.8. *The following are equivalent:*

- *For any $C > 1$, there is a polynomial τ such that $\mathsf{Gap}_{\tau,\delta}\mathsf{M(c)pK^tP} \in \mathsf{promise\text{-}BPP}$, for $\delta(n,t) \le n/C$.*
- *For any constant $D > 1$, there is a BPP-computable natural property for (conditional) pK^t on n-bit strings x with usefulness $s(n,t) = n - n/D$.*

Proof sketch. By similar arguments, we can prove analogs of Lemma 3.4 and Lemma 3.5 for the case of $\mathsf{Gap}_{\tau,\delta}\mathsf{M(c)pK^tP}$ and BPP-computable natural properties for (conditional) pK^t. For the analog of Lemma 3.5, we do not assume $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$, and use Lemma 2.8. □

4 Chain Rules from Natural Properties

4.1 Case of K^t

Theorem 4.1 (Chain Rule for Conditional K^t)**.** *Suppose that*

- $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$, *and*
- *there is a P-computable natural property $\mathcal{A}$ for conditional K^t on n-bit inputs with usefulness $s(n,t) = n - \delta(n,t)$, where $\delta(n,t) \le n/C$ for some constant $C > 0$.*

Then there exist constants $c_0, c_1 \in \mathbb{N}$ such that, for all sufficiently large $x_1, \ldots, x_\ell \in \{0,1\}^*$ of lengths $n_1, \ldots, n_\ell$, respectively, for any $\ell \in \mathbb{N}$, for any $y \in \{0,1\}^*$, and for every $t \ge (|y| + N)^{c_0}$, where $N = \sum_{i=1}^{\ell} n_i$, we have

$$\mathsf{K}^t(x_1, \ldots, x_\ell \mid y) \ge \sum_{i=1}^{\ell} \mathsf{K}^{t^{c_1}}(x_i \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - 6N/C. \tag{9}$$

Proof. We will apply generator $G_{k_i}^{x_i}$ from Definition 2.6 to each of the ℓ strings $x_1, \ldots, x_\ell$, with the parameters $k_i \approx \mathsf{K}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1})$ to be determined. We will use the same seed for each of these ℓ generators. Each generator has the seed length $c \log^3 n_0 \le c \log^3 N$ for some constant $c > 0$, where $n_0 = \max\{n_i \mid 1 \le i \le \ell\}$. So it suffices to have a common seed z of length

N. Each of the ℓ generators will use a prefix of the same z, of the appropriate length. We provide more details next.

Given strings $x_1, \ldots, x_\ell$, with $|x_i| = n_i$, for all $1 \le i \le \ell$, and integer parameters $k_1, \ldots, k_\ell$ (to be determined), with each $0 \le k_i \le 2n_i$, we define the following generator $G\colon \{0,1\}^{3N} \to \{0,1\}^{3N}$. For $z \in \{0,1\}^N$, strings $u_i \in \{0,1\}^{2n_i}$, for $1 \le i \le \ell$, define

$$G(z, u_1, \ldots, u_\ell) := z \circ (G^{x_1}_{k_1}(z_1) \circ u'_1) \circ \cdots \circ (G^{x_\ell}_{k_\ell}(z_\ell) \circ u'_\ell),$$

where each z_i is the prefix of z of the length required by $G^{x_i}_{k_i}$, and u'_i is the prefix of u_i of length $2n_i - k_i$, for all $1 \le i \le \ell$. Note that the output of G consists of z and ℓ blocks $(G^{x_i}_{k_i}(z_i) \circ u'_i)$, for $1 \le i \le \ell$, where each ith block is of length exactly $2n_i$. Both the total input length and the total output length of G is

$$M := |z| + \sum_{i=1}^{\ell} 2n_i = N + 2N = 3N.$$

First, we show how to choose the parameters k_i, $1 \le i \le \ell$, to make G a $(1/4)$-pseudorandom generator against all time $\tau(3N + |y|, 2t)$-time algorithms using advice (y, N, t), where $\tau(n + |y|, t)$ is the polynomial runtime of the assumed natural property $\mathcal{A}(-, y, 1^t)$ for conditional $\mathsf{K}^t(- \mid y)$ on n-bit strings.[8]

We will use a hybrid argument. Suppose there is a $\tau(3N + |y|, 2t)$-time distinguisher D, using advice (y, N, t), between the uniform distribution and the output of the generator G, with a distinguishing probability at least $1/4$. Define hybrid distributions $\mathcal{D}_0, \ldots, \mathcal{D}_\ell$ so that, for each $0 \le i \le \ell$, and uniform distribution Z over $\{0,1\}^N$,

$$\mathcal{D}_i := Z \circ (G^{x_1}_{k_1}(Z_1) \circ U'_1) \circ \cdots \circ (G^{x_i}_{k_i}(Z_i) \circ U'_i) \circ \mathcal{U}_{2N - \sum_{1 \le j \le i} 2n_j},$$

where each Z_j is the prefix of Z of appropriate length, and each U'_j is the uniform distribution over binary strings of length $2n_j - k_j$, for all $1 \le j \le i$; the distributions $Z, U'_1, \ldots, U'_i$ are mutually independent. (Note that we do not need to know individual lengths $n_{i+1}, \ldots, n_\ell$, but can compute the total length of the remaining uniformly random string in $\mathcal{D}_i$, if we know N and $n_1, \ldots, n_i$.)

By assumption, D is a $(1/4)$-distinguisher between $\mathcal{D}_0$ and $\mathcal{D}_\ell$. By a hybrid argument, there must exist an index $1 \le i \le \ell$ such that D is a $(1/(4\ell))$-distinguisher between $\mathcal{D}_{i-1}$ and $\mathcal{D}_i$. We claim that D can be used to define the following randomized distinguisher D' between $\mathsf{s}\text{-}G^{x_i}_{k_i}(z_i)$, for $z_i \in \mathcal{U}_{|z_i|}$, and $\mathcal{U}_{|z_i|+k_i}$, given as advice $y, N, t, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}$:

> The distinguisher D' on inputs z_i and $w \in \{0,1\}^{k_i}$, will do as follows:
>
> - Sample a uniformly random string $\alpha \in \{0,1\}^{N - |z_i|}$ to define $z = z_i \circ \alpha$ of length N, which will be placed in the position of Z in $\mathcal{D}_i$.
> - For the G positions $1 \le j \le i-1$ in $\mathcal{D}_i$, we know both x_j *and* the parameter $0 \le k_j \le 2n_j$, and so we can sample $G^{x_j}_{k_j}(Z_j)$, as well as a uniformly random string of length $2n_j - k_j$ to make a string in this block of total length $2n_j$.
> - Place the string w in the ith G position in $\mathcal{D}_i$ (to be extended to the right total length with a uniformly random string next).

[8] It would suffice for us if G were $(1/4)$-fooling only *one particular algorithm*, namely $\mathcal{A}(-, y, 1^{2t})$ on inputs of length $3N$. We will make use of this observation later to get a chain rule for unconditional K^t in Theorem 4.2.

- For the remaining suffix (consisting of independent uniformly random strings of length $2n_i - k_i$ and $2N - \sum_{j=1}^{i} 2n_j$), sample a uniformly random string of total length
$$2N - \sum_{1 \le j \le i-1} 2n_j - k_i,$$
which we can do since we know N, $x_1, \ldots, x_{i-1}$, and w.

Finally, D' simulates D on the resulting binary string of length $3N$, accepting iff D accepts.

The runtime of the randomized distinguisher D' is that of D plus the time to sample the other positions of the hybrid distribution. The latter is dominated by the time needed to evaluate $G_{k_j}^{x_j}(Z_j)$, for all $1 \le j \le i-1$. Since G_k^x is computable in time $p(|x|)$ for some polynomial p, we conclude that all $i - 1 \le N$ such evaluations can be done in time $N \cdot p(N) \le p'(N)$, for some polynomial p'. Thus we can upperbound the runtime of D' by $\tau(3N + |y|, 2t) + p'(N) \le \mathsf{poly}(t)$, for t larger than $N + |y|$.

By Lemma 2.7 we get that, for a sufficiently large c_0 such that $t \ge N^{c_0}$, and for $q'(t) = p_{\mathsf{G}}(\mathsf{poly}(t))$,
$$\mathsf{K}^{q'(t)}(x_i \mid y, N, t, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) \le k_i + \log q'(t).$$
Since N and t can be described with $O(\log t)$ bits, the above inequality implies that, for, say, $q(t) = (q'(t))^2$, we get
$$\mathsf{K}^{q(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) \le k_i + \log q(t). \tag{10}$$

Next we would like to set each k_i, for $1 \le i \le \ell$, so that the inequality in (10) *cannot* hold, and so G is (1/4)-pseudorandom with respect to $\mathcal{A}(-, y, 1^{2t})$. For example, setting
$$k_i = \max\{0, \mathsf{K}^{q(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) - \log q(t) - 1\}$$
would do the job. (Note that if $k_i = 0$, then $\mathcal{D}_{i-1}$ and $\mathcal{D}_i$ are identical, and hence indistinguishable by any algorithm D.) However, these k_is are not necessarily efficiently computable. It will be necessary for us to have some polynomial-time computable k_is satisfying the following two conditions:

1. each k_i is smaller than $\mathsf{K}^{q(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) - \log q(t)$, and so the inequality in (10) cannot hold for it (implying that G is pseudorandom with respect to $\mathcal{A}(-, y, 1^{2t})$), and

2. each k_i is (approximately) larger than some $\mathsf{K}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1})$ (which is used together with the usefulness property of $\mathcal{A}$ to conclude deriving the chain rule for K^t).

We show how to define such k_is next.

First, by Lemma 3.5, for some polynomial σ, we have that $\mathsf{Gap}_{\sigma,\delta'}\mathsf{McK}^{\mathsf{t}}\mathsf{P} \in \mathsf{promise\text{-}P}$, where $\delta'(n,t) = \delta(3n, 2t) + O(\log(nt))$. The latter implies by Lemma 3.3 that there is a polynomial-time algorithm $\tilde{K}$ such that for all $x, y \in \{0,1\}^*$ and large enough $t \ge \mathsf{poly}(|x|, |y|)$,
$$\mathsf{K}^{\sigma(t)}(x \mid y) - \delta(3|x|, 2t) - O(\log(t)) \le \tilde{K}(x, y, 1^t) \le \mathsf{K}^t(x \mid y). \tag{11}$$

Suppose we set
$$k_1 = \max\{0, \tilde{K}(x_1, y, 1^{q(t)}) - \log q(t) - 1\}.$$

Since $\tilde{K}(x_1, y, 1^{q(t)}) \leq \mathsf{K}^{q(t)}(x_1 \mid y)$, we get that inequality in (10) cannot hold for this k_1. Once we have computed k_1 this way, we can compute

$$k_2 = \max\{0, \tilde{K}(x_2, (y, x_1, k_1), 1^{q(t)}) - \log q(t) - 1\}.$$

Again, it is easy to see that inequality in (10) cannot hold for this k_2. We continue this way to compute all $k_1, \ldots, k_\ell$, where

$$k_i = \max\{0, \tilde{K}(x_i, (y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}), 1^{q(t)}) - \log q(t) - 1\}. \tag{12}$$

By their definition, we get that G for these $k_1, \ldots, k_\ell$ is $(1/4)$-pseudorandom against $\tau(3N, 2t)$-time algorithms with advice (y, N, t), and hence against $\mathcal{A}(-, y, 1^{2t})$.

Set $s := M - \delta(M, 2t)$. By the definition of a natural property, $\mathcal{A}(-, y, 1^{2t})$ rejects with probability at least $1/2$ on the uniform distribution $\mathcal{D}_0$. Hence, by the $(1/4)$-pseudorandomness of G, $\mathcal{A}(-, y, 1^{2t})$ rejects with probability at least $1/2 - 1/4 = 1/4$ on the outputs of G.

Next, by averaging, there exists a string $z \in \{0,1\}^N$ and strings $u'_1 \in \{0,1\}^{2n_1 - k_1}, \ldots, u'_\ell \in \{0,1\}^{2n_\ell - k_\ell}$ such that

$$\mathcal{A}(z \circ G^{x_1}_{k_1}(z_1) \circ u'_1 \circ \cdots \circ G^{x_\ell}_{k_\ell}(z_\ell) \circ u'_\ell, y, 1^{2t}) = 0,$$

where, as before, each z_i is the prefix of z of appropriate length. By the usefulness s of the natural property (and recalling the definition of $s = M - \delta(M, 2t)$), we get that

$$\mathsf{K}^{2t}\left(z \circ G^{x_1}_{k_1}(z_1) \circ u'_1 \circ \cdots \circ G^{x_\ell}_{k_\ell}(z_\ell) \circ u'_\ell \mid y\right) > s = \sum_{i=1}^{\ell} k_i + \sum_{i=1}^{\ell} |u'_i| + |z| - \delta(M, 2t). \tag{13}$$

On the other hand, for every seed $z \in \{0,1\}^N$, and any $u'_1, \ldots, u'_\ell$, we have

$$\mathsf{K}^{2t}\left(z \circ G^{x_1}_{k_1}(z_1) \circ u'_1 \circ \cdots \circ G^{x_\ell}_{k_\ell}(z_\ell) \circ u'_\ell \mid y\right) \leq \mathsf{K}^t(x_1, \ldots, x_\ell \mid y) + |z| + \sum_{i=1}^{\ell} |u'_i| + c' \cdot \ell \cdot \log(N/\ell), \tag{14}$$

where the last term is to take into account encoding of all n_i's and k_i's, which can be done using at most $c' \cdot \sum_{i=1}^{\ell} \log n_i$ bits, for some constant $c' > 0$; using the AM-GM inequality, we have $\sum_{i=1}^{\ell} \log n_i \leq \ell \cdot \log(N/\ell)$.

Combining (13) and (14), we get that

$$\mathsf{K}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} k_i - \delta(M, 2t) - \ell \cdot O(\log N). \tag{15}$$

By (11) and (12), we have that, for each $1 \leq i \leq \ell$,

$$\begin{aligned} k_i &\geq \tilde{K}(x_i, (y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}), 1^{q(t)}) - \log q(t) - 1 \\ &\geq \mathsf{K}^{\sigma(q(t))}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) - \delta(3|x_i|, 2t) - O(\log t) - \log q(t) - 1. \end{aligned}$$

Plugging this lower bound on k_is in (15) (and recalling that $\delta(n, t) \leq n/C$ for some constant $C > 0$), we then get that

$$\mathsf{K}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{K}^{\sigma(q(t))}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) - 3N/C - \delta(M, 2t) - \ell \cdot O(\log t).$$

Finally, since we can compute $k_1, \ldots, k_\ell$ in some fixed polynomial time $\mathsf{poly}(N,t) \le \mathsf{poly}(t)$, we get, for each $1 \le i \le \ell$,

$$\mathsf{K}^{\sigma(q(t))}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) \ge \mathsf{K}^{\mathsf{poly}(\sigma(q(t)))}(x_i \mid y, x_1, \ldots, x_{i-1}).$$

The latter implies

$$\mathsf{K}^t(x_1, \ldots, x_\ell \mid y) \ge \sum_{i=1}^{\ell} \mathsf{K}^{\mathsf{poly}(\sigma(q(t)))}(x_i \mid y, x_1, \ldots, x_{i-1}) - 6N/C - \ell \cdot O(\log t),$$

as required. □

Theorem 4.2 (Chain Rule for K^t)**.** *Suppose that*

- $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$*, and*
- *there is a* P*-computable natural property* $\mathcal{A}$ *for* K^t *on* n*-bit inputs with usefulness* $s(n,t) = n - \delta(n,t)$*.*

Then there exist constants $c_0, c_1 \in \mathbb{N}$ *such that, for all sufficiently large* $x_1, \ldots, x_\ell \in \{0,1\}^*$ *of lengths* $n_1, \ldots, n_\ell$*, respectively, for any* $\ell \in \mathbb{N}$*, and for every* $t \ge N^{c_0}$*, where* $N = \sum_{i=1}^{\ell} n_i$*, we have*

$$\mathsf{K}^t(x_1, \ldots, x_\ell) \ge \sum_{i=1}^{\ell} \mathsf{K}^{t^{c_1}}(x_i \mid x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - \delta(3N, 2t). \tag{16}$$

Proof. We would like to argue as in Theorem 4.1 above, with the auxiliary string y set to be the empty string ϵ everywhere. However, in the proof of Theorem 4.1 we used the assumed polynomial-time natural property for *conditional* K^t to define *efficiently computable* parameters $k_1, \ldots, k_\ell$ of the generators $G^{x_1}_{k_1}, \ldots, G^{x_\ell}_{k_\ell}$. In the current theorem, on the other hand, we assume a P-computable natural property for the *unconditional* K^t only. So we must compute the necessary parameters $k_1, \ldots, k_\ell$ in a different way. We will borrow an idea from [HN25].

We will use the same generator G as in in the proof of Theorem 4.1. For $z \in \{0,1\}^N$, strings $u_i \in \{0,1\}^{2n_i}$, for $1 \le i \le \ell$, and for the parameters $0 \le k_i \le 2n_i$, for $1 \le i \le \ell$, to be determined, define

$$G(z, u_1, \ldots, u_\ell) := z \circ (G^{x_1}_{k_1}(z_1) \circ u'_1) \circ \cdots \circ (G^{x_\ell}_{k_\ell}(z_\ell) \circ u'_\ell),$$

where each z_i is the prefix of z of the length required by $G^{x_i}_{k_i}$, and u'_i is the prefix of u_i of length $2n_i - k_i$, for all $1 \le i \le \ell$. The total output length (and input length) of G is $M := 3N$.

As in the proof of Theorem 4.1, we want to compute $k_1, \ldots, k_\ell$ satisfying the following:

1. the resulting generator G $(1/4)$-fools the natural property $\mathcal{A}(-, 1^{2t})$ on inputs in $\{0,1\}^M$, and
2. each k_i is (approximately) larger than some $\mathsf{K}^{\mathsf{poly}(t)}(x_i \mid x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1})$.

Note that condition (1) above is weaker than condition (1) in the proof of Theorem 4.1, where we required that each k_i is smaller than some $\mathsf{K}^{\mathsf{poly}'(t)}(x_i \mid x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1})$. Condition (1) in the proof of Theorem 4.1 implied that G is $(1/4)$-pseudorandom against *all* $\tau(3N + |y|, 2t)$-time algorithms, whereas it would have sufficed to fool just one such algorithm, namely the algorithm $\mathcal{A}$ for the natural property for conditional K^t. Our condition (1) in the current theorem asks exactly that: just fool $\mathcal{A}(-, 1^{2t})$ on inputs in $\{0,1\}^M$.

We define hybrid distributions $\mathcal{D}_0, \ldots, \mathcal{D}_\ell$ so that, for each $0 \le i \le \ell$, and the uniform distribution Z over $\{0,1\}^N$,

$$\mathcal{D}_i := Z \circ (G_{k_1}^{x_1}(Z_1) \circ U_1') \circ \cdots \circ (G_{k_i}^{x_i}(Z_i) \circ U_i') \circ \mathcal{U}_{2N - \sum_{1 \le j \le i} 2n_j},$$

where each Z_j is the prefix of Z of appropriate length, and each U_j' is the independent uniform distribution over binary strings of length $2n_j - k_j$, for all $1 \le j \le i$. As shown in the proof of Theorem 4.1, each distribution $\mathcal{D}_i$, for $1 \le i \le \ell$, can be sampled by a randomized algorithm in time $\mathsf{poly}(N)$, given advice $x_1, \ldots, x_i, k_1, \ldots, k_i, N$. Hence, $\mathcal{D}_i$ can be sampled by a circuit $\mathcal{C}_i$ of size $\mathsf{poly}(N)$.

We will use the PRG of Theorem 2.4 to estimate $\Pr_r[\mathcal{A}(\mathcal{C}_0(r), 1^{2t}) = 1]$ and, for every $0 \le k_1 \le 2n_1$, $\Pr_r[\mathcal{A}(\mathcal{C}_1(r), 1^{2t}) = 1]$, to within an additive error $1/(32\ell)$ each. We define $0 \le k_1 \le 2n_1$ to be the *maximum* integer such that the estimates of $\Pr_r[\mathcal{A}(\mathcal{C}_0(r), 1^{2t}) = 1]$ and $\Pr_r[\mathcal{A}(\mathcal{C}_1(r), 1^{2t}) = 1]$ (where $\mathcal{C}_1$ is defined for k_1) are within $1/(8\ell)$ from each other. It follows that, for this value of k_1, $\mathcal{A}(-, 1^{2t})$ cannot $(1/(4\ell))$-distinguish between $\mathcal{D}_0$ and $\mathcal{D}_1$. Using the PRG of Theorem 2.4, we can compute k_1 in time $\mathsf{poly}(N, t)$.

Once we have computed k_1, we can compute k_2 as the maximum integer $0 \le k_2 \le 2n_2$ such that the estimates of $\Pr_r[\mathcal{A}(\mathcal{C}_1(r), 1^{2t}) = 1]$ and $\Pr_r[\mathcal{A}(\mathcal{C}_2(r), 1^{2t}) = 1]$ (computed using PRG of Theorem 2.4 to within additive error $1/(32\ell)$ each) are within $1/(8\ell)$ from each other (where $\mathcal{C}_2$ is defined for k_2). This implies that, for the computed k_1 and k_2, the algorithm $\mathcal{A}(-, 1^{2t})$ cannot $(1/(4\ell))$-distinguish between $\mathcal{D}_1$ and $\mathcal{D}_2$. We can compute k_2 in the same time $\mathsf{poly}(N, t)$ as for k_1.

Continuing this way, we can compute $k_1, \ldots, k_\ell$, in total time $\mathsf{poly}(N, t)$, so that, by the hybrid argument, $\mathcal{A}(-, 1^{2t})$ cannot $(1/4)$-distinguish between $\mathcal{D}_0$ and $\mathcal{D}_\ell$.

For each k_i, we have that either $k_i < 2n_i$ or $k_i = 2n_i$. If $k_i < 2n_i$, then it must be the case that $\mathcal{A}(-, 1^{2t})$ *can* μ-distinguish between $\mathcal{D}_{i-1}$ and $\mathcal{D}_i$, for $\mathcal{D}_i$ defined with the parameter $k_i' = k_i + 1$, for $\mu \ge 1/(8\ell) - 2/(32\ell) = 1/(16\ell)$. By Lemma 2.7, this implies that, for some polynomial p (dependent on the runtime to sample $\mathcal{D}_{i-1}$ and the runtime of $\mathcal{A}(-, 1^{2t})$),

$$\begin{aligned} \mathsf{K}^{p(t)}(x_i \mid x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) &\le k_i' + O(\log t) \\ &\le k_i + O(\log t). \end{aligned}$$

On the other hand, if $k_i = 2n_i$, then, for any polynomial $p(t) \ge n_i$,

$$\begin{aligned} \mathsf{K}^{p(t)}(x_i \mid x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) &\le n_i + O(1) \\ &\le k_i + O(1), \end{aligned}$$

where the first inequality is by the basic fact that, for some universal constant $c > 0$, and for any $v, w \in \{0,1\}^*$, $\mathsf{K}^t(v \mid w) \le \mathsf{K}^t(v) \le |v| + c$, for any $t \ge |v|$.

It follows that, for each k_i, we have that

$$\mathsf{K}^{p(t)}(x_i \mid x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) \le k_i + O(\log t).$$

Since we can compute all $k_1, \ldots, k_\ell$ in time $\mathsf{poly}(N, t) \le \mathsf{poly}(t)$, we conclude that, for some polynomial q,

$$\mathsf{K}^{q(p(t))}(x_i \mid x_1, \ldots, x_{i-1}) \le \mathsf{K}^{p(t)}(x_i \mid x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) \le k_i + O(\log t). \tag{17}$$

Now we continue as in the proof of Theorem 4.1. Set $s := M - \delta(M, 2t)$. By the definition of a natural property, $\mathcal{A}(-, 1^{2t})$ rejects with probability at least $1/2$ on the uniform distribution

$\mathcal{D}_0$. Hence, by the (1/4)-pseudorandomness of G, $\mathcal{A}(-, 1^{2t})$ rejects with probability at least $1/2 - 1/4 = 1/4$ on the outputs of G. Next, by averaging, there exists a string $z \in \{0,1\}^N$ and strings $u'_1 \in \{0,1\}^{2n_1-k_1}, \ldots, u'_\ell \in \{0,1\}^{2n_\ell - k_\ell}$ such that

$$\mathcal{A}(z \circ G^{x_1}_{k_1}(z_1) \circ u'_1 \circ \cdots \circ G^{x_\ell}_{k_\ell}(z_\ell) \circ u'_\ell, 1^{2t}) = 0,$$

where, as before, each z_i is the prefix of z of appropriate length. By the usefulness s of the natural property (and recalling the definition of $s = M - \delta(M, 2t)$), we get that

$$\mathsf{K}^{2t}\left(z \circ G^{x_1}_{k_1}(z_1) \circ u'_1 \circ \cdots \circ G^{x_\ell}_{k_\ell}(z_\ell) \circ u'_\ell\right) > s = \sum_{i=1}^{\ell} k_i + \sum_{i=1}^{\ell} |u'_i| + |z| - \delta(M, 2t). \tag{18}$$

On the other hand, for every seed $z \in \{0,1\}^N$, and any $u'_1, \ldots, u'_\ell$, we have

$$\mathsf{K}^{2t}\left(z \circ G^{x_1}_{k_1}(z_1) \circ u'_1 \circ \cdots \circ G^{x_\ell}_{k_\ell}(z_\ell) \circ u'_\ell\right) \leq \mathsf{K}^t(x_1, \ldots, x_\ell) + |z| + \sum_{i=1}^{\ell} |u'_i| + c' \cdot \ell \cdot \log(N/\ell), \tag{19}$$

where the last term is to take into account encoding of all n_i's and k_i's, which can be done using at most $c' \cdot \sum_{i=1}^{\ell} \log n_i$ bits, for some constant $c' > 0$; using the AM-GM inequality, we have $\sum_{i=1}^{\ell} \log n_i \leq \ell \cdot \log(N/\ell)$.

Combining (18) and (19), we get that

$$\mathsf{K}^t(x_1, \ldots, x_\ell) \geq \sum_{i=1}^{\ell} k_i - \delta(M, 2t) - \ell \cdot O(\log N). \tag{20}$$

By (17), we then get that

$$\mathsf{K}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{K}^{q(p(t))}(x_i \mid x_1, \ldots, x_{i-1}) - \delta(M, 2t) - \ell \cdot O(\log t),$$

as required. □

4.2 Case of pK^t

Here we prove that a chain rule for (conditional) probabilistic Kolmogorov complexity pK^t follows from a BPP-computable natural property for (conditional) pK^t. The difference from Theorem 4.1 above is that (1) no circuit lower bound for E (derandomization assumption) is needed, and (2) a natural property for (conditional) pK^t can be BPP-computable (rather than P-computable).

Theorem 4.3 (Chain Rule for Conditional pK^t)**.** *Suppose that there is a BPP-computable natural property $\mathcal{A}$ for conditional pK^t on n-bit inputs with usefulness $s(n,t) = n - \delta(n,t)$. Then there exist constants $c_0, c_1 \in \mathbb{N}$ such that for all sufficiently large $x_1, \ldots, x_\ell \in \{0,1\}^*$, for any $\ell \in \mathbb{N}$, for every $y \in \{0,1\}^*$, and for every $t \geq (N + |y|)^{c_0}$, where $N = \sum_{i=1}^{\ell} |x_i|$, we have*

$$\mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{pK}^{t^{c_1}}(x_i \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - \delta(3N, 2t). \tag{21}$$

Proof. We will argue as in the proof of Theorem 4.2. Let $n_i = |x_i|$, for all $1 \le i \le \ell$. For $z \in \{0,1\}^N$, strings $u_i \in \{0,1\}^{2n_i}$, for $1 \le i \le \ell$, and for the parameters $0 \le k_i \le 2n_i$, for $1 \le i \le \ell$, to be determined, define the generator

$$G(z, u_1, \ldots, u_\ell) := z \circ (G_{k_1}^{x_1}(z_1) \circ u_1') \circ \cdots \circ (G_{k_\ell}^{x_\ell}(z_\ell) \circ u_\ell'),$$

where each z_i is the prefix of z of the length required by $G_{k_i}^{x_i}$, and u_i' is the prefix of u_i of length $2n_i - k_i$, for all $1 \le i \le \ell$. The total output length (and input length) of G is $M := 3N$.

Similarly to the proofs of Theorem 4.1 and Theorem 4.2, we want to compute $k_1, \ldots, k_\ell$ satisfying the following:

1. the resulting generator G (1/4)-fools the natural property $\mathcal{A}(-, y, 1^{2t})$ on inputs in $\{0,1\}^M$, and
2. each k_i is (approximately) larger than some $\mathsf{pK}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1})$.

We define hybrid distributions $\mathcal{D}_0, \ldots, \mathcal{D}_\ell$ so that, for each $0 \le i \le \ell$, and the uniform distribution Z over $\{0,1\}^N$,

$$\mathcal{D}_i := Z \circ (G_{k_1}^{x_1}(Z_1) \circ U_1') \circ \cdots \circ (G_{k_i}^{x_i}(Z_i) \circ U_i') \circ \mathcal{U}_{2N - \sum_{1 \le j \le i} 2n_j},$$

where each Z_j is the prefix of Z of appropriate length, and each U_j' is the independent uniform distribution over binary strings of length $2n_j - k_j$, for all $1 \le j \le i$. As shown in the proof of Theorem 4.1, each distribution $\mathcal{D}_i$, for $1 \le i \le \ell$, can be sampled by a randomized algorithm in time $\mathsf{poly}(N)$, given advice $x_1, \ldots, x_i, k_1, \ldots, k_i, N$. Hence, $\mathcal{D}_i$ can be sampled by a circuit $\mathcal{C}_i$ of size $\mathsf{poly}(N)$.

In Theorem 4.2, we used a PRG IW from Theorem 2.4 to compute the required parameters $k_1, \ldots, k_\ell$ deterministically in polynomial time, under the assumption of exponential circuit lower bounds for E. In the present Theorem, we do not assume any derandomization, and so cannot use PRG IW directly. However, we know that, for any polynomial $q(n) \ge n^2$,

$$\mathsf{IW}^{hard} \colon \{0,1\}^{O(\log n)} \to \{0,1\}^n$$

does $1/q(n)$-fool all n-input circuits of size $q(n)$, *provided* we have access to any truth table of a Boolean function $hard \colon \{0,1\}^{O(\log n)} \to \{0,1\}$ of exponential circuit complexity $q_1(q(n))$, for some fixed polynomial q_1. Our idea will be to pick the truth table for $hard$ uniformly at random, and then run PRG IW^{hard}. By counting, a randomly chosen truth table of length $2^{O(\log n)} \le \mathsf{poly}(n)$ will have circuit complexity at least $q_1(q(n))$, with probability at least $1 - 1/\mathsf{poly}(n)$.

Choose $q(t)$ to be (at least) the size of a deterministic circuit simulating $\mathcal{A}(\mathcal{C}_i(r), y, 1^{2t})$ on inputs r of length $\mathsf{poly}(N, |y|, t) \le \mathsf{poly}(t)$ (which includes the randomness to sample from $\mathcal{D}_i$ and the internal randomness of $\mathcal{A}$). Suppose the random choice of $hard$ was successful, i.e., $hard$ has exponential circuit complexity and so $\mathsf{IW}^{hard} \colon \{0,1\}^{O(\log t)} \to \{0,1\}^{\mathsf{poly}(t)}$ does $1/q(t)$-fool all $\mathsf{poly}(t)$-input circuits of size $q(t)$. Then IW^{hard} can be used to compute $k_1, \ldots, k_\ell$ as in the proof of Theorem 4.2. Note that once the truth table of $hard$ is chosen and fixed, the sequence $k_1, \ldots, k_\ell$ is computed deterministically.

As in the proof of Theorem 4.2, the choice of $k_1, \ldots, k_\ell$ (for a "good" truth table of $hard$) will have the properties:

1. for some polynomial τ,

$$\mathsf{pK}^{\tau(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) \leq k_i + O(\log N), \tag{22}$$

where we used the reconstruction property for pK^t of Lemma 2.8 instead of Lemma 2.7, and

2. $\mathcal{A}(-, y, 1^{2t})$ has the same acceptance probability, up to $\pm 1/4$, on $\mathcal{D}_0$ and $\mathcal{D}_\ell$, where the probability is over both the internal randomness of $\mathcal{A}$ and the randomness to sample $\mathcal{D}_i$, for $i = 0$ and $i = \ell$.

By the definition of a BPP-computable natural property $\mathcal{A}$ for conditional pK^t, $\mathcal{A}(-, y, 1^{2t})$ rejects on $\mathcal{D}_0$ with an overall probability at least 0.45. By (1/4)-indistinguishability in Item (2) above, $\mathcal{A}(-, y, 1^{2t})$ rejects on $\mathcal{D}_\ell$ with an overall probability at least $0.45 - 0.25 = 0.2$. On the other hand, if, for every choice of seeds $z, u_1, \ldots, u_\ell$, it were the case that $\mathsf{pK}^{2t}(G(z, u_1, \ldots, u_\ell) \mid y) \leq s(3N, 2t)$, then (by usefulness) $\mathcal{A}(-, y, 1^{2t})$ would accept $\mathcal{D}_\ell$ with an overall probability at least 0.9. Hence, it must be the case for some choice of $z, u_1, \ldots, u_\ell$ that

$$\mathsf{pK}^{2t}(G(z, u_1, \ldots, u_\ell) \mid y) > s(3N, 2t) = 3N - \delta(M, 2t).$$

It follows that

$$\mathsf{pK}^{2t}(G(z, u_1, \ldots, u_\ell) \mid y) > |z| + \sum_{i=1}^{\ell} k_i + \sum_{i=1}^{\ell} |u_i'| - \delta(M, 2t), \tag{23}$$

where u_i' is the prefix of u_i of length $2n_i - k_i$.

On the other hand, for any $z, u_1, \ldots, u_\ell$,

$$\mathsf{pK}^{2t}(G(z, u_1, \ldots, u_\ell) \mid y) \leq \mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) + |z| + \sum_{i=1}^{\ell} |u_i'| + \ell \cdot O(\log N). \tag{24}$$

Combining (23) and (24), we get

$$\mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} k_i - \delta(M, 2t) - \ell \cdot O(\log N).$$

Recalling (22), we get

$$\mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{pK}^{\tau(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) - \delta(M, 2t) - \ell \cdot O(\log N). \tag{25}$$

Next we use an idea of [HN25] to argue that (25) also (approximately) holds in expectation. Remember that (25) holds for $k_1, \ldots, k_\ell$ chosen using IW^{hard} with high probability $1 - 1/\mathsf{poly}(t)$ over the choice of *hard*. On the other hand, when *hard* is not successful (which happens with probability at most $1/\mathsf{poly}(t)$), then, trivially,

$$\sum_{i=1}^{\ell} \mathsf{pK}^{\tau(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1}) \leq \sum_{i=1}^{\ell} n_i + \ell \cdot O(\log N) \leq N^2.$$

We conclude that

$$\begin{aligned}
&\mathbf{Exp}_{hard}\left[\sum_{i=1}^{\ell} \mathsf{pK}^{\tau(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1})\right] \\
&\quad \le \left(\mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) + \delta(M, 2t) + \ell \cdot O(\log N)\right) + (1/\mathsf{poly}(t)) \cdot N^2 \\
&\quad \le \mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) + \delta(M, 2t) + \ell \cdot O(\log N), \qquad (26)
\end{aligned}$$

for $\mathsf{poly}(t) \ge N^2$, where k_i's are computed deterministically, given a particular choice of *hard*. By the linearity of expectation, we get for some polynomial p that

$$\begin{aligned}
&\mathbf{Exp}_{hard}\left[\sum_{i=1}^{\ell} \mathsf{pK}^{\tau(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1})\right] \\
&\quad = \sum_{i=1}^{\ell} \mathbf{Exp}_{hard}\left[\mathsf{pK}^{\tau(t)}(x_i \mid y, x_1, \ldots, x_{i-1}, k_1, \ldots, k_{i-1})\right] \\
&\quad \ge \sum_{i=1}^{\ell} \mathbf{Exp}_{hard}\left[\mathsf{pK}^{p(\tau(t))}(x_i \mid y, x_1, \ldots, x_{i-1}, hard)\right] \\
&\quad =: \sum_{i=1}^{\ell} \mu_i, \qquad (27)
\end{aligned}$$

where the last inequality is because, for each summand $1 \le i \le \ell$, the values $k_1, \ldots, k_{i-1}$ can be computed in time $\mathsf{poly}(t)$ from the truth table of *hard*, using the PRG IW^{hard} of Theorem 2.4.

By Markov's inequality, we have for each $1 \le i \le \ell$ that

$$\Pr_{hard}\left[\mathsf{pK}^{p(\tau(t))}(x_i \mid y, x_1, \ldots, x_{i-1}, hard) > (1 + 1/(2N))\mu_i\right] < (1 + 1/(2N))^{-1} \le 1 - 1/(4N).$$

Hence, with the probability at least $1/(4N)$ over the choice of *hard*,

$$\begin{aligned}
\mathsf{pK}^{p(\tau(t))}(x_i \mid y, x_1, \ldots, x_{i-1}, hard) &\le (1 + 1/(2N))\mu_i \\
&\le \mu_i + (n_i + c)/(2N) \\
&\le \mu_i + 1,
\end{aligned}$$

for a universal constant $c > 0$. For each such string *hard*, we get by the definition of pK^t (see (5)) that, with probability at least $2/3$ over a uniformly random string *rand* of length at most $p(\tau(t))$,

$$\mathsf{K}^{p(\tau(t))}(x_i \mid y, x_1, \ldots, x_{i-1}, hard, rand) \le \mu_i + 1.$$

By putting *hard* and *rand* into a single uniformly random string, we get that with probability at least $1/(6N)$ over $hard \circ rand$ that

$$\mathsf{K}^{p(\tau(t))}(x_i \mid y, x_1, \ldots, x_{i-1}, hard \circ rand) \le \mu_i + O(\log t),$$

where the extra $O(\log t)$ term is to account for encoding the length of *hard*. By the definition of pK^t, this implies that

$$\mathsf{pK}^{p(\tau(t))}_{1/(6N)}(x_i \mid y, x_1, \ldots, x_{i-1}) \le \mu_i + O(\log t).$$

By the success amplification of Lemma 2.1, we conclude that

$$\mathsf{pK}^{O(N\cdot p(\tau(t)))}(x_i \mid y, x_1, \ldots, x_{i-1}) \leq \mu_i + O(\log t). \tag{28}$$

Finally, by (26), (27), and (28), we have for some polynomial p' that

$$\begin{aligned}\mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) + \delta(M, 2t) + \ell \cdot O(\log N) &\geq \sum_{i=1}^{\ell} \mu_i \\ &\geq \sum_{i=1}^{\ell} \mathsf{pK}^{p'(t)}(x_1 \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t),\end{aligned}$$

which implies the required chain rule. □

Theorem 4.4 (Chain Rule for pK^t)**.** *Suppose that there is a* BPP*-computable natural property* $\mathcal{A}$ *for* pK^t *on* n*-bit inputs with usefulness* $s(n,t) = n - \delta(n,t)$*. Then there exist constants* $c_0, c_1 \in \mathbb{N}$ *such that for all sufficiently large* $x_1, \ldots, x_\ell \in \{0,1\}^*$*, for any* $\ell \in \mathbb{N}$*, and for every* $t \geq (N)^{c_0}$*, where* $N = \sum_{i=1}^{\ell} |x_i|$*, we have*

$$\mathsf{pK}^t(x_1, \ldots, x_\ell) \geq \sum_{i=1}^{\ell} \mathsf{pK}^{t^{c_1}}(x_i \mid x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - \delta(3N, 2t). \tag{29}$$

Proof. The proof is identical to that of Theorem 4.3, with $y = \epsilon$ everywhere. □

5 Natural Properties from Chain Rules

5.1 Case of K^t

Theorem 5.1 (Natural Property for Conditional K^t)**.** *Assume that, for every constant* $C > 1$*, the Chain Rule for conditional* K^t *as in Eq. (9) of Theorem 4.1 holds for* $\delta(N,t) \leq N/C$*. Then, for every constant* $D \geq 1$*, there is a* P*-computable natural property* $\mathcal{A}(x, y, 1^t)$ *for conditional* K^t *on* n*-bit inputs* x*, conditioned on* y*, with usefulness* $s(n,t) = n - n/D$*.*

Proof. Given $x \in \{0,1\}^n$, partition x into $1 \leq \ell \leq n$ strings $x_1, \ldots, x_\ell$ of length n/ℓ each, for some $\ell = n/(c \log t)$, for a constant $c > 0$ to be chosen later. Note that

$$\mathsf{K}^{2t}(x_1, \ldots, x_\ell \mid y) \leq \mathsf{K}^t(x \mid y) + O(\log n).$$

On the other hand, by the assumed Chain Rule, we have, for sufficiently large $t \geq \mathsf{poly}(n, |y|)$, that

$$\mathsf{K}^{2t}(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{K}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - \delta(3n, 2t).$$

Consider the case that $\mathsf{K}^t(x \mid y) \leq s(n,t)$. Then

$$\sum_{i=1}^{\ell} \mathsf{K}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) \leq s(n,t) + \ell \cdot O(\log t) + \delta(3n, 2t).$$

Hence, by averaging, there is some $1 \leq i \leq \ell$ such that

$$\mathsf{K}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) \leq \left(s(n,t) + \ell \cdot O(\log t) + \delta(3n, 2t)\right)/\ell \;=: S. \tag{30}$$

Note that, by brute force, for each $1 \leq i \leq \ell$, we can compute $\mathsf{K}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1})$ in time $2^{n/\ell} \cdot \mathsf{poly}(t)$, which is at most $\mathsf{poly}(t)$ for $\ell = n/(c \log t)$, for a constant $c > 0$ to be determined.

Consider the following decision algorithm $\mathcal{A}(x, y, 1^t)$:

> On input $x \in \{0,1\}^n$, $y \in \{0,1\}^*$, and 1^t, output 1 iff there is some $1 \leq i \leq \ell$ such that Eq. (30) is satisfied.

Clearly, $\mathcal{A}$ is a polynomial-time algorithm. By definition, it accepts all pairs of strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^*$ with $\mathsf{K}^t(x \mid y) \leq s(n,t)$. We need to argue that, for every $y \in \{0,1\}^*$, $\mathcal{A}(z, y, 1^t)$ will reject at least $1/2$ of random input strings $z \in \{0,1\}^n$, for an appropriately chosen $s(n,t)$.

Suppose that $s(n,t)$ is such that

$$S \leq (n/\ell) - 2\log \ell, \tag{31}$$

for S defined in (30). By a simple counting argument, we have, for any fixed string w,

$$\mathbf{Pr}_{a \in \{0,1\}^{n/\ell}}\left[\mathsf{K}(a \mid w) \leq (n/\ell) - 2\log \ell\right] \leq 2/\ell^2.$$

Imagine picking a uniformly random $z \in \{0,1\}^n$ in stages, where we first pick $z_1 \in \{0,1\}^{n/\ell}$, then $z_2 \in \{0,1\}^{n/\ell}$, and so on until $z_\ell \in \{0,1\}^{n/\ell}$, and finally output $z = z_1 \circ z_2 \circ \cdots \circ z_\ell$. By the union bound, the probability that at least one of $z_1, \ldots, z_\ell \in \{0,1\}^{n/\ell}$ has

$$\mathsf{K}(z_i \mid y, z_1, \ldots, z_{i-1}) \leq (n/\ell) - 2\log \ell$$

is at most $\ell \cdot (2/\ell^2) \leq 2/\ell$. It follows that for $s(n,t)$ satisfying (31) above, we have

$$\mathbf{Pr}_{z \in \{0,1\}^n}[\mathcal{A}(z, y, 1^t) = 1] \leq 2/\ell,$$

which is at most $1/2$ for $\ell \geq 4$.

It remains to see which values $s(n,t)$ will satisfy (31). We get that $s(n,t)$ must satisfy the following:

$$s(n,t) \leq n - 2\ell \cdot \log \ell - \ell \cdot O(\log t) - \delta(3n, 2t).$$

We will set $s(n,t)$ equal to the expression above. Recalling that $\ell = n/(c \log t)$, we get for any given constant $d > 0$ (by choosing c a large enough constant) that

$$\begin{aligned} s(n,t) &= n - (2n \log \ell)/(c \log t) - n \cdot O(\log t)/(c \log t) - \delta(3n, 2t) \\ &\geq n - n/d - \delta(3n, 2t) \\ &\geq n - n/d - (3n)/C \\ &= n - n/D, \end{aligned}$$

for $C = 3d$ and $d = 2D$. The theorem follows. □

Theorem 5.2 (Natural Property for K^t)**.** *Assume that, for every constant $C > 1$, the Chain Rule for K^t as in Eq. (16) of Theorem 4.2 holds for $\delta(N,t) \leq N/C$. Then, for every constant $D \geq 1$, there is a P-computable natural property $\mathcal{A}$ for K^t on n-bit inputs with usefulness $s(n,t) = n - n/D$.*

Proof. The proof is analogous to that of Theorem 5.1 above, with $y = \epsilon$ everywhere. □

5.2 Case of pK^t

Theorem 5.3 (Natural Property for Conditional pK^t)**.** *Assume that, for every constant $C > 1$, the Chain Rule for conditional pK^t as in Eq. (21) of Theorem 4.3 holds for $\delta(N,t) \leq N/C$. Then, for every constant $D \geq 1$, there is a BPP-computable natural property $\mathcal{A}$ for conditional pK^t on n-bit inputs with usefulness $s(n,t) = n - n/D$.*

Proof. The proof is similar to that of Theorem 5.1, adapted to the case of conditional pK^t. Given $x \in \{0,1\}^n$, partition x into $1 \leq \ell \leq n$ strings $x_1, \ldots, x_\ell$ of length n/ℓ each, for some ℓ to be chosen later. Note that, for every $y \in \{0,1\}^*$,

$$\mathsf{pK}^{2t}(x_1, \ldots, x_\ell \mid y) \leq \mathsf{pK}^t(x \mid y) + O(\log n).$$

On the other hand, by the assumed Chain Rule, we have, for sufficiently large $t \geq \mathsf{poly}(n, |y|)$, that

$$\mathsf{pK}^{2t}(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{pK}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log n) - \delta(3n, 2t).$$

Consider the case that $\mathsf{pK}^t(x \mid y) \leq s(n,t)$, and so

$$\sum_{i=1}^{\ell} \mathsf{pK}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) \leq s(n,t) + \ell \cdot O(\log n) + \delta(3n, 2t).$$

By averaging, there is some $1 \leq i \leq \ell$ such that

$$\mathsf{pK}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) \leq \left(s(n,t) + \ell \cdot O(\log n) + \delta(3n, 2t)\right)/\ell \;=: S. \tag{32}$$

By brute force, for each $1 \leq i \leq \ell$, we can "approximate" $\mathsf{pK}^{\mathsf{poly}(t)}(x_i \mid y, x_1, \ldots, x_{i-1})$ (by random sampling) in randomized time $2^{n/\ell} \cdot \mathsf{poly}(t)$, which is at most $\mathsf{poly}(t)$ for $\ell = n/(c \log t)$, for some sufficiently large constant $c > 0$ to be determined. We provide the details next.

Consider the following randomized decision algorithm $\mathcal{A}(x, y, 1^t)$:

> On input $x \in \{0,1\}^n$, $y \in \{0,1\}^*$, and 1^t, output 1 iff there is some $1 \leq i \leq \ell$ such that Eq. (32) is "approximately satisfied". More precisely, output 1 if, for some $1 \leq i \leq \ell$, for each of at least 0.6 fraction of randomly sampled n strings $r \in \{0,1\}^{t'}$, for $t' = \mathsf{poly}(t)$, there is a string $w_r \in \{0,1\}^{\leq S}$ such that $U(w_r, r, y, x_1, \ldots, x_{i-1})$ outputs x_i within t' steps.

Clearly, $\mathcal{A}$ is a randomized polynomial-time algorithm. It accepts every string $x \in \{0,1\}^n$ with $\mathsf{K}^t(x \mid y) \leq s(n,t)$, with high probability over its internal randomness (which can be argued by the Chernoff bounds). We need to argue that, for every y, $\mathcal{A}(z, y, 1^t)$ will reject at least $1/2$ of random input strings $z \in \{0,1\}^n$, with high probability over its internal randomness, for an appropriately chosen $s(n,t)$.

Suppose that $s(n,t)$ is such that

$$S \leq (n/\ell) - 2\log \ell, \tag{33}$$

for S defined in (32). By Lemma 2.2, we have, for any fixed string w,

$$\mathbf{Pr}_{a \in \{0,1\}^{n/\ell}}\left[\mathsf{pK}^{t'}_{0.55}(a \mid w) \leq (n/\ell) - 2\log \ell\right] \leq 4/\ell^2.$$

Imagine picking a uniformly random $z \in \{0,1\}^n$ in stages, where we first pick $z_1 \in \{0,1\}^{n/\ell}$, then $z_2 \in \{0,1\}^{n/\ell}$, and so on until $z_\ell \in \{0,1\}^{n/\ell}$, and finally output $z = z_1 \circ z_2 \circ \cdots \circ z_\ell$. By the union bound, we have with probability at least $1 - 4/\ell \geq 1/2$ over random $z = z_1 \dots z_\ell \in \{0,1\}^n$ that, for every one of $z_1, \dots, z_\ell \in \{0,1\}^{n/\ell}$, it holds that

$$\mathsf{pK}^{t'}_{0.55}(z_i \mid y, z_1, \dots, z_{i-1}) > (n/\ell) - 2\log \ell.$$

Consider any such $z = z_1 \dots z_\ell \in \{0,1\}^n$. For each $1 \leq i \leq \ell$, there are fewer than 0.55 fraction of "good" random strings $r \in \{0,1\}^{t'}$ for which there is a description $w_r \in \{0,1\}^{\leq S}$ such that $U(w_r, r, y, z_1, \dots, z_{i-1})$ outputs z_i within t' steps. By the Chernoff bounds, algorithm $\mathcal{A}$ will see at least 0.6 fraction of "good" random strings r in its sample of size n with probability at most $2^{-\Omega(n)}$. By the union bound, the probability that $\mathcal{A}$ will accept for at least one $1 \leq i \leq \ell$ is at most $\ell \cdot 2^{-\Omega(n)}$. Hence, the probability over the internal randomness of $\mathcal{A}$ that $\mathcal{A}(z, y, 1^t) = 0$ is at least $1 - \ell \cdot 2^{-\Omega(n)} \geq 0.9$, for all sufficiently large n.

It remains to see which values $s(n,t)$ will satisfy (33). We get that $s(n,t)$ must satisfy the following:

$$s(n,t) \leq n - 2\ell \cdot \log \ell - \ell \cdot O(\log n) - \delta(3n, 2t).$$

Recalling that $\ell = n/(c \log t)$, we set, for any given constant $D > 0$ (by choosing c large enough),

$$\begin{aligned} s(n,t) &= n - (2n \log \ell)/(c \log t) - n \cdot O(\log n)/(c \log t) - \delta(3n, 2t) \\ &\geq n - n/(2D) - \delta(3n, 2t) \\ &= n - n/(2D) - (3n)/C \\ &\leq n - n/D, \end{aligned}$$

for $C = 6D$. □

Theorem 5.4 (Natural Property for pK^t)**.** *Assume that, for every constant $C > 1$, the Chain Rule for pK^t as in Eq. (29) of Theorem 4.4 holds for $\delta(N,t) \leq N/C$. Then, for every constant $D \geq 1$, there is a BPP-computable natural property $\mathcal{A}$ for pK^t on n-bit inputs with usefulness $s(n,t) = n - n/D$.*

Proof. The proof is identical to that of Theorem 5.3, with $y = \epsilon$. □

6 Circuit Lower Bounds from Chain Rules

6.1 Case of K^t

Here we get strongly exponential, almost everywhere, circuit lower bounds from a Chain Rule for K^t, using an idea from [Per07]; see also [All+06] for a similar argument in the setting of *time-unbounded* Kolmogorov complexity.[9] We will need the following lemma.

Lemma 6.1. *Let $x \in \{0,1\}^N$, for $N = 2^n$, be such that, for for $t \geq N^3$, we have*

$$\mathsf{K}^t(x) \geq (0.9) \cdot N. \tag{34}$$

[9] More precisely, [All+06] re-prove a result of [Buh+05] that one can efficiently deterministically construct a string of high Kolmogorov complexity, given oracle access to the set of Kolmogorov-random strings. The proof relies on the Symmetry of Information for time-unbounded Kolmogorov complexity.

Then x, viewed as a truth table of an n-variate Boolean function, requires circuit size at least $2^n/(c \cdot n)$, for a sufficiently large constant $c \in \mathbb{N}$.

Proof. Suppose towards a contradiction that x can be computed by a circuit of size $\sigma < N/(c \cdot n)$. Such a circuit can be described using

$$\sigma' = \sigma \cdot d \cdot \log \sigma < \frac{N \cdot d \cdot n}{c \cdot n}$$

bits, for some constant $d \in \mathbb{N}$. We can make $\sigma' < N/2$ by choosing $c = 2d$. Given the description of this circuit, one can compute the entire string x bit by bit, by evaluating the circuit on all possible n-bit inputs. This can be done in time $2^n \cdot (\sigma')^2 \leq N^3$. Hence, for $t \geq N^3$, we have that $\mathsf{K}^t(x) < N/2$, contradicting (34). □

Theorem 6.2. *Assume that, for any constant $D > 1$, the Chain Rule for K^t as in Eq. (16) of Theorem 4.2 holds for $\delta(N, t) \leq N/D$. Then*

$$\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[o(2^n/n)].$$

Proof. Let $N = 2^n$, $m = c \cdot n$ for some constant $c > 0$ to be determined, and $\ell = N/m$. Let $N^{c_0} \leq t = N^{c_1}$, for some constant $c_1 \geq 3$. Let $t' = \mathsf{poly}(t)$ be the polynomial time bound on the right-hand side of the Chain Rule for K^t.

By a brute-force algorithm, we construct $A_1 \in \{0,1\}^m$ so that

$$\mathsf{K}^{t'}(A_1) \geq m. \tag{35}$$

Note that by a counting argument, such a string A_1 must exist. We find the lexicographically first such string by enumerating all m-bit strings $a \in \{0,1\}^m$, and checking for each if some $w \in \{0,1\}^{\leq m}$ exists such that $U(w)$ outputs a within t' steps; if yes, we skip over to the next $a \in \{0,1\}^m$. It takes time $2^{O(n)}$ to find A_1.

Similarly, given the found string A_1, we next find (again by brute force, in time $2^{O(n)}$) a string $A_2 \in \{0,1\}^m$ such that

$$\mathsf{K}^{t'}(A_2 \mid A_1) \geq m.$$

Such a string must exist by a counting argument. To find it, we proceed similarly to the case of A_1, but now simulating $U(w, A_1)$ for t' steps.

We continue this way to define $A_1, A_2, \ldots, A_\ell$ so that, for all $1 \leq i \leq \ell$,

$$\mathsf{K}^{t'}(A_i \mid A_1, \ldots, A_{i-1}) \geq m. \tag{36}$$

Next, by the assumed Chain Rule, we get from Eq. (36) that

$$\begin{aligned}
\mathsf{K}^t(A_1, \ldots, A_\ell) &\geq \sum_{i=1}^{\ell} \mathsf{K}^{t'}(A_i \mid A_1, \ldots, A_{i-1}) - \ell \cdot O(\log t) - \delta(2(m^2 + N), 2t) \\
&\geq \ell \cdot m - \ell \cdot O(c_1 n) - 4N/D \\
&= \frac{2^n}{cn} \cdot (cn - O(c_1 n) - 4cn/D)\,.
\end{aligned} \tag{37}$$

Choose the constants c and D large enough so that $(cn - O(c_1 n) - 4cn/D) \geq (0.9) \cdot cn$. Then $B = A_1 \ldots A_\ell \in \{0,1\}^{2^n}$ requires circuit size $\Omega(2^n/n)$ by Lemma 6.1.

Finally, observe that each A_i, for $1 \leq i \leq \ell$, is computable in time $2^{O(n)}$. Hence, the truth table $B \in \{0,1\}^{2^n}$ is computable in time $2^{O(n)}$. Collecting such B's over all input lengths n, we get a language $L \in \mathsf{E}$ that, almost everywhere, requires circuit size $\Omega(2^n/n)$. □

Remark 6.3. *Under the assumption that a variant of the two-string Conditional Chain Rule for* K^t *holds, Perifel [Per07] proved that* $\mathsf{EXP} \not\subset \mathsf{P/poly}$. *One reason that [Per07] obtained only a superpolynomial rather than exponential circuit lower bound for* EXP *is that it used this two-string Conditional Chain Rule to derive an* ℓ*-string Chain Rule for* K^t*, for a super-constant* ℓ*, suffering significant blowups in the time bounds for the* K^t *complexities involved. This precluded the use of sufficiently large values of* ℓ *in the derived Chain Rule. In contrast, we start with an optimal* ℓ*-string Chain Rule for* K^t *for an arbitrary* $\ell \in \mathbb{N}$*, and so we can choose a sufficiently large value of* ℓ *to get exponential circuit lower bounds.*

6.2 Case of pK^t

We get circuit lower bounds for randomized complexity classes from the assumption that a Chain Rule for pK^t holds.

Lemma 6.4. *A* BPP*-computable natural property for* K^{n^3} *on* n*-bit inputs with usefulness* $s(n) \geq (0.9) \cdot n$ *yields a* BPP*-computable natural property for circuit size on truth tables of length* $N = 2^n$ *with usefulness* $s(N) \geq N^{\varepsilon}$*, for some* $0 < \varepsilon < 1$.

Proof. Let $\mathcal{A}$ be a BPP-computable natural property for K^{n^3} on n-bit strings with usefulness $s(n) \geq (0.9) \cdot n$. Define $\mathcal{A}' \colon \{0,1\}^N \to \{0,1\}$ as follows:

On input $x \in \{0,1\}^N$, $\mathcal{A}'(x) = \mathcal{A}(x, 1^{N^3})$.

By definition, $\mathcal{A}'$ is a randomized $\mathsf{poly}(N)$-time algorithm. For at least $1/2$ of random strings $w \in \{0,1\}^N$, $\mathcal{A}'(w)$ rejects with probability at least 0.9 because so does $\mathcal{A}(w)$. For $x \in \{0,1\}^N$ with $\mathsf{size}(x) \leq 2^{(0.9) \cdot n}$, we get by Lemma 6.1 that $\mathsf{K}^{N^3}(x) < (0.9) \cdot N$, for all sufficiently large $n \in \mathbb{N}$. Hence, $\mathcal{A}(x, 1^{N^3})$ accepts such an x with probability at least 0.9, and therefore so does $\mathcal{A}'(x)$. □

Theorem 6.5. *Assume the Chain Rule for* pK^t *as in Eq. (29) of Theorem 4.4, for* $\delta(n,t) \leq n/c$*, for a sufficiently large* $c > 1$. *Then*

- $\mathsf{BPEXP} \not\subset \mathsf{P/poly}$*, and,*
- $\mathsf{BPP}/1 \not\subset \mathsf{SIZE}[n^k]$*, and* $\mathsf{promise\text{-}BPP} \not\subset \mathsf{SIZE}[n^k]$ *for any fixed* $k \geq 1$.

Proof. By our assumption and by Theorem 5.4, we get that there is a BPP-computable natural property $\mathcal{A}$ for K^{n^3} on n-bit strings with usefulness $s(n) = (0.9) \cdot n$. By Lemma 6.4 and Corollary 2.11, the theorem follows. □

7 Proofs of Equivalences for Chain Rules

Here we prove the main equivalences stated in the introduction.

7.1 Case of K^t

Consider the following assumptions:

CKT-LB: $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$,

PROP-cK^t: for any constant $C > 1$, there is a P-computable natural property $\mathcal{P}$ for conditional K^t with usefulness $s(n,t) = n - n/C$,

GAP-$\mathsf{McK^tP}$: for any constant $C' > 1$, there is a polynomial τ such that $\mathsf{Gap}_{\tau,\delta'}\mathsf{McK^tP} \in \mathsf{promise\text{-}P}$, where $\delta'(n,t) = n/C'$, and

CHAIN-cK^t: for any constant $D > 1$, for some polynomial p, for any $\ell \in \mathbb{N}$, for any $x_1, \ldots, x_\ell \in \{0,1\}^*$ with $N = \sum_{i=1}^{\ell} |x_i|$, and any $y \in \{0,1\}^*$, and all sufficiently large t (at least polynomial in $N + |y|$), we have

$$\mathsf{K}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{K}^{p(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - N/D.$$

Theorem 7.1 (Equivalence for Conditional K^t)**.** *We have that*

$$\text{CKT-LB} \;\&\; \text{PROP-}\mathsf{cK}^t \;\Leftrightarrow\; \text{CHAIN-}\mathsf{cK}^t \;\Leftrightarrow\; \text{CKT-LB} \;\&\; \text{GAP-}\mathsf{McK^tP}.$$

Proof. The direction CKT-LB & PROP-$\mathsf{cK}^t \Rightarrow$ CHAIN-cK^t follows by Theorem 4.1. The direction CHAIN-$\mathsf{cK}^t \Rightarrow$ CKT-LB follows by Theorem 6.2. The direction CHAIN-$\mathsf{cK}^t \Rightarrow$ PROP-cK^t follows by Theorem 5.1. Finally, the equivalence CKT-LB & PROP-$\mathsf{cK}^t \;\Leftrightarrow\;$ CKT-LB & GAP-$\mathsf{McK^tP}$ is by Corollary 3.6. □

We also have the version of Theorem 7.1 for unconditional K^t. Consider the following assumptions:

PROP-K^t: for any constant $C > 1$, there is a P-computable natural property $\mathcal{P}$ for K^t with usefulness $s(n,t) = n - n/C$,

GAP-$\mathsf{MK^tP}$: for any constant $C' > 1$, there is a polynomial τ such that $\mathsf{Gap}_{\tau,\delta'}\mathsf{MK^tP} \in \mathsf{promise\text{-}P}$, where $\delta'(n,t) = n/C'$, and

CHAIN-K^t: for any constant $D > 1$, for some polynomial p, for any $\ell \in \mathbb{N}$, for any $x_1, \ldots, x_\ell \in \{0,1\}^*$ with $N = \sum_{i=1}^{\ell} |x_i|$, and all sufficiently large t (at least polynomial in N), we have

$$\mathsf{K}^t(x_1, \ldots, x_\ell) \geq \sum_{i=1}^{\ell} \mathsf{K}^{p(t)}(x_i \mid x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - N/D.$$

Theorem 7.2 (Equivalence for K^t)**.** *We have that*

$$\text{CKT-LB} \;\&\; \text{PROP-}\mathsf{K}^t \;\Leftrightarrow\; \text{CHAIN-}\mathsf{K}^t \;\Leftrightarrow\; \text{CKT-LB} \;\&\; \text{GAP-}\mathsf{MK^tP}.$$

Proof. The direction CKT-LB & PROP-$\mathsf{K}^t \Rightarrow$ CHAIN-K^t follows by Theorem 4.2. That CHAIN-$\mathsf{K}^t \Rightarrow$ CKT-LB follows by Theorem 6.2. The direction CHAIN-$\mathsf{K}^t \Rightarrow$ PROP-K^t follows by Theorem 5.2. Finally, the equivalence CKT-LB & PROP-$\mathsf{K}^t \;\Leftrightarrow\;$ CKT-LB & GAP-$\mathsf{MK^tP}$ is by Corollary 3.6. □

7.2 Case of pK^t

Consider the following assumptions:

PROP-cpK^t: For any constant $C > 1$, there is a BPP-computable natural property $\mathcal{P}$ for conditional pK^t with usefulness $s(n,t) = n - n/C$.

GAP-$\mathsf{McpK^tP}$: For any constant $C' > 1$, there is a polynomial τ such that $\mathsf{Gap}_{\tau,\delta}\mathsf{McpK^tP} \in \mathsf{promise\text{-}BPP}$, for $\delta(n,t) \leq n/C'$.

CHAIN-cpK^t: For any constant $D > 1$, for some polynomial p, for any $\ell \in \mathbb{N}$, for any $x_1, \ldots, x_\ell \in \{0,1\}^*$ and any $y \in \{0,1\}^*$, and all sufficiently large t (at least polynomial in the total length of all strings), we have

$$\mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{pK}^{p(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - N/D,$$

where $N = \sum_{i=1}^{\ell} |x_i|$.

Theorem 7.3 (Equivalence for Conditional pK^t)**.** *We have that*

$$\text{PROP-}\mathsf{cpK}^t \Leftrightarrow \text{CHAIN-}\mathsf{cpK}^t \Leftrightarrow \text{GAP-}\mathsf{McpK^tP}.$$

Proof. The direction PROP-$\mathsf{cpK}^t \Rightarrow$ CHAIN-cpK^t is by Theorem 4.3. The direction CHAIN-$\mathsf{cpK}^t \Rightarrow$ PROP-cpK^t is by Theorem 5.3. The equivalence PROP-$\mathsf{cpK}^t \Leftrightarrow$ GAP-$\mathsf{McpK^tP}$ is by Corollary 3.8. □

For unconditional pK^t, consider the following assumptions:

PROP-pK^t: For any constant $C > 1$, there is a BPP-computable natural property $\mathcal{P}$ for pK^t with usefulness $s(n,t) = n - n/C$.

GAP-$\mathsf{MpK^tP}$: For any constant $C' > 1$, there is a polynomial τ such that $\mathsf{Gap}_{\tau,\delta}\mathsf{MpK^tP} \in \mathsf{promise\text{-}BPP}$, for $\delta(n,t) \leq n/C'$.

CHAIN-pK^t: For any constant $D > 1$, for some polynomial p, for any $\ell \in \mathbb{N}$, for any $x_1, \ldots, x_\ell \in \{0,1\}^*$, and all sufficiently large t (at least polynomial in the total length of all strings), we have

$$\mathsf{pK}^t(x_1, \ldots, x_\ell) \geq \sum_{i=1}^{\ell} \mathsf{pK}^{p(t)}(x_i \mid x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - N/D,$$

where $N = \sum_{i=1}^{\ell} |x_i|$.

Theorem 7.4 (Equivalence for pK^t)**.** *We have that*

$$\text{PROP-}\mathsf{pK}^t \Leftrightarrow \text{CHAIN-}\mathsf{pK}^t \Leftrightarrow \text{GAP-}\mathsf{MpK^tP}.$$

Proof. The direction PROP-$\mathsf{pK}^t \Rightarrow$ CHAIN-cpK^t is by Theorem 4.4. The direction CHAIN-$\mathsf{pK}^t \Rightarrow$ PROP-pK^t is by Theorem 5.4. The equivalence PROP-$\mathsf{pK}^t \Leftrightarrow$ GAP-$\mathsf{MpK^tP}$ is by Corollary 3.8. □

7.3 If a Gap-Version of (Conditional) K^t Were NP-Hard

In [HIR23], it is shown under cryptographic assumptions that a certain gap-version of computing conditional K^t is NP-hard.

Theorem 7.5 ([HIR23])**.** *Assume subexponentially-secure witness encryption for* NP*. Then the following promise-problem is* NP*-hard under randomized polynomial-time (black-box) reductions: For* $x, y \in \{0,1\}^*$*, with* $|x| = n$ *and* $|y| = \mathsf{poly}(n)$*, and some polynomial* t*, distinguish between*

- $\Pi_{\text{YES}} = \{(x,y) \mid \mathsf{K}^{t(n)}(x \mid y) \leq \sqrt{n}\}$*, and*
- $\Pi_{\text{NO}} = \{(x,y) \mid \mathsf{K}^{2^{n^2}}(x \mid y) \geq n - O(1)\}$.

Note that if, for $\delta(n,t) = n/C$, for some large constant $c > 1$, and for some polynomial τ, it would be the case that $\mathsf{Gap}_{\tau,\delta}\mathsf{McK^tP} \in \mathsf{promise\text{-}BPP}$, then we would be able to solve the promise-problem from Theorem 7.5 above in $\mathsf{promise\text{-}BPP}$ as well. Thus, it is plausible that, for any large $c > 1$ and any polynomial τ, the problem $\mathsf{Gap}_{\tau,n/c}\mathsf{McK^tP}$ may in fact be NP-hard (under randomized polynomial-time reductions). This would dramatically simplify the equivalence for the Chain Rule for conditional K^t in Theorem 7.1, as we show next.

Theorem 7.6. *Suppose that, for any large* $c > 1$ *and any polynomial* τ*, the problem* $\mathsf{Gap}_{\tau,n/c}\mathsf{McK^tP}$ *is* NP*-hard. Then the following are equivalent:*

- $\mathsf{NP} = \mathsf{P}$*, and*
- *for any constant* $D > 1$*, for some polynomial* p*, for any* $\ell \in \mathbb{N}$*, for any* $x_1, \ldots, x_\ell \in \{0,1\}^*$ *with* $N = \sum_{i=1}^{\ell} |x_i|$*, and any* $y \in \{0,1\}^*$*, and all sufficiently large* t *(at least polynomial in* $N + |y|$*), we have*

$$\mathsf{K}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{K}^{p(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - N/D.$$

Proof. In the forward direction, $\mathsf{NP} = \mathsf{P}$ implies $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$ [Kan82], as well as a P-computable natural property for conditional K^t with usefulness $s(n,t) = n - o(n)$. Hence we can apply Theorem 7.1 to get the Chain Rule for conditional K^t, as required.

In the reverse direction, by Theorem 7.1, we get from the assumed Chain Rule for conditional K^t, for any constant $C' > 1$ and some polynomial τ, a $\mathsf{promise\text{-}P}$ algorithm for $\mathsf{Gap}_{\tau,n/C'}\mathsf{McK^tP}$. The latter problem was assumed to be NP-hard (under randomized reductions). Hence, we conclude that $\mathsf{NP} \subseteq \mathsf{BPP}$. Since the Chain Rule also implies that $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$, we get that $\mathsf{BPP} = \mathsf{P}$ (by Theorem 2.4). □

Similarly, for the case of $\mathsf{Gap}_{\tau,\Omega(n)}\mathsf{MK}^t\mathsf{P}$ problem we consider, there is some evidence that this problem may be NP-complete.

Theorem 7.7 ([Ila23])**.** *Let* $p(t) \geq t$ *be any polynomial. There is a deterministic polynomial-time oracle algorithm* $A^{(\cdot)}$ *such that, with probability one over a random oracle* $\mathcal{O}$*,* $A^{\mathcal{O}}$ *is a many-one reduction from standard, non-relativized* SAT *to the following relativized promise problem over inputs* $x \in \{0,1\}^n$*:*

- $\Pi_{\text{YES}} = \{x \in \{0,1\}^n \mid \mathsf{K}^{t,\mathcal{O}}(x) \leq \theta(n)\}$,
- $\Pi_{\text{NO}} = \{x \in \{0,1\}^n \mid \mathsf{K}^{p(t),\mathcal{O}}(x) \geq \theta(n) + \varepsilon n\}$,

where $\theta\colon \mathbb{N} \to \mathbb{N}$ is a fixed function, $\varepsilon > 0$ is a fixed constant, and t is some polynomial in n.

Note that the promise problem in Theorem 7.7 *without* an oracle $\mathcal{O}$ is a special case of $\mathsf{Gap}_{\mathsf{poly},\Omega(n)}\mathsf{MK}^t\mathsf{P}$ that we consider in this paper (with the threshold parameter $s = \theta(|x|)$ set to be a fixed function of the input length). Thus, one can view Theorem 7.7 as evidence that $\mathsf{Gap}_{\mathsf{poly},\Omega(n)}\mathsf{MK}^t\mathsf{P}$ may be NP-complete.

We have the following analogue of Theorem 7.6.

Theorem 7.8. *Suppose that, for any large $c > 1$ and any polynomial τ, the problem $\mathsf{Gap}_{\tau,n/c}\mathsf{MK}^t\mathsf{P}$ is NP-hard. Then the following are equivalent:*

- $\mathsf{NP} = \mathsf{P}$, *and*
- *for any constant $D > 1$, for some polynomial p, for any $\ell \in \mathbb{N}$, for any $x_1, \ldots, x_\ell \in \{0,1\}^*$ with $N = \sum_{i=1}^{\ell} |x_i|$, and all sufficiently large t (at least polynomial in N), we have*

$$\mathsf{K}^t(x_1, \ldots, x_\ell) \geq \sum_{i=1}^{\ell} \mathsf{K}^{p(t)}(x_i \mid x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - N/D.$$

Proof. The proof is similar to that of Theorem 7.6 above, but using Theorem 7.2 instead of Theorem 7.1. □

Of course, we also get similar equivalences for pK^t, assuming that the gap-version of approximating (conditional) pK^t is NP-hard. For example, we have the following analogue of Theorem 7.6.

Theorem 7.9. *Suppose that, for any large $c > 1$ and any polynomial τ, the problem $\mathsf{Gap}_{\tau,n/c}\mathsf{McpK}^{\mathsf{t}}\mathsf{P}$ is NP-hard (under randomized reductions). Then the following are equivalent:*

- $\mathsf{NP} \subseteq \mathsf{BPP}$, *and*
- *for any constant $D > 1$, for some polynomial p, for any $\ell \in \mathbb{N}$, for any $x_1, \ldots, x_\ell \in \{0,1\}^*$ with $N = \sum_{i=1}^{\ell} |x_i|$, and any $y \in \{0,1\}^*$, and all sufficiently large t (at least polynomial in $N + |y|$), we have*

$$\mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{pK}^{p(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - N/D.$$

Proof. The reverse direction is immediate by Theorem 7.3. For the forward direction, $\mathsf{NP} \subseteq \mathsf{BPP}$ implies that $\mathsf{promise\text{-}AM} \subseteq \mathsf{promise\text{-}BPP}$ (via a standard argument). Hence, we get that $\mathsf{Gap}_{\tau,n/c}\mathsf{McpK}^{\mathsf{t}}\mathsf{P} \in \mathsf{promise\text{-}BPP}$. Then one applies Theorem 7.3. □

8 Applications

8.1 Sparse Natural Property for Conditional pK^t

Definition 8.1 (Sparse natural property for conditional pK^t)**.** *A natural property for conditional pK^t with usefulness $s(n,t)$ is a predicate $\mathcal{P}\colon \{0,1\}^* \times \{0,1\}^* \times 1^* \to \{0,1\}$ such that, for some polynomial p, we have for all large $n, m \in \mathbb{N}$ and $t \geq p(n,m)$ that*

1. *for all $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, if $\mathsf{pK}^t(x \mid y) \leq s(n,t)$, then $\mathcal{P}(x,y,1^t) = 1$, and*
2. *for every $y \in \{0,1\}^m$, $\mathbf{Pr}_{x\in\{0,1\}^n}\left[\mathcal{P}(x,y,1^t) = 0\right] \geq 1/2$.*

When $n \leq \log t$, we call such a property sparse*.*

Lemma 8.2. *For every $s(n,t) \leq n-3$, a natural property for conditional $\mathsf{pK}^t(x \mid y)$, for $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, with usefulness $s(n,t)$ is $\mathsf{BPTIME}(\mathsf{poly}(2^n, m, t))$-computable. Hence, a sparse natural property for conditional pK^t with usefulness $s(n,t)$ is BPP-computable.*

Proof. For given $x, y \in \{0,1\}^*$, take at random strings $r_1, \ldots, r_n \in \{0,1\}^t$. For each r_i, compute $\mathsf{K}^t(x \mid r_i, y)$ by "brute force" in time $\mathsf{poly}(2^{|x|}, t, |y|)$. If the fraction of r_i's with $\mathsf{K}^t(x \mid r_i, y) \leq s(n,t)$ is at least 0.6, then accept; otherwise, reject.

Note that if x and y are such that $\mathsf{pK}^t(x \mid y) \leq s(n,t)$, then the algorithm above accepts with probability at least $1 - 2^{-\Omega(n)}$, by Chernoff bounds.

By Lemma 2.2, for any string $y \in \{0,1\}^*$ and any time bound t, the fraction of strings $x \in \{0,1\}^n$ with $\mathsf{pK}^t_{1/2}(x \mid y) \leq n-3$ is at most $1/2$. It means that for each of at least $1/2$ of strings $x \in \{0,1\}^n$, there are at most $1/2$ of strings $r \in \{0,1\}^t$ such that $\mathsf{K}^t(x \mid r, y) \leq s(n,t)$. By Chernoff bounds, getting a fraction of 0.6 such r's in a random sample of size n is at most $2^{-\Omega(n)}$. Thus our randomized algorithm will reject with high probability on each of at least $1/2$ of random strings $x \in \{0,1\}^n$. □

Theorem 8.3. $\mathsf{promise\text{-}BPP} \subseteq \mathsf{promise\text{-}P}^{\mathcal{A}}$ *for any sparse natural property $\mathcal{A}$ for conditional pK^t with usefulness $s(n,t) \geq \Omega(n)$. Hence, if for some $s(n,t) \geq \Omega(n)$, there is a P-computable sparse natural property $\mathcal{A}$ for conditional pK^t with usefulness $s(n,t)$, then* $\mathsf{promise\text{-}BPP} = \mathsf{promise\text{-}P}$*.*

Proof. For simplicity, assume that we have a P-computable sparse natural property $\mathcal{P}$ with usefulness $s(n,t) \geq n/2$; a similar argument would work for $s(n,t) \geq \Omega(n)$. Under our assumption, we will show how to solve the canonical $\mathsf{promise\text{-}BPP}$ complete problem CAPP: given a Boolean circuit $C\colon \{0,1\}^n \to \{0,1\}$ of size at most n, estimate its acceptance probability $\Pr_{z\in\{0,1\}^n}[C(x) = 1]$ to within an additive constant error, say $1/8$; see, e.g., [Vad12].

For strings $x_1, \ldots, x_n \in \{0,1\}^{10\log n}$ to be determined, consider the following generator from $10\log n$ to n bits: for $r \in \{0,1\}^{10\log n}$,

$$G^{x_1,\ldots,x_n}(r) = x_1 \cdot r, \ldots, x_n \cdot r,$$

where $x \cdot y$ denotes the inner product modulo 2 of binary strings x and y.

Suppose some circuit $C\colon \{0,1\}^n \to \{0,1\}$ (of size at most n) distinguishes between the outputs $G^{x_1,\ldots,x_n}(r)$, for uniformly random seeds $r \in \{0,1\}^{10\log n}$, and the uniform distribution $\mathcal{U}_n$, with the distinguishing probability at least $1/8$. (In other words, the generator G fails to solve CAPP on the circuit C.) We will argue this implies the following.

Claim 8.4. *There exists an $1 \le i \le n$ such that*

$$\mathsf{pK}^t(x_i \mid C, x_1, \ldots, x_{i-1}) \le 4 \log n,$$

for some $t \le \mathsf{poly}(n)$.

Proof of Claim. First, by a hybrid argument (see, e.g., [Vad12]), there must exist an index $1 \le i \le n$ such that C distinguishes between

$$x_1 \cdot r, \ldots, x_{i-1} \cdot r, x_i \cdot r, \mathcal{U}_{n-i}$$

and

$$x_1 \cdot r, \ldots, x_{i-1} \cdot r, \mathcal{U}_1, \mathcal{U}_{n-i},$$

for a random $r \in \{0,1\}^{10 \log n}$, with the distinguishing probability at least $1/(8n)$.

Next, by Yao's distinguisher-to-predictor reduction, we get a randomized predictor algorithm C' that on input r computes $x_i \cdot r$ with probability at least $1/2 + 1/(8n)$ over r and its internal randomness, given (as advice) the bits $x_1 \cdot r, \ldots, x_{i-1} \cdot r$. This $C'(r)$ samples at random a bit $b \in \{0,1\}$ and a string $w \in \{0,1\}^{n-i}$, and simulates $C(x_1 \cdot r, \ldots, x_{i-1} \cdot r, b, w)$, outputting b if C accepts, and $1-b$ if C rejects.

By an averaging argument, with probability at least $1/(16n)$ over fixing the internal randomness b, w, we get a deterministic circuit $C''_{b,w}(r)$ that computes $x_i \cdot r$ with probability at least $1/2 + 1/(16n)$ over r, when given as advice $x_1, \ldots, x_{i-1}$ (so that it can compute the required bits $x_1 \cdot r, \ldots, x_{i-1} \cdot r$). Call b, w *good* if $C''_{b,w}(r)$ is correct with probability at least $1/2 + 1/(16n)$ in computing $x_i \cdot r$.

Next, for given b, w, we build a list of all $z \in \{0,1\}^{10 \log n}$ such that

$$\Pr_r[C''_{b,w}(r) = z \cdot r] \ge 1/2 + 1/(16n). \tag{38}$$

We do so by enumerating all strings z, computing the probability in (38) for each z, and keeping only those z that satisfy (38). This can be done in deterministic time $\mathsf{poly}(n)$, say at most n^{22}. By the well-known combinatorial bound on the list size of Hadamard error-correcting codes, the size of such a list of z's is at most $(8n)^2$.[10]

For good b, w, such a list of z's must contain the string x_i, which can be identified by its index on the list. Since the list is of size at most $64n^2$, such an index is of binary length at most $2 \log n + 6$.

If we repeatedly sample random b, w for $32n$ rounds, at least one of b, w will be good with probability at least $1 - (1 - 1/(16n))^{32n} \ge 1 - e^{-2} \ge 2/3$. Assuming this happened, the description of x_i will consist of about $\log n + 5$ bits to specify the successful round, plus $2 \log n + 6$ bits to specify x_i on the list produced in that round. It follows that $\mathsf{pK}^t(x_i \mid C, x_1, \ldots, x_{i-1}) \le 4 \log n$, for $t = \mathsf{poly}(n)$, say $t \le n^{24}$. □

By the claim, if we have $x_1, \ldots, x_n$ with $\mathsf{pK}^t(x_i \mid C, x_1, \ldots, x_{i-1}) > 5 \log n$ for all $1 \le i \le n$, then $G^{x_1, \ldots, x_n}$ will fool the circuit C. That is, we get a PRG for a given circuit C, a *targeted* PRG in the terminology of [Gol11].

We can find such x_i's one by one, starting with x_1, by enumerating over all strings $z \in \{0,1\}^{10 \log n}$, until we find the first one such that $\mathcal{A}(z, C \circ x_1 \circ \cdots \circ x_{i-1}, 1^t) = 0$, for $t = n^{24}$.

[10] Each such z corresponds to a Fourier coefficient of the Boolean function $C''_{b,w}(r)$ of weight at least $1/(8n)$. By Parseval's identity, the sum of the squares of all Fourier coefficients of a given Boolean function is 1. Hence, the number of "heavy" z's is at most $(8n)^2$.

By the assumption on the natural property algorithm $\mathcal{A}$, each such z has the required conditional pK^t complexity greater than $5 \log n$. To find such $x_1, \ldots, x_n$ takes time at most $\mathsf{poly}(n)$. It follows that we can solve CAPP in polynomial time, and so $\mathsf{promise\text{-}BPP} = \mathsf{promise\text{-}P}$.

Finally, observe that to find the required strings $x_1, \ldots, x_n$, we just need oracle access to the natural property $\mathcal{A}$. So we get an (adaptive) polynomial-time Turing reduction from CAPP to any sparse natural property for conditional pK^t with usefulness $s(n,t) \geq \Omega(n)$, as required. □

Remark 8.5. *It is interesting to contrast the result in Theorem 8.3 with the result of [Buh+05] (see also [All+06]) that* $\mathsf{BPP} \subseteq \mathsf{P}^{\mathcal{K}}$*, where* $\mathcal{K}$ *is a natural property for time-unbounded Kolmogorov complexity* K*, with usefulness* $s(n) \geq \Omega(n)$*. Both results are proved by using the oracle adaptively, for a polynomial number of steps. The crucial difference is that, in their case, the oracle* $\mathcal{K}$ *is uncomputable, whereas in our case, a required sparse natural property* $\mathcal{A}$ *is* BPP*-computable.*

8.2 pK^t-Based promise-BPP Complete Problem

The sparse natural property for conditional pK^t considered above is not a promise-problem (as it has no well-defined set of No-instances). We can easily modify it to be a $\mathsf{promise\text{-}BPP}$ problem as follows.

Definition 8.6 ($\mathsf{promise\text{-}BPP}$ problem for estimating conditional pK^t on short inputs)**.** *Consider the following* $\mathsf{Gap\text{-}pK}^t$*-problem. For* $x, y \in \{0,1\}^*$ *and* $t \in \mathbb{N}$*, with* $t = \mathsf{poly}(|y|)$ *and* $|x| \leq \log t$*, we have the following Yes and No instances:*

- $\Pi_{\text{YES}} = \{(x, y, 1^t) \mid \mathsf{pK}^t_{2/3}(x \mid y) \leq |x|/2\}$
- $\Pi_{\text{NO}} = \{(x, y, 1^t) \mid \mathsf{pK}^t_{1/2}(x \mid y) > |x| - 3\}$

Theorem 8.7. *The* $\mathsf{Gap\text{-}pK}^t$*-problem of Definition 8.6 above is* $\mathsf{promise\text{-}BPP}$*-complete (under polynomial-time Turing reductions).*

Proof. That this problem is in $\mathsf{promise\text{-}BPP}$ was already argued in the proof of Lemma 8.2 above. The $\mathsf{promise\text{-}BPP}$-hardness for this problem follows from Theorem 8.3. □

We also get that the following problem to approximate the conditional K^t complexity over random conditioned strings is $\mathsf{promise\text{-}BPP}$-complete.

Corollary 8.8. *The problem to estimate, for given* $x, y \in \{0,1\}^*$ *and* $s, t \in \mathbb{N}$ *such that* $|x| \leq \log t$*, the following probability*

$$\Pr_{r \in \{0,1\}^t} \left[\mathsf{K}^t(x \mid r, y) \leq s\right]$$

to within an additive error less than $1/6$ *is* $\mathsf{promise\text{-}BPP}$*-complete (under polynomial-time Turing reductions).*

Proof. This is immediate by Theorem 8.7 and the definition of pK^t_λ. □

Remark 8.9. *We note that the work by Liu and Pass [LP22a] also studies derandomization of* $\mathsf{promise\text{-}BPP}$ *through the lens of (Levin's version) of Kolmogorov complexity. Their main result is that* $\mathsf{promise\text{-}BPP} = \mathsf{promise\text{-}P}$ *iff a certain gap-version of computing conditional Levin's complexity* $\mathsf{Kt}(x \mid y)$ *is almost-everywhere worst-case hard (for almost all strings* $y \in \{0,1\}^*$*) with respect to probabilistic polynomial-time algorithms. In contrast, we characterize* $\mathsf{promise\text{-}BPP} = \mathsf{promise\text{-}P}$ *in terms of worst-case* easiness *(being in* $\mathsf{promise\text{-}P}$*) of a* $\mathsf{promise\text{-}BPP}$ *computable version of conditional* $\mathsf{pK}^t(x \mid y)$ *problem (for logarithmically short inputs* x*).*

8.3 On Derandomizing Yao's Distinguisher-to-Predictor Transformation

Here we observe that Theorem 8.3 can be used to give an alternative proof of (one direction) of the recent result by Li, Pyne, and Tell [LPT24]. One of their main results is that derandomizing Yao's distinguisher-to-predictor transformation (of the kind described in our proof of Theorem 8.3) is *equivalent* to proving promise-BPP = promise-P. We will show the following.

Lemma 8.10 ([LPT24])**.** *If Yao's distinguisher-to-predictor transformation can be made deterministic polynomial time, then* promise-BPP = promise-P.

Proof. Given a Boolean circuit $C\colon \{0,1\}^n \to \{0,1\}$ of size n, we will try to fool it with the generator $G^{x_1,\dots,x_n}\colon \{0,1\}^{10\log n} \to \{0,1\}^n$ as in Theorem 8.3. Assuming that C is *not* fooled by this G, we follow the proof of Theorem 8.3 to get that, for some $1 \le i \le n$,

$$\mathsf{K}^t(x_i \mid C, x_1, \dots, x_{i-1}) \le 4\log n,$$

rather than $\mathsf{pK}^t(x_i \mid C, x_1, \dots, x_{i-1}) \le 4\log n$ of Claim 8.4. The reason is that the *only randomized step* in the proof of Claim 8.4 was Yao's distinguisher-to-predictor transformation. Under our current assumption that this step can be made in deterministic polynomial time, we get a bound on the *deterministic* conditional K^t description of string x_i.

Thus, to fool C, we just need to find n strings $x_1, \dots, x_n \in \{0,1\}^{10\log n}$ such that, for each $1 \le i \le n$,

$$\mathsf{K}^t(x_i \mid C, x_1, \dots, x_{i-1}) > 4\log n.$$

This can be accomplished in deterministic polynomial time, using a brute-force algorithm that, for each $i = 1, \dots, n$, looks for the lexicographically first string in $\{0,1\}^{10\log n}$ of large conditional K^t complexity, for $t \le n^{24}$. We conclude that CAPP $\in$ promise-P. □

8.4 If Conditional pK^t Were Easy

As noted earlier in Section 7.3, it is plausible (and true under cryptographic assumptions) that a gap-version of computing conditional $\mathsf{K}^t(x \mid y)$ might be NP-hard. It is reasonable to assume that a gap-version of computing conditional $\mathsf{pK}^t(x \mid y)$ might also be NP-hard. However, given the lack of actual proofs of NP-hardness for these problems, we may ask ourselves what interesting algorithmic consequences would follow if we were to assume the *easiness* of these problems instead.

Ideally, assuming such easiness, we would like to conclude that SAT is also easy (thereby essentially establishing the NP-hardness of the corresponding gap-problem). We cannot achieve this yet, but we can show the following.

Theorem 8.11. *Suppose that, for $\delta(n,t) \le \varepsilon n$, for any constant $\varepsilon > 0$, there is a polynomial τ such that* $\mathsf{Gap}_{\tau,\delta}\mathsf{McpK^tP} \in$ promise-P*. Then we have*

1. $\mathsf{E} \not\subset$ io-SIZE$[2^{o(n)}]$*, and*
2. (1/3)-CRA $\in$ FP*.*

Proof. We give two proofs. The first one will use the chain rule.

Under our assumption, we get by Corollary 3.8 the existence of a BPP-computable (actually, P-computable) natural property for conditional $\mathsf{K}^t(x \mid y)$ for $x \in \{0,1\}^n$ with usefulness $s(n,t) = n - \gamma n$, for any sufficiently small $\gamma > 0$ (dependent on ε).

Next, by Theorem 4.3, we get a chain rule for conditional pK^t: for some polynomial p, for any $\ell \in \mathbb{N}$, for any $x_1, \ldots, x_\ell \in \{0,1\}^*$ and any $y \in \{0,1\}^*$, and all sufficiently large t (at least polynomial in the total length of all strings), we have

$$\mathsf{pK}^t(x_1, \ldots, x_\ell \mid y) \geq \sum_{i=1}^{\ell} \mathsf{pK}^{p(t)}(x_i \mid y, x_1, \ldots, x_{i-1}) - \ell \cdot O(\log t) - N/D, \tag{39}$$

where $N = \sum_{i=1}^{\ell} |x_i|$ and $D > 1$ is any sufficiently large constant (dependent on γ).

By setting $\ell = N/(c \log t)$ and $|x_1| = \cdots = |x_\ell| = N/\ell = c \log t$, for a sufficiently large constant $c > 0$, we can use the chain rule above to construct, in deterministic polynomial time, a string $x = x_1, \ldots, x_\ell \in \{0,1\}^N$ of complexity $\mathsf{pK}^t(x \mid y) \geq (1-\tau)N$, for any given string $y \in \{0,1\}^*$ and any constant $\tau > 0$. We do so by constructing each $x_1, \ldots, x_\ell$ in turn, by brute-forcing over all possible strings in $\{0,1\}^{c \log t}$ and taking the first one of high conditional $\mathsf{pK}^{p(t)}$ complexity; the latter can be done in deterministic polynomial time by our assumption that $\mathsf{Gap}_{\tau,\delta}\mathsf{McpK^tP} \in \mathsf{promise\text{-}P}$.

It follows that we can construct in polynomial time a string $a \in \{0,1\}^{2^n}$ of $\mathsf{K}^{2^{O(n)}}(a) \geq \Omega(2^n)$, and hence a requires circuit size at least $\Omega(2^n/n)$. This implies item (1).

Similarly, constructing in polynomial time a string $b \in \{0,1\}^n$ with high conditional complexity $\mathsf{pK}^t(b \mid y)$ is equivalent to solving the range avoidance problem (1/3)-CRA by Lemma 2.17. This implies item (2).

The second proof. The assumption that $\mathsf{Gap}_{\tau,\delta}\mathsf{McpK^tP} \in \mathsf{promise\text{-}P}$ implies that there is a P-computable natural property for conditional pK^t with usefulness $s(n,t) \geq n/2$. By Theorem 8.3, we conclude that $\mathsf{promise\text{-}BPP} = \mathsf{promise\text{-}P}$. By the standard "search-to-decision" reduction for promise-BPP (see, e.g., [Vad12]), we can construct in deterministic polynomial time a string $x \in \{0,1\}^n$ that is accepted by a given circuit $C\colon \{0,1\}^n \to \{0,1\}$, provided C accepts a significant fraction of n-bit strings (e.g., at least 1/2). In our case, we use the assumed polynomial-time algorithm A for $\mathsf{Gap}_{\tau,\delta}\mathsf{McpK^tP}$ to look for a string $x \in \{0,1\}^n$ such that $A(x, y, 1^s, 1^t)$ rejects, for given $y \in \{0,1\}^*$ and parameters s and t. By choosing s and t appropriately, we can find strings of high (conditional) pK^t complexity in deterministic polynomial time. This suffices for items (1) and (2), as noted above. □

9 Concluding Remarks

The complexity of (variants of) K^t appears to be the right notion in the context of (variants of) chain rules for K^t. By [Hir+23] and [LP20], the *average-case* SoI (two-string chain rule) for pK^t over polynomial-time samplable distributions is equivalent to the (error-prone) *average-case* easiness of (a variant of) $\mathsf{MK^tP}$ (and also of computing conditional K^t [LP22b] and conditional pK^t [LS24]). By [LP24], it is also equivalent to the *worst-case* easiness of a certain (somewhat complicated) version of the promise-problem $\mathsf{GapMK}^t\mathsf{P}$ (where the inputs are restricted to be of "shallow computational depth").

Our work shows that the *worst-case multi-string* chain rule for (conditional) pK^t is equivalent to $\mathsf{Gap}_{\mathsf{poly},o(n)}\mathsf{M(c)pK^tP} \in \mathsf{promise\text{-}BPP}$, and the chain rule for (conditional) K^t is equivalent to $\mathsf{Gap}_{\mathsf{poly},o(n)}\mathsf{M(c)K^tP} \in \mathsf{promise\text{-}P}$ (and exponential circuit lower bounds for E).

A very interesting open question is whether $\mathsf{Gap}_{\mathsf{poly},o(n)}\mathsf{McK^tP}$ (or $\mathsf{Gap}_{\mathsf{poly},o(n)}\mathsf{McpK^tP}$) can be shown NP-hard unconditionally (without any cryptographic assumptions). Such a result would

provide an extremely clean equivalence: the chain rule for conditional K^t (or pK^t) is equivalent to $\mathsf{P} = \mathsf{NP}$ (or $\mathsf{NP} \subseteq \mathsf{BPP}$).

Another open question is to understand what complexity assumption would be equivalent to the worst-case SoI, or chain rule on a constant number of strings, for (conditional) K^t or pK^t. One specific technical challenge is to try to derive a polynomial-time computable natural property, say for K^t, from the assumed SoI for K^t. It is possible to get a $2^{O(n/\log n)}$ computable natural property from SoI for K^t (see Appendix C), but it is not clear how to get time, say $2^{O(\sqrt{n})}$.

Can one prove better circuit lower bounds from the assumed chain rule for pK^t? Under that assumption, we can show that $\mathsf{BPEXP} \not\subset \mathsf{P/poly}$. Is it possible to show, under the same assumption, that $\mathsf{BPEXP} \not\subset \mathsf{SIZE}[2^{o(n)}]$?

Finally, returning to Kolmogorov's suggestion of using the chain rules to attack the "P vs. NP" question, we wonder if one can disprove the multi-string chain rule for K^t for some specific choices of parameters. For example, consider a string $x \in \{0,1\}^n$ constructed as $x = x_1 \dots x_\ell$, for $\ell = n/(4 \log n)$ and $|x_i| = 4 \log n$ for all $1 \leq i \leq \ell$ via the following $\mathsf{poly}(n)$-time algorithm: "For $i = 1$ to ℓ, construct x_i by brute-force as the lexicographically first string such that $\mathsf{K}^{n^3}(x_i \mid x_1, \dots, x_{i-1}) \geq |x_i|$." Can we show $\mathsf{K}^{n^2}(x) < 0.9n$? Or, is it reasonable to hypothesize that such a $\mathsf{poly}(n)$-time algorithm may produce n-bit strings of high circuit complexity?

Acknowledgements. We wish to thank the organizers of the workshop at CIRM on *Randomness, Information, and Complexity*, February 19-23, 2024, where we started working on the present paper. Special thanks are to Andrei Romashchenko for bringing [Per07] to our attention, and for his comments on an early draft of this paper. We wish to thank Igor C. Oliveira and the participants of his complexity theory online seminar for their comments on the early presentation of this work; special thanks are to Jinqiao Hu who reminded us about the weak-designs-based PRG G_k^x. Also special thanks are to Shuichi Hirahara and Mikito Nanashima who pointed out a missing step in our earlier proof of the multi-string chain rule; we are grateful to Shuichi Hirahara for explaining to us the main idea of the "parameter-tuning" approach in [HN25] (adapting the arguments from [Hir18; Hir20b]). Finally, we thank the organizers of the Clay Mathematics Institute workshop on *P vs NP and Complexity Lower Bounds*, September 29 - October 3, 2025, where some of these discussions took place.

References

[All+06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. "Power from Random Strings". In: *SIAM J. Comput.* 35.6 (2006), pp. 1467–1493. DOI: 10.1137/050628994.

[Bau+22] Bruno Bauwens, Péter Gács, Andrei E. Romashchenko, and Alexander Shen. "Inequalities for space-bounded Kolmogorov complexity". In: *Computability* 11.3-4 (2022), pp. 165–185. DOI: 10.3233/COM-210374.

[Bin+22] Eric Binnendyk, Marco Carmosino, Antonina Kolokolova, R. Ramyaa, and Manuel Sabin. "Learning with Distributional Inverters". In: *International Conference on Algorithmic Learning Theory, 29 March - 1 April 2022, Paris, France*. Ed. by Sanjoy Dasgupta and Nika Haghtalab. Vol. 167. Proceedings of Machine Learning Research. PMLR, 2022, pp. 90–106.

[BLM05] Harry Buhrman, Troy Lee, and Dieter van Melkebeek. "Language compression and pseudorandom generators". In: *Comput. Complex.* 14.3 (2005), pp. 228–255. DOI: 10.1007/S00037-005-0199-5.

[Buh+05] Harry Buhrman, Lance Fortnow, Ilan Newman, and Nikolai K. Vereshchagin. "Increasing Kolmogorov Complexity". In: *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*. Ed. by Volker Diekert and Bruno Durand. Vol. 3404. Lecture Notes in Computer Science. Springer, 2005, pp. 412–421. DOI: 10.1007/978-3-540-31856-9_34.

[Car+16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. "Learning Algorithms from Natural Proofs". In: *Conference on Computational Complexity* (CCC). 2016, 10:1–10:24.

[Car+17] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. "Agnostic Learning from Tolerant Natural Proofs". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*. Ed. by Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala. Vol. 81. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 35:1–35:19. DOI: 10.4230/LIPICS.APPROX-RANDOM.2017.35.

[CHV22] Lijie Chen, Shuichi Hirahara, and Neekon Vafa. "Average-Case Hardness of NP and PH from Worst-Case Fine-Grained Assumptions". In: *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*. Ed. by Mark Braverman. Vol. 215. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 45:1–45:16. DOI: 10.4230/LIPIcs.ITCS.2022.45.

[Gaj+23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. "Range Avoidance for Constant Depth Circuits: Hardness and Algorithms". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, September 11-13, 2023, Atlanta, Georgia, USA*. Ed. by Nicole Megow and Adam D. Smith. Vol. 275. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 65:1–65:18. DOI: 10.4230/LIPICS.APPROX/RANDOM.2023.65.

[Gar+13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. "Witness encryption and its applications". In: *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM, 2013, pp. 467–476. DOI: 10.1145/2488608.2488667.

[GK22] Halley Goldberg and Valentine Kabanets. "A Simpler Proof of the Worst-Case to Average-Case Reduction for Polynomial Hierarchy via Symmetry of Information". In: *Electron. Colloquium Comput. Complex.* TR22-007 (2022). ECCC: TR22-007.

[GK23] Halley Goldberg and Valentine Kabanets. "Improved Learning from Kolmogorov Complexity". In: *38th Computational Complexity Conference, CCC 2023, July 17-20, 2023, Warwick, UK*. Ed. by Amnon Ta-Shma. Vol. 264. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 12:1–12:29. DOI: 10.4230/LIPICS.CCC.2023.12.

[GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. "Range Avoidance for Low-Depth Circuits and Connections to Pseudorandomness". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*. Ed. by Amit Chakrabarti and Chaitanya Swamy. Vol. 245. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 20:1–20:21. DOI: 10.4230/LIPICS.APPROX/RANDOM.2022.20.

[Gol+22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. "Probabilistic Kolmogorov Complexity with Applications to Average-Case Complexity". In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Ed. by Shachar Lovett. Vol. 234. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 16:1–16:60. DOI: 10.4230/LIPICS.CCC.2022.16.

[Gol11] Oded Goldreich. "In a World of P=BPP". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*. Ed. by Oded Goldreich. Vol. 6650. Lecture Notes in Computer Science. Springer, 2011, pp. 191–232. DOI: 10.1007/978-3-642-22670-0_20.

[Hås+99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function". In: *SIAM J. Comput.* 28.4 (1999), pp. 1364–1396. DOI: 10.1137/S0097539793244708.

[Hir+23] 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: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*. Ed. by Barna Saha and Rocco A. Servedio. ACM, 2023, pp. 1039–1050. DOI: 10.1145/3564246.3585138.

[Hir+24] Shuichi Hirahara, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. "Exact Search-To-Decision Reductions for Time-Bounded Kolmogorov Complexity". In: *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*. Ed. by Rahul Santhanam. Vol. 300. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 29:1–29:56. DOI: 10.4230/LIPICS.CCC.2024.29.

[Hir18] Shuichi Hirahara. "Non-Black-Box Worst-Case to Average-Case Reductions within NP". In: *Symposium on Foundations of Computer Science* (FOCS). 2018, pp. 247–258.

[Hir20a] Shuichi Hirahara. "Characterizing Average-Case Complexity of PH by Worst-Case Meta-Complexity". In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*. Ed. by Sandy Irani. IEEE, 2020, pp. 50–60. DOI: 10.1109/FOCS46700.2020.00014.

[Hir20b] Shuichi Hirahara. "Non-Disjoint Promise Problems from Meta-Computational View of Pseudorandom Generator Constructions". In: *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*. Ed. by Shubhangi Saraf. Vol. 169. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 20:1–20:47. DOI: 10.4230/LIPICS.CCC.2020.20.

[Hir20c] Shuichi Hirahara. "Unexpected hardness results for Kolmogorov complexity under uniform reductions". In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*. Ed. by Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy. ACM, 2020, pp. 1038–1051. DOI: 10.1145/3357713.3384251.

[Hir21] Shuichi Hirahara. "Average-case hardness of NP from exponential worst-case hardness assumptions". In: *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*. Ed. by Samir Khuller and Virginia Vassilevska Williams. ACM, 2021, pp. 292–302. DOI: 10.1145/3406325.3451065.

[Hir22] Shuichi Hirahara. "Symmetry of Information from Meta-Complexity". In: *Computational Complexity Conference* (CCC). 2022, 26:1–26:41. DOI: 10.4230/LIPIcs.CCC.2022.26.

[HIR23] Yizhi Huang, Rahul Ilango, and Hanlin Ren. "NP-Hardness of Approximating Meta-Complexity: A Cryptographic Approach". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*. Ed. by Barna Saha and Rocco A. Servedio. ACM, 2023, pp. 1067–1075. DOI: 10.1145/3564246.3585154.

[Hir23] Shuichi Hirahara. "Capturing One-Way Functions via NP-Hardness of Meta-Complexity". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*. Ed. by Barna Saha and Rocco A. Servedio. ACM, 2023, pp. 1027–1038. DOI: 10.1145/3564246.3585130.

[HN21] Shuichi Hirahara and Mikito Nanashima. "On Worst-Case Learning in Relativized Heuristica". In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. IEEE, 2021, pp. 751–758. DOI: 10.1109/FOCS52979.2021.00078.

[HN25] Shuichi Hirahara and Mikito Nanashima. "Complexity-Theoretic Inductive Inference". In: *Electron. Colloquium Comput. Complex.* TR25-092 (2025). ECCC: TR25-092.

[IKV23] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. "The Power of Natural Properties as Oracles". In: *Comput. Complex.* 32.2 (2023), p. 6. DOI: 10.1007/S00037-023-00241-0.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. "In Search of an Easy Witness: Exponential Time vs. Probabilistic Polynomial Time". In: *J. Comput. Syst. Sci.* 65.4 (2002), pp. 672–694. DOI: 10.1016/S0022-0000(02)00024-7.

[Ila23] Rahul Ilango. "SAT Reduces to the Minimum Circuit Size Problem with a Random Oracle". In: *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*. IEEE, 2023, pp. 733–742. DOI: 10.1109/FOCS57990.2023.00048.

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. "Indistinguishability Obfuscation, Range Avoidance, and Bounded Arithmetic". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*. Ed. by Barna Saha and Rocco A. Servedio. ACM, 2023, pp. 1076–1089. DOI: 10.1145/3564246.3585187.

[IRS22] Rahul Ilango, Hanlin Ren, and Rahul Santhanam. "Robustness of average-case meta-complexity via pseudorandomness". In: *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. Ed. by Stefano Leonardi and Anupam Gupta. ACM, 2022, pp. 1575–1583. DOI: 10.1145/3519935.3520051.

[IW97] Russell Impagliazzo and Avi Wigderson. "$P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma". In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*. Ed. by Frank Thomson Leighton and Peter W. Shor. ACM, 1997, pp. 220–229. DOI: 10.1145/258533.258590.

[Jer04] Emil Jerábek. "Dual weak pigeonhole principle, Boolean complexity, and derandomization". In: *Ann. Pure Appl. Log.* 129.1-3 (2004), pp. 1–37. DOI: 10.1016/J.APAL.2003.12.003.

[Jeř07] Emil Jeřábek. "On independence of variants of the weak pigeonhole principle". In: *J. Log. Comput.* 17.3 (2007), pp. 587–604. DOI: 10.1093/logcom/exm017.

[Kan82] Ravi Kannan. "Circuit-Size Lower Bounds and Non-Reducibility to Sparse Sets". In: *Inf. Control.* 55.1-3 (1982), pp. 40–56. DOI: 10.1016/S0019-9958(82)90382-5.

[Kar24] Ari Karchmer. "Distributional PAC-Learning from Nisan's Natural Proofs". In: *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*. Ed. by Venkatesan Guruswami. Vol. 287. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 68:1–68:23. DOI: 10.4230/LIPICS.ITCS.2024.68.

[KC00] Valentine Kabanets and Jin-yi Cai. "Circuit minimization problem". In: *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*. Ed. by F. Frances Yao and Eugene M. Luks. ACM, 2000, pp. 73–79. DOI: 10.1145/335305.335314.

[Kle+21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. "Total Functions in the Polynomial Hierarchy". In: *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*. Ed. by James R. Lee. Vol. 185. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 44:1–44:18. DOI: 10.4230/LIPICS.ITCS.2021.44.

[Ko91] Ker-I Ko. "On the Complexity of Learning Minimum Time-Bounded Turing Machines". In: *SIAM J. Comput.* 20.5 (1991), pp. 962–986. DOI: 10.1137/0220059.

[Kor21] Oliver Korten. "The Hardest Explicit Construction". In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. IEEE, 2021, pp. 433–444. DOI: 10.1109/FOCS52979.2021.00051.

[Kra01] Jan Krajícek. "Tautologies from pseudo-random generators". In: *Bull. Symb. Log.* 7.2 (2001), pp. 197–212. DOI: 10.2307/2687774.

[Kra04] Jan Krajícek. "Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds". In: *J. Symb. Log.* 69.1 (2004), pp. 265–286. DOI: 10.2178/JSL/1080938841.

[Lev03] Leonid A. Levin. "The Tale of One-Way Functions". In: *Probl. Inf. Transm.* 39.1 (2003), pp. 92–103. DOI: 10.1023/A\%3A1023634616182.

[LM93] Luc Longpré and Sarah Mocas. "Symmetry of Information and One-Way Functions". In: *Inf. Process. Lett.* 46.2 (1993), pp. 95–100. DOI: 10.1016/0020-0190(93)90204-M.

[LO22] Zhenjian Lu and Igor C. Oliveira. "Theory and Applications of Probabilistic Kolmogorov Complexity". In: *Bull. EATCS* 137 (2022).

[LP20] Yanyi Liu and Rafael Pass. "On One-way Functions and Kolmogorov Complexity". In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*. 2020, pp. 1243–1254. DOI: 10.1109/FOCS46700.2020.00118.

[LP22a] Yanyi Liu and Rafael Pass. "Characterizing Derandomization Through Hardness of Levin-Kolmogorov Complexity". In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Ed. by Shachar Lovett. Vol. 234. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 35:1–35:17. DOI: 10.4230/LIPICS.CCC.2022.35.

[LP22b] Yanyi Liu and Rafael Pass. "On One-Way Functions from NP-Complete Problems". In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Ed. by Shachar Lovett. Vol. 234. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 36:1–36:24. DOI: 10.4230/LIPICS.CCC.2022.36.

[LP24] Yanyi Liu and Rafael Pass. "On One-Way Functions, the Worst-Case Hardness of Time-Bounded Kolmogorov Complexity, and Computational Depth". In: *Theory of Cryptography - 22nd International Conference, TCC 2024, Milan, Italy, December 2-6, 2024, Proceedings, Part I*. Ed. by Elette Boyle and Mohammad Mahmoody. Vol. 15364. Lecture Notes in Computer Science. Springer, 2024, pp. 222–252. DOI: 10.1007/978-3-031-78011-0_8.

[LPT24] Jiatu Li, Edward Pyne, and Roei Tell. "Distinguishing, Predicting, and Certifying: On the Long Reach of Partial Notions of Pseudorandomness". In: *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*. IEEE, 2024, pp. 1–13. DOI: 10.1109/FOCS61266.2024.00095.

[LR05] Troy Lee and Andrei E. Romashchenko. "Resource bounded symmetry of information revisited". In: *Theor. Comput. Sci.* 345.2-3 (2005), pp. 386–405. DOI: 10.1016/J.TCS.2005.07.017.

[LS24] Zhenjian Lu and Rahul Santhanam. "Impagliazzo's Worlds Through the Lens of Conditional Kolmogorov Complexity". In: *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*. Ed. by Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson. Vol. 297. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 110:1–110:17. DOI: 10.4230/LIPICS.ICALP.2024.110.

[LV19] 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.

[LW95] Luc Longpré and Osamu Watanabe. "On Symmetry of Information and Polynomial Time Invertibility". In: *Inf. Comput.* 121.1 (1995), pp. 14–22. DOI: 10.1006/inco.1995.1120.

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs Randomness". In: *J. Comput. Syst. Sci.* 49.2 (1994), pp. 149–167. DOI: 10.1016/S0022-0000(05)80043-1.

[OS17] Igor C. Oliveira and Rahul Santhanam. "Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness". In: *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*. Ed. by Ryan O'Donnell. Vol. 79. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 18:1–18:49. DOI: 10.4230/LIPICS.CCC.2017.18.

[Per07] Sylvain Perifel. "Symmetry of Information and Nonuniform Lower Bounds". In: *Computer Science - Theory and Applications, Second International Symposium on Computer Science in Russia, CSR 2007, Ekaterinburg, Russia, September 3-7, 2007, Proceedings*. Ed. by Volker Diekert, Mikhail V. Volkov, and Andrei Voronkov. Vol. 4649. Lecture Notes in Computer Science. Springer, 2007, pp. 315–327. DOI: 10.1007/978-3-540-74510-5_32.

[Ron04] Detlef Ronneburger. "Kolmogorov Complexity and Derandomization". PhD thesis. Rutgers University, 2004.

[RR97] Alexander A. Razborov and Steven Rudich. "Natural Proofs". In: *J. Comput. Syst. Sci.* 55.1 (1997), pp. 24–35. DOI: 10.1006/JCSS.1997.1494.

[RRV02] Ran Raz, Omer Reingold, and Salil P. Vadhan. "Extracting all the Randomness and Reducing the Error in Trevisan's Extractors". In: *J. Comput. Syst. Sci.* 65.1 (2002), pp. 97–128. DOI: 10.1006/JCSS.2002.1824.

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. "On the Range Avoidance Problem for Circuits". In: *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*. IEEE, 2022, pp. 640–650. DOI: 10.1109/FOCS54457.2022.00067.

[Tre01] Luca Trevisan. "Extractors and pseudorandom generators". In: *J. ACM* 48.4 (2001), pp. 860–879. DOI: 10.1145/502090.502099.

[Vad12] Salil P. Vadhan. "Pseudorandomness". In: *Found. Trends Theor. Comput. Sci.* 7.1-3 (2012), pp. 1–336. DOI: 10.1561/0400000010.

[ZL70] A.K. Zvonkin and L.A. Levin. "The Complexity of Finite Objects and the Development of the Concepts of Information and Randomness by means of the Theory of Algorithms". In: *Russian Mathematical Surveys* 25 (1970), pp. 83–124.

A Proof of SoI for Time-Unbounded Kolmogorov Complexity

We sketch the standard argument proving the classical result of Kolmogorov and Levin on SoI for K. For any $x, y \in \{0,1\}^*$, the upper bound

$$\mathsf{K}(x,y) \leq \mathsf{K}(x) + \mathsf{K}(y \mid x) + O(\log(|x| + |y|))$$

is straightforward. We prove the lower bound next, following the presentation in [Bau+22].

Theorem A.1 (Chain Rule for K [ZL70])**.** *For any $x, y \in \{0,1\}^*$,*

$$\mathsf{K}(x,y) \geq \mathsf{K}(x) + \mathsf{K}(y \mid x) - O(\log(|x| + |y|)).$$

Proof. Define $m = \mathsf{K}(x, y)$. Consider all pairs (x', y') such that $\mathsf{K}(x', y') \leq m$. Call this set S. There are at most $O(2^m)$ of such pairs in S, and our pair (x, y) is among them.

Define $S_x \subseteq S$ to be all pairs $(x, y') \in S$ that have x in the first coordinate. Choose an integer k so that $2^k \leq |S_x| \leq 2^{k+1}$.

Observe that y can be reconstructed from x, if we know the ordinal number of (x, y) in the enumeration of all pairs in S_x (this requires $k + O(1)$ bits) and if we know m (so we can start the enumeration). Overall, we have that

$$\mathsf{K}(y \mid x) \leq k + O(\log m). \tag{40}$$

We can enumerate all x' such that there are at least 2^k different y' such that $\mathsf{K}(x', y') \leq m$ (i.e., $(x', y') \in S$). There are at most $O(2^{m-k})$ of such strings x', and our x is among them. We can specify x by its ordinal number in this enumeration (which requires at most $m - k + O(1)$ bits), plus we need to know m and k (which requires at most $O(\log m)$ bits). Overall, we get that

$$\mathsf{K}(x) \leq m - k + O(\log m). \tag{41}$$

Adding up (40) and (41), and recalling the definition of $m = \mathsf{K}(x, y)$, we get

$$\mathsf{K}(x) + \mathsf{K}(y \mid x) \leq \mathsf{K}(x, y) + O(\log m),$$

as required. □

B Inverting One-Way Functions from a Chain Rule

Longpré and Watanabe [LW95] proved that a chain rule for K^t, for two strings (Symmetry of Information for K^t) implies that no one-way functions exist. That is, every one-way function candidate $f\colon \{0,1\}^* \to \{0,1\}^*$ (which is length-preserving) can be inverted by a randomized algorithm, with high probability over uniformly random inputs x to f. Using Theorem 5.2 and Theorem 6.2, we get the following version of their result.

Theorem B.1. *Assume the Chain Rule for K^t as in Eq. (16) of Theorem 4.2, for some $\delta(n) \leq o(n^{1/3})$. Then any function family f of length-preserving functions $f = \{f_n\colon \{0,1\}^n \to \{0,1\}^n\}$, computable in polynomial time, can be inverted in deterministic polynomial time on almost all inputs.*

Proof Sketch. By [Hås+99], one-way functions exist if and only if there are secure PRGs with polynomial stretch, i.e., polynomial-time computable $G\colon \{0,1\}^{n^\varepsilon} \to \{0,1\}^n$, for some $0 < \varepsilon < 1$, such that no randomized polynomial-time algorithm can distinguish the outputs of G from the uniform distribution with a distinguishing probability at least $1/\mathsf{poly}(n)$, for some polynomial $\mathsf{poly}(n)$. Note that, for every seed $z \in \{0,1\}^{n^\varepsilon}$, we have $\mathsf{K}^t(G(z)) \leq |z| + O(1)$, for t being the polynomial runtime of the generator G. Hence, a polynomial-time computable natural property for K^t on n-bit strings with usefulness $s(n) = n^\varepsilon$ would be a distinguisher for G.

By Theorem 5.2, we get from our assumption the existence of a polynomial-time computable natural property with even larger usefulness $s(n) = 0.9 \cdot n$. Thus, we can break any candidate PRG G. Hence, by [Hås+99], we get a randomized polynomial-time algorithm A for inverting a candidate one-way function family f that the PRG G was based on. This algorithm A succeeds

(with high probability over its internal randomness) on almost all inputs $y = f(x)$ for uniformly random $x \in \{0,1\}^n$.

Note that such a randomized inverting algorithm A for f can also always check in polynomial time whether it has succeeded at getting an inverse z of f on a given input $y = f(x)$ (by checking if $f(z) = y$). Hence, we can derandomize this randomized inverter A in polynomial time, using a Nisan-Wigderson-style PRG (from $\{0,1\}^{O(\log n)}$ to $\{0,1\}^{\mathsf{poly}(n)}$ bits) that exists under the assumption that $\mathsf{E} \not\subset \mathsf{io\text{-}SIZE}[2^{o(n)}]$ [NW94; IW97]; see Theorem 2.4. The latter assumption holds in our case by Theorem 6.2. The theorem follows. □

We also get an analogue of Theorem B.1 from the Chain Rule for pK^t as in in Eq. (21) of Theorem 4.3, except we end up with a randomized inverting algorithm rather than a deterministic one.

C Natural property from SoI

Borrowing some ideas from [LW95] and [Hir21] (see also [Hir22]), we show how to achieve usefulness $s(n) = n - 2$, in time $2^{O(n/\log n)}$, from SoI.

We need the following lemma.

Lemma C.1 (Computational Depth Upper Bound [Hir21])**.** *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 $q_{\mathsf{dpt}}(n) \le t^* \le 2^{n^\varepsilon}$, and*

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

Theorem C.2. *Assume the Chain Rule for K^t as in Eq. (16) of Theorem 4.2, for some $\delta(n,t) \le O(n^{1/2}/\log n)$. Let t be any sufficiently large polynomial in n. Then there is a $2^{O(n/\log n)}$-time computable natural property $\mathcal{A}$ for K^t on n-bit inputs with usefulness $s(n) = n - 2$.*

Proof. For any $x \in \{0,1\}^n$ and any time bound $\tau \in \mathbb{N}$, let $y_\tau \in \{0,1\}^{\mathsf{K}^\tau(x)}$ be the lexicographically first string of length $\mathsf{K}^\tau(x)$ such that the universal TM $U(y)$ outputs x within τ steps; that is, y_τ is the lexicographically first $\mathsf{K}^\tau(x)$-witness.

By the 2-string Chain Rule (Symmetry of Information) for strings x and y_t, we have for some polynomial p that, for any large τ,

$$\begin{aligned}
\mathsf{K}^{p(\tau)}(y_\tau \mid x) &\le \mathsf{K}^{2\tau}(x, y_\tau) - \mathsf{K}^{p(\tau)}(x) + O(\log \tau) + O(n/\log n) \\
&\le \mathsf{K}^{\tau}(y_\tau) - \mathsf{K}^{p(\tau)}(x) + O(\log \tau) + O(n/\log n) && (y_\tau \text{ determines } x \text{ in } \tau \text{ steps}) \\
&\le |y_\tau| - \mathsf{K}^{p(\tau)}(x) + O(\log \tau) + O(n/\log n) && (\text{since } \mathsf{K}^t(y_\tau) \le |y_\tau| + O(1)) \\
&= \mathsf{K}^{\tau}(x) - \mathsf{K}^{p(\tau)}(x) + O(\log \tau) + O(n/\log n). && (\text{since } |y_\tau| = \mathsf{K}^\tau(x))
\end{aligned}$$

By Lemma C.1, for every $\varepsilon > 0$, there exists a time bound $t \le t^* \le 2^{n^\varepsilon}$ such that

$$\mathsf{K}^{t^*}(x) - \mathsf{K}^{p(t^*)}(x) \le O(n/\log n).$$

Hence, by the above, we get for $\tau = t^*$ that

$$\mathsf{K}^{p(t^*)}(y_{t^*} \mid x) \le O(n/\log n).$$

It follows that, by exhaustive search in time $2^{O(n/\log n)}$, we can find a list of strings that contains the $\mathsf{K}^{t^*}(x)$-witness y_{t^*}. Let y be the smallest-length string on the list such that $U(y)$ prints out x within t' time steps, for $t \leq t' \leq 2^{n^\varepsilon}$. By the choice of y, the definition of y_{t^*}, and since $t^* \geq t$, we have that

$$|y| \leq |y_{t^*}| = \mathsf{K}^{t^*}(x) \leq \mathsf{K}^t(x). \tag{42}$$

On the other hand, since y is a time-bounded Kolmogorov description of x, we also have

$$K(x) \leq |y|. \tag{43}$$

Consider the following algorithm $\mathcal{A}$:

> On input $x \in \{0,1\}^n$ and 1^t, try all strings $w \in \{0,1\}^{O(n/\log n)}$, running $U(w)$ for at most 2^{n^ε} steps on each. Collect all those outputs y of $U(w)$ such that $U(y)$ prints x within 2^{n^ε} steps. Let d be the length of the shortest such y. If $d \leq s(n)$, then accept; otherwise, reject.

Observe that $\mathcal{A}$ runs in time $2^{O(n/\log n)}$. For correctness, if $\mathsf{K}^t(x) \leq s(n)$, then by (42), $\mathcal{A}(x, 1^t)$ will accept. On the other hand, by a simple counting argument, with probability at least $1 - 2^{-c+1}$ over uniformly random $x \in \{0,1\}^n$, we have that $\mathsf{K}(x) > n - c$. Hence, by (43), $\mathcal{A}(x, 1^t)$ will reject at least $1/2$ random strings $x \in \{0,1\}^n$, for $s(n) = n - 2$. □

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il