Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 203 (2025)

Hardness of Computing Nondeterministic Kolmogorov Complexity

Jinqiao Hu*
Department of Computer Science
University of Warwick

Zhenjian Lu†
Department of Computer Science
University of Victoria

Igor C. Oliveira‡
Department of Computer Science
University of Warwick

February 8, 2026

Abstract

Meta-complexity investigates the complexity of computational problems and tasks that are themselves about computations and their complexity. Understanding whether such problems can capture the hardness of NP is a central research direction. A longstanding open problem in this area is to establish the NP-hardness of MINKT [Ko91], the problem of estimating time-bounded Kolmogorov complexity.

We contribute to this research direction by studying nK^t, a natural variant of Kolmogorov complexity that captures the complexity of representing a string using time-bounded *nondeterministic* computations [BFL01]. Let MINnKT denote the task of estimating $\mathsf{nK}^t(x)$ of a given input string x. We prove that $\mathsf{MINnKT} \in \mathsf{BPP}$ if and only if $\mathsf{NP} \subseteq \mathsf{BPP}$. In contrast with prior work, this result provides the first non-conditional, non-oracle, non-partial version of a natural meta-computational problem whose hardness characterizes $\mathsf{NP} \nsubseteq \mathsf{BPP}$.

Crucial to our result is the investigation of a new notion of *probabilistic nondeterministic* time-bounded Kolmogorov complexity called pnK^t. This measure can be seen as an extension of pK^t complexity [GKLO22a] obtained by replacing K^t with nK^t. We establish unconditionally that pnK^t has nearly all key properties of (time-unbounded) Kolmogorov complexity, such as language compression, conditional coding, and a form of symmetry of information. Finally, we show that the corresponding meta-computational problem MINpnKT also captures the hardness of NP, and that extending this result to the closely related problem Gap-MINpnKT would imply the exclusion of PH-Heuristica.

*E-mail: jinqiao.hu@warwick.ac.uk
†E-mail: zhenjianlu@uvic.ca
‡E-mail: igor.oliveira@warwick.ac.uk

ISSN 1433-8092

Contents

1 Introduction

1.1 Overview

Meta-complexity investigates the complexity of computational problems and tasks that are themselves about computations and their complexity. Two central problems in meta-complexity are MCSP [KC00], the task of estimating the circuit complexity of a given function, and MINKT [Ko91], the task of estimating the time-bounded Kolmogorov complexity of a given string. The study of the computational complexity of these and related problems has had a significant impact in theoretical computer science over the last decade, with influential developments in areas such as learning theory [CIKK16], cryptography [LP20], and average-case complexity [Hir21].

A major open problem is to show the NP-hardness of MCSP and MINKT. Beyond the intrinsic interest in establishing the hardness of natural problems, recent results have highlighted the broader significance of this research direction for algorithms and complexity theory. In particular, NP-hardness results in meta-complexity offer a promising approach toward excluding Heuristica, i.e., the quest to establish an equivalence between the worst-case and average-case complexities of NP (see [Hir22a]). In more detail, under the average-case easiness of NP, certain "gap" versions of MINKT and MCSP admit worst-case to average-case reductions. As a result, if solving Gap-MINKT in the worst case implies that NP is easy in the worst case, then the average-case easiness of NP (which entails the average-case easiness of MINKT and, via the worst-case to average-case reduction, the worst-case easiness of Gap-MINKT) would also lead to the worst-case easiness of NP. Ruling out Heuristica would constitute a major breakthrough in complexity theory and a crucial step toward basing cryptography on the worst-case assumption that $\mathsf{NP} \not\subseteq \mathsf{BPP}$.

In a celebrated result, Hirahara [Hir22b] established the NP-hardness of "partial" versions of MINKT and MCSP, where the input string $x \in \{0,1,\star\}^n$ is allowed to contain unspecified bits encoded by "$\star$". Unfortunately, for such computational problems worst-case to average-case reductions remain unknown. This further motivates the task of establishing the hardness of "total" meta-computational problems, such as MINKT and its variants, which tend to be susceptible to worst-case to average-case reductions.

Summary of Contributions. We explore nK^t complexity, a natural variant of Kolmogorov complexity that considers the complexity of describing a string using time-bounded *nondeterministic* computations. We employ this extension of time-bounded Kolmogorov complexity to develop the meta-complexity of nondeterministic Kolmogorov complexity. In more detail, our main contributions can be summarized as follows:

– We show that the task of computing $\mathsf{nK}^t(x)$ of a given string $x \in \{0,1\}^n$ captures the worst-case hardness of NP. This hardness result holds for standard strings and does not rely on the use of "$\star$". In order to prove this result, we introduce conceptual and technical ideas that are likely to find further applications.

– In particular, we introduce a useful notion of probabilistic nondeterministic Kolmogorov complexity called pnK^t, and establish that it has nearly all key properties of (time-unbounded) Kolmogorov complexity. This means that in many applications we can replace Kolmogorov complexity by a time-bounded measure with similar features, thereby retaining information about the complexity of computations.

– Finally, we describe a striking connection between the complexity of computing $\mathsf{nK}^t(x)$ and its variants and average-case complexity. More precisely, we show that a seemingly minor extension of our hardness results would have a significant consequence to the average-case complexity of the polynomial hierarchy.

Next, we explain our results and contributions in detail.

1.2 Results

1.2.1 Capturing the Hardness of NP via Nondeterministic Kolmogorov Complexity

For $x \in \{0,1\}^*$ and $t \in \mathbb{N}$, the *t-time-bounded nondeterministic Kolmogorov complexity of x* is defined as

$$\mathsf{nK}^t(x) := \min_{\text{program } \Pi} \left\{ |\Pi| \;\middle|\; \begin{array}{l} \bullet\ \forall w \in \{0,1\}^t, \Pi(w) \text{ outputs } x \text{ or } \perp \text{ in } t \text{ steps} \\ \bullet\ \exists w \in \{0,1\}^t, \Pi(w) \text{ outputs } x \text{ in } t \text{ steps} \end{array} \right\}.$$

We extend this definition to $\mathsf{nK}^t(x \mid y)$, the conditional version of nK^t, in the natural way (see Section 2.3). The complexity measure nK^t was studied under the name CND^t in [BFL01] and CN^t in [BLvM05].[1]

Let MINnKT be the following problem: Given $(x, 1^s, 1^t)$, where $x \in \{0,1\}^*$ and $s, t \in \mathbb{N}$, decide whether $\mathsf{nK}^t(x) \leq s$. Note that $\mathsf{MINnKT} \in \Sigma_2^p$, the second level of the polynomial hierarchy.

Theorem 1 stated below requires a computational model that behaves well with respect to the time and description length overhead of composing computations. We refer to Section 1.3 and Section 2 for more details about the mild conditions imposed on the encodings of programs. (We note that the conditions we need are often implicitly assumed in the literature on meta-complexity and time-bounded Kolmogorov complexity.)

Main Contribution. We are ready to state the main result of this paper.

Theorem 1 (Characterizing $\mathsf{NP} \not\subseteq \mathsf{BPP}$ by the Worst-Case Hardness of nK^t)**.**
The following equivalence holds.

$$\mathsf{MINnKT} \in \mathsf{BPP} \iff \mathsf{NP} \subseteq \mathsf{BPP}.$$

Recall that it is a longstanding open problem to show the hardness of MINKT [Ko91]. Theorem 1 can be interpreted as a solution to this problem in the setting of nondeterministic time-bounded Kolmogorov complexity. In contrast with prior work, such as [Hir20c, ACM+21, Hir22b, Hir22c, LP22, HIR23, Ila23], this result provides the first non-conditional, non-oracle, non-partial version of a natural meta-computational problem whose hardness *characterizes* $\mathsf{NP} \not\subseteq \mathsf{BPP}$.

In order to establish Theorem 1, we rely on a new approach that explores a delicate regime of parameters of the MINnKT instances. We explain this in more detail in Section 1.2.3, where we also discuss its relevance to average-case complexity. We note that, due to the use of non-black-box techniques from meta-complexity, our result does not provide a Karp-reduction from an NP-complete problem to MINnKT. In any case, the equivalence established in Theorem 1 is sufficient when exploring implications to average-case complexity (see Section 1.2.3).

Related Results. Next, we show that if MINnKT is average-case easy under the uniform distribution (see Section 2.7 for a precise statement), then NP can be solved in polynomial time in the average case.

Theorem 2 (Average-Case Hardness of nK^t)**.** *The following holds.*

$$(\mathsf{MINnKT}, \mathcal{U}) \in \mathsf{AvgBPP} \implies \mathsf{DistNP} \subseteq \mathsf{AvgBPP}.$$

It is instructive to compare Theorem 1 and Theorem 2 with some related results from [Hir20a, Hir20c, Hir22c, LP22, HIR23]. Let $\mathsf{MINcKT}^{\mathsf{SAT}}$ denote the SAT-oracle version of conditional MINKT. The following results are known.

[1] While the definition of the measure presented in these papers is slightly different than the one given here, it is not hard to see that they are equivalent.

(*i*) Worst-Case Hardness: $\mathsf{MINcKT}^{\mathsf{SAT}} \in \mathsf{BPP} \iff \mathsf{NP} \subseteq \mathsf{BPP}$ [Hir20c].[2] Moreover, computing sublinear conditional time-bounded Kolmogorov complexity is NP-hard [Hir22c, LP22], while computing conditional time-bounded Kolmogorov complexity is NP-hard under a strong cryptographic assumption [HIR23].

(*ii*) Average-Case Hardness: $\mathsf{MINKT}^{\mathsf{SAT}} \in \mathsf{BPP} \implies \mathsf{DistNP} \subseteq \mathsf{HeurBPP}$ [Hir20c, Theorem 3.5].

(*iii*) PH Average-Case Hardness: $\mathsf{GapMINKT}^{\mathsf{PH}} \in \mathsf{P} \implies \mathsf{DistPH} \subseteq \mathsf{AvgP}$ [Hir20a].

In comparison with the results in Item (*i*), Theorem 1 exhibits the hardness of a *non-conditional* meta-computational problem, i.e., for instances without a conditional string. On the other hand, in comparison with Item (*ii*), our proof of Theorem 2 establishes a stronger conclusion under a weaker assumption, as explained next. For the DistNP-hardness of $\mathsf{MINKT}^{\mathsf{SAT}}$ stated in Item (*ii*), it is enough to have an efficient algorithm that accepts the set of strings with low $\mathsf{K}^{\mathsf{poly},\mathsf{NP}}$-complexity while rejecting a large fraction of random strings. In contrast, in Theorem 2 we only need an efficient algorithm that accepts strings with low $\mathsf{nK}^{\mathsf{poly}}$-complexity while rejecting a large fraction of random strings. Since the set of strings with low $\mathsf{nK}^{\mathsf{poly}}$-complexity is a subset of those with low $\mathsf{K}^{\mathsf{poly},\mathsf{NP}}$-complexity, the actual assumption required in Theorem 2 is implied by the one needed in Item (*ii*). Moreover, in Theorem 2 we obtain the stronger consequence that NP is average-case easy in the errorless setting, rather than in the error-prone setting considered in Item (*ii*). Finally, in Item (*iii*), [Hir20a] shows the DistPH-hardness of a generalization of a gap formulation of MINKT that considers compression with access to a PH-oracle. While it is possible to use a weaker assumption to derive the weaker conclusion $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, similarly to the discussion about Item (*ii*) the corresponding meta-computational problem is harder than the one in the assumption of Theorem 2 (see [Hir20a, Theorem 1.16]).

In our next result, we show that if MINnKT is average-case easy under the uniform distribution, then NP can be solved in sub-exponential time in the worst case.

Theorem 3 (Worst-Case to Average-Case Reduction)**.** *The following holds.*

$$(\mathsf{MINnKT}, \mathcal{U}) \in \mathsf{AvgBPP} \implies \mathsf{NP} \subseteq \mathsf{RTIME}\left[2^{O(n/\log n)}\right].$$

Previous works [Hir21, GKLO22a] have shown that if Σ_2^{P} is average-case easy, then NP can be solved in time $2^{O(n/\log n)}$. Theorem 3 obtains the same conclusion under the weaker assumption that MINnKT is average-case easy. Indeed, MINnKT can be solved in NP using two non-adaptive calls to an NP oracle, so it is in $\mathsf{NP}^{\mathsf{NP}} = \Sigma_2^{\mathsf{P}}$. Hence, if Σ_2^{P} is easy on average, so is MINnKT. However, the reverse direction remains unclear. Also, note that if MINnKT is average-case easy, then by Theorem 2, the same holds for NP. Yet, it is unknown if Σ_2^{P} is average-case easy under the assumption that NP is average-case easy.

1.2.2 Probabilistic Nondeterministic Kolmogorov Complexity Is (Essentially) All You Need

The proof of Theorem 1 relies on certain useful properties of a probabilistic variant of nK^t called pnK^t. First, we review the following definition, which will be useful when comparing our results with prior work. For $x \in \{0,1\}^*$ and $t \in \mathbb{N}$, the *t-time-bounded randomized nondeterministic Kolmogorov complexity of* x is defined as

$$\mathsf{rnK}^t(x) := \min_{\text{program } \Pi} \left\{ |\Pi| \;\middle|\; \Pr_{r \sim \{0,1\}^t} \begin{bmatrix} \forall w \in \{0,1\}^t, \Pi(w;r) \text{ outputs } x \text{ or } \perp \text{ in } t \text{ steps}; \\ \exists w \in \{0,1\}^t, \Pi(w;r) \text{ outputs } x \text{ in } t \text{ steps} \end{bmatrix} \geq \frac{2}{3} \right\}.$$

The complexity measure rnK^t was studied under the name CAM^t in [BLvM05].

[2] In fact, [Hir20c] showed the stronger result that P^{NP} reduces to $\mathsf{MINcKT}^{\mathsf{SAT}}$ under deterministic polynomial-time reductions.

pnK^t **Complexity.** Inspired by pK^t complexity [GKLO22a] and its properties (see, e.g., [LO22]), we introduce the following notion of probabilistic nondeterministic Kolmogorov complexity. For $x \in \{0,1\}^*$ and $t \in \mathbb{N}$, the *t-time-bounded probabilistic nondeterministic Kolmogorov complexity of x* is defined as

$$\mathsf{pnK}^t(x) := \min\left\{ k \in \mathbb{N} \;\middle|\; \Pr_{r\sim\{0,1\}^t}\begin{bmatrix}\exists \text{ program } \Pi \text{ of size at most } k \text{ s.t. the following conditions hold:}\\ \forall w \in \{0,1\}^t, \Pi(w;r) \text{ outputs } x \text{ or } \perp \text{ in } t \text{ steps;}\\ \exists w \in \{0,1\}^t, \Pi(w;r) \text{ outputs } x \text{ in } t \text{ steps}\end{bmatrix} \geq \frac{2}{3}\right\}.$$

Equivalently,

$$\mathsf{pnK}^t(x) := \min\left\{ k \in \mathbb{N} \;\middle|\; \Pr_{r\sim\{0,1\}^t}\left[\mathsf{nK}^t(x \mid r) \leq k\right] \geq \frac{2}{3}\right\}.$$

In other words, while pK^t corresponds to K^t complexity in the presence of a random string, analogously, pnK^t is simply nK^t complexity in the presence of a random string.

We extend the definitions of rnK^t and pnK^t to their corresponding oracle versions in the natural way, i.e., by giving to the program oracle access to the corresponding set $\mathcal{O} \subseteq \{0,1\}^*$. We write $\mathsf{rnK}^{t,\mathcal{O}}$ and $\mathsf{pnK}^{t,\mathcal{O}}$, respectively.

We show that pnK^t admits nearly all key properties of (time-unbounded) Kolmogorov complexity, such as a general form of language compression, (conditional) coding, and useful variants of symmetry of information. As a consequence, in many scenarios, pnK^t can replace Kolmogorov complexity while still providing useful bounds on the complexity of computations.

Below we discuss the properties of pnK^t in more detail and contrast our results with prior work.

Language Compression for pnK^t. As established by [Lee06, Theorem 4.1.4] (improving [BLvM05]), nondeterminism allows us to get non-trivial language compression in the time-bounded setting. In other words, there is a polynomial $p(\cdot)$ such that for every set $A := \{A_n \subseteq \{0,1\}^n\}_{n\in\mathbb{N}}$, and for all $n \in \mathbb{N}$ and $x \in A_n$,

$$\mathsf{nK}^{p(n),A_n}(x) \leq \log|A_n| + O(\sqrt{\log|A_n|}\cdot\log n).$$

[BLvM05] showed that an improved language compression theorem holds if we employ randomness in addition to nondeterminism. More precisely, there is a polynomial $p(\cdot)$ such that for every set $A := \{A_n \subseteq \{0,1\}^n\}_{n\in\mathbb{N}}$, and for all $n \in \mathbb{N}$ and $x \in A_n$,

$$\mathsf{rnK}^{p(n),A_n}(x) \leq \log|A_n| + \log^3 p(n).$$

This result is still suboptimal due to the additive $O(\log^3 n)$ term. This super-logarithmic overhead can lead to super-polynomial running times in applications.

In contrast to the aforementioned results, we observe that an optimal language compression theorem holds for pnK^t. This result plays a key role in the proof of Theorem 1.

Theorem 4 (Language Compression for pnK^t)**.** *There is a polynomial $p_{\mathsf{LC}}(\cdot)$ such that for every set $A := \{A_n \subseteq \{0,1\}^n\}_{n\in\mathbb{N}}$, and for all $n \in \mathbb{N}$ and $x \in A_n$,*

$$\mathsf{pnK}^{p_{\mathsf{LC}}(n),A_n}(x) \leq \log|A_n| + \log p_{\mathsf{LC}}(n).$$

On the other hand, we provide evidence that nondeterminism is indeed crucial for language compression, even when restricted to predicates in P.

Theorem 5. *Item 1 implies Item 2 in the statements appearing below.*

(*i*) (Average-Case pK^t Language Compression for P.) *For any set* $A := \{A_n \subseteq \{0,1\}^n\}_{n\in\mathbb{N}}$ *that is in* P*, there exists a constant* $c > 0$ *such that for all* $n \in \mathbb{N}$

$$\Pr_{x\sim A_n}\left[\mathsf{pK}^{n^c}(x) \leq \log|A_n| + c\cdot\log n\right] \geq \frac{1}{n^c}.$$

(*ii*) $\mathsf{Unary\text{-}NP} \subseteq \mathsf{BPP}$.

Theorem 5 complements a result from [BLvM05] showing that for all $n, t, k \in \mathbb{N}$ such that $k \leq n$, there exists a set A such that $\log|A_n| = k$ and for every $x \in A_n$,

$$\mathsf{rK}^{t,A_n}(x) \geq n - \log|A_n| - \log t - O(1).$$

This bound on rK^{t,A_n} complexity rules out the possibility of getting a language compression theorem without nondeterminism for *every* set A, while Theorem 5 rules out language compression without nondeterminism (even on average) for sets decidable in polynomial time (under the assumption that $\mathsf{Unary\text{-}NP} \not\subseteq \mathsf{BPP}$). The latter case is of interest in the proof of Theorem 1.

Conditional Coding for pnK^t. Since pnK^t is a more "powerful" notion than pK^t, and the latter admits a coding theorem [LOZ22], it follows that pnK^t also has this property. Here, we show that pnK^t possesses an even stronger *conditional* coding property.

Theorem 6 (Conditional Coding for pnK^t)**.** *For every polynomial-time samplable distribution family* $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$ *supported over* $\{0,1\}^n \times \{0,1\}^n$*, there exists a polynomial* p *such that for all* $n \in \mathbb{N}$*,*

$$\mathsf{pnK}^{p(n)}(x \mid y) \leq \log\frac{1}{\mathcal{D}_n(x \mid y)} + \log p(n).$$

In contrast, conditional coding does not hold for pK^t under the assumption that $\mathsf{NP} \not\subseteq \mathsf{BPP}$ [HIL+23]. This gives another natural setting where nondeterminism seems essential and leads to an advantage of pnK^t over pK^t.

Symmetry of Information for pnK^t. Finally, we show that non-trivial forms of the symmetry of information principle hold for pnK^t complexity. The results discussed below improve the parameters of analogous results established for rnK^t complexity in [LR05] (see also [Lee06, Theorem 6.3.3]).

Theorem 7 (Semi-Symmetry of Information for pnK^t)**.** *There exist polynomials* $p_{\mathsf{Sol}}(\cdot)$ *and* $p_0(\cdot)$ *such that for all sufficiently large* $x, y \in \{0,1\}^*$ *and every* $t \geq p_0(|x| + |y|)$*,*

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

The reason we call Theorem 7 semi-symmetry of information is because on the left-hand side of the inequality we obtain $\mathsf{pK}^t(x,y)$ as opposed to $\mathsf{pnK}^t(x,y)$. We are able to overcome this limitation in the average-case setting, which is sufficient in many applications.

Theorem 8 (Average-Case Symmetry of Information for pnK^t)**.** *For every polynomial-time samplable distribution family* $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$ *supported over* $\{0,1\}^n \times \{0,1\}^n$*, and for every polynomial* q*, there exists a polynomial* p *such that for every computable time bound* $t \geq p(n)$ *and for all large enough* n*,*

$$\Pr_{(x,y)\sim\mathcal{D}_n}\left[\mathsf{pnK}^t(x,y) \geq \mathsf{pnK}^t(y \mid x) + \mathsf{pnK}^t(x) - c\cdot\log t\right] \geq 1 - \frac{1}{q(n)},$$

where $c > 0$ *is a universal constant.*

In contrast, average-case symmetry of information does not hold for pK^t under the existence of one-way functions [HIL+23].

To sum up, pnK^t satisfies nearly all pillars of time-unbounded Kolmogorov complexity (as discussed in [Lee06]), which makes the new measure attractive in applications of meta-complexity in time-bounded settings. The only missing property is *worst-case* symmetry of information, which we leave as an interesting open problem. It is worth pointing out that there is an oracle world where symmetry of information for nK^t does not hold (see [LR05, Corollary 1] and [Lee06, Corollary 6.5.2]).

Using a straightforward adaptation of the proof of Theorem 1, we also establish the hardness of MINpnKT, the problem of computing pnK^t complexity (we refer to Appendix C for the precise definition of this problem and the minor modifications needed in the proof).

Theorem 9 (Characterizing $\mathsf{NP} \not\subseteq \mathsf{BPP}$ by the Worst-Case Hardness of pnK^t)**.** *The following holds.*

$$\mathsf{MINpnKT} \in \mathsf{prBPP} \iff \mathsf{NP} \subseteq \mathsf{BPP}.$$

Similarly to Theorem 1, Theorem 9 requires the use of a computational model and encodings for which the composition of programs does not yield a significant overhead in running time and program length, as explained in Section 1.3 and Section 2.2.

1.2.3 Gap Problems in Meta-Complexity and Connections to Average-Case Complexity

In this section, we elaborate on some aspects of our hardness results and their connections to average-case complexity. In more detail, we discuss a striking phenomenon where a mild extension of Theorem 9 would have a significant impact on our understanding of the computational complexity of the polynomial hierarchy.

Related Work. [Hir22c] shows that if NP is average-case easy, then a specific gap version of computing *conditional* K^t is also easy. More precisely, given $(x, y, 1^s, 1^t)$, one can efficiently decide whether $\mathsf{K}^t(y \mid x) \leq s$ or $\mathsf{K}^{\mathsf{poly}(t,|x|)}(y \mid x) > s + O(\log(t+|x|)) + \mathsf{K}^t(x) - \mathsf{K}^{\mathsf{poly}(t,|x|)}(x)$. Thus, proving the NP-hardness of this problem would suffice to rule out Heuristica. However, the gap in this problem involves the quantity $\mathsf{K}^t(x) - \mathsf{K}^{\mathsf{poly}(t,|x|)}(x)$ (known as the computational depth of x), which can be large. This makes it more challenging to establish the NP-hardness of this class of instances. In fact, there remains a significant gap between existing NP-hardness results for computing conditional K^t and what is needed to exclude Heuristica (see [HIR23] for a detailed discussion).

Shifting focus to the *non-conditional* case, if NP is average-case easy, then a simpler gap version of computing K^t is worst-case easy: given $(x, 1^s, 1^t)$, one can efficiently decide whether $\mathsf{K}^t(x) \leq s$ or $\mathsf{K}^{\mathsf{poly}(t)}(x) > s + O(\log t)$ [Hir18, Hir20b]. Thus, showing that this "small-gap" version (known as Gap-MINKT) is NP-hard would also rule out Heuristica. However, currently we do not know even if MINKT is hard, let alone Gap-MINKT.

We also mention related results from [Hir22b], which show that obtaining a significantly better hardness result for a meta-computational problem called MINLT is sufficient to rule out Heuristica.

Theorem 1 and Theorem 9 make progress on this front by presenting non-conditional meta-computational problems whose hardness characterize $\mathsf{NP} \not\subseteq \mathsf{BPP}$. Furthermore, existing worst-case to average-case reduction techniques can be adapted to the setting of MINpnKT. As explained below, these results bear an intriguing connection to the longstanding problem of excluding PH-Heuristica, i.e., showing that if

$\mathsf{DistPH} \subseteq \mathsf{AvgBPP}$ then $\mathsf{PH} \subseteq \mathsf{BPP}$ (see, e.g., [Hir20a] for more information about this problem).[3]

Consider the following statements:

(*i*) If $\mathsf{MINpnKT} \in \mathsf{prBPP}$ then $\mathsf{NP} \subseteq \mathsf{BPP}$ (Theorem 9).

(*ii*) If $(\mathsf{MINnKT}, \mathcal{U}) \in \mathsf{AvgBPP}$ then $\mathsf{Gap\text{-}MINpnKT} \in \mathsf{prBPP}$ (see Proposition 58 in Appendix B).

Recall that, as alluded to above, we have $\mathsf{MINnKT} \in \Sigma_2^{\mathsf{P}}$. As a consequence, under the assumption that $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$, it follows from Item (*ii*) that $\mathsf{Gap\text{-}MINpKT} \in \mathsf{prBPP}$. Therefore, if we could replace $\mathsf{MINpnKT}$ with $\mathsf{Gap\text{-}MINpnKT}$ in Item (*i*) (or replace $\mathsf{Gap\text{-}MINpnKT}$ with $\mathsf{MINpnKT}$ in Item (*ii*)), it would follow that $\mathsf{NP} \subseteq \mathsf{BPP}$ (and consequently $\mathsf{PH} \subseteq \mathsf{BPP}$). In other words, the seemingly minor distinction between $\mathsf{MINpnKT}$ and $\mathsf{Gap\text{-}MINpnKT}$ is currently the only issue preventing us from excluding PH-Heuristica by showing that if $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$ then $\mathsf{PH} \subseteq \mathsf{BPP}$ (see Corollary 59 in Appendix B).

Interpretation. There are two ways of interpreting these results. An optimistic perspective on the scenario described above is that we are close to excluding PH-Heuristica, since this merely requires the hardness of $\mathsf{Gap\text{-}MINpnKT}$. In other words, under this perspective, the hope is that $\mathsf{MINpnKT}$ might be similar to $\mathsf{Gap\text{-}MINpnKT}$, even if our current techniques distinguish these two problems. Moreover, a few existing hardness results, such as [Hir22b], extend to the gap version of the problem, perhaps indicating that a new idea or different proof of Theorem 9 might suffice to generalize the existing result. On the other hand, a pessimistic interpretation is that we remain far from excluding PH-Heuristica, and our results and techniques simply suggest a fundamental distinction between establishing the hardness of MINnKT and $\mathsf{MINpnKT}$ compared with the variants of these problems with a small gap. In particular, under this interpretation, our hardness results shed light on why existing worst-case to average-case reductions in meta-complexity inherently produce a gap between positive and negative instances.

While we are currently unable to exclude PH-Heuristica by establishing the hardness of computing nondeterministic Kolmogorov complexity with a small gap, we make progress by showing that the *conditional* version of $\mathsf{Gap\text{-}MINnKT}$ is hard. More precisely, for $\tau\colon \mathbb{N} \to \mathbb{N}$, let $\mathsf{Gap}_\tau\text{-}\mathsf{Cond\text{-}MINnKT}$ be the following promise problem: Given $(x, y, 1^s, 1^t)$, where $x, y \in \{0,1\}^*$ and $s, t \in \mathbb{N}$, decide whether $\mathsf{nK}^t(y \mid x) \leq s$ or $\mathsf{nK}^{\tau(t,|x|)}(y \mid x) > s + \log \tau(t + |x|)$. We say that $\mathsf{Gap\text{-}Cond\text{-}MINnKT} \in \mathsf{prBPP}$ if there is a probabilistic polynomial-time algorithm that solves $\mathsf{Gap}_\tau\text{-}\mathsf{Cond\text{-}MINnKT}$ for some polynomial τ.

Theorem 10. *The following holds.*

$$\mathsf{Gap\text{-}Cond\text{-}MINnKT} \in \mathsf{prBPP} \iff \mathsf{NP} \subseteq \mathsf{BPP}.$$

Thus, as a corollary of our results (Theorem 1 and Theorem 10), we relate the complexities of MINnKT and $\mathsf{Gap\text{-}Cond\text{-}MINnKT}$, exhibiting an interesting link between the non-conditional version and the conditional version. Formally, it is possible to establish the following slightly more general equivalence.

Corollary 11. *The following holds.*

$$\mathsf{MINnKT} \in \mathsf{BPP} \iff \mathsf{Cond\text{-}MINnKT} \in \mathsf{BPP} \iff \mathsf{Gap\text{-}Cond\text{-}MINnKT} \in \mathsf{prBPP}.$$

[3] Recall that $\mathsf{PH} \subseteq \mathsf{BPP}$ if and only if $\mathsf{NP} \subseteq \mathsf{BPP}$. Consequently, excluding PH-Heuristica is a necessary step toward excluding Heuristica (i.e., proving that if $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$ then $\mathsf{NP} \subseteq \mathsf{BPP}$).

1.3 Techniques

In this section, we describe some ideas employed in the proof of our results. We primarily focus on the proof that if $\mathsf{MINnKT} \in \mathsf{BPP}$ then $\mathsf{NP} \subseteq \mathsf{BPP}$ (Theorem 1). Beyond being our main contribution, its proof provides insights into the techniques employed in other parts of the paper and highlights the usefulness of pnK^t complexity. (For technical reasons, we will introduce and use a further complexity measure in the proof, as described later in this section.)

In order to establish the hardness of MINnKT, we explore different properties of nondeterminism Kolmogorov complexity and its probabilistic extension. The first component of the argument is the optimal language compression theorem for pnK^t (Theorem 4).

1. Language compression for pnK^t. We provide two proofs that pnK^t admits language compression with optimal parameters. Our first discovered proof, which relies on techniques from [BLvM05] and [Hir21], served as the inspiration for the definition of pnK^t complexity. This proof appears in Section 3.2. Below we present a different, simpler argument that nicely illustrates the benefits of combining *randomness* and *nondeterminism* for string compression. A similar technique is used in [Sip83] and [BFL01].

Let $A_n \subseteq \{0,1\}^n$, $x \in A_n$, and set $k = \lceil \log |A_n| \rceil + 2$. We employ an efficiently computable family $H \subseteq \{0,1\}^n \to \{0,1\}^k$ of hash functions such that $\mathbf{Pr}_{h \sim H}[h(x) = h(y)] \leq 2^{-k}$ for every $y \in A_n \setminus \{x\}$. By a union bound and our choice of k,

$$\mathop{\mathbf{Pr}}_{h \sim H}[\exists y \in A_n \setminus \{x\} \text{ s.t. } h(x) = h(y)] \leq 2^{-k} \cdot |A_n| \leq \frac{1}{4}. \tag{1}$$

The idea is that, for a randomly sampled $h \sim H$ (corresponding to the *randomness* in our pnK^t description), we can store the integers n and k and the value $h(x) \in \{0,1\}^k$ in the program (which has access to h) using $k + O(\log n)$ bits, then use *nondeterminism* to guess x. In more detail, let v be the hash value stored in the description, and let x' be a nondeterministic input string of length n. If $h(x') = v$ and $x' \in A_n$, we output x'; otherwise we output $\perp$. By Equation (1), with probability at least $3/4$, we sample a good h such that the value $h(x)$ is *unique* among the elements in A_n. This shows that, for a good choice of h, there is an efficient nondeterministic program of length at most $k + O(\log n) = \log |A_n| + O(\log n)$ that (*i*) outputs x on at least one computation path, and (*ii*) outputs either x or $\perp$ on every computation path.

It follows from the argument described above that there is a polynomial $p_{\mathsf{LC}}(\cdot)$ such that, for every set $A := \{A_n \subseteq \{0,1\}^n\}_{n \in \mathbb{N}}$ and for every $n \in \mathbb{N}$ and $w \in A_n$,

$$\mathsf{pnK}^{p_{\mathsf{LC}}(n), A_n}(w) \leq \log |A_n| + \log p_{\mathsf{LC}}(n), \tag{2}$$

as desired.

Next, we explain the connection between language compression and the complexity of NP employed in the proof of Theorem 1. The argument is inspired by ideas from [AFPS12] (see also [HIL+23] for an equivalence between compression and the easiness of NP).

2. Compressing witnesses and the easiness of NP. Let $L \in \mathsf{NP}$. Fix a verifier $V(\cdot,\cdot)$ for L. Let $x \in L$, where $|x| = n$, and consider the set $W^x = \{w \in \{0,1\}^m \mid V(x,w) = 1\}$, where $m = \mathsf{poly}(n)$. Note that, by language compression (Equation (2)) applied to the set W^x, for every $w \in W^x$,

$$\mathsf{pnK}^{\mathsf{poly}(n), W^x}(w) \leq \log |W^x| + O(\log n).$$

Moreover, since given x the set W^x can be decided in polynomial time, we have

$$\mathsf{pnK}^{\mathsf{poly}(n)}(w \mid x) \leq \log |W^x| + O(\log n). \tag{3}$$

Due to the use of the nondeterministic measure $\mathsf{pnK}^{\mathsf{poly}(n)}$, this bound is not yet sufficient to allow us to efficiently find a witness $w \in W^x$ given x. However, let us suppose for a moment that, under the assumption that $\mathsf{MINnKT} \in \mathsf{BPP}$, we could show that for every pair (y, x) of strings of length $\mathsf{poly}(n)$ the following holds:

$$\mathsf{pK}^{\mathsf{poly}(t)}(y \mid x) \leq \mathsf{pnK}^t(y \mid x) + \lambda(n), \tag{4}$$

for some function $\lambda(n)$.[4] In other words, assume that conditional pnK^t complexity collapses to conditional $\mathsf{pK}^{\mathsf{poly}(t)}$ with an additive overhead term $\lambda(n)$.

Under the hypothesis that Equation (4) holds, we obtain together with Equation (3) that

$$\mathsf{pK}^{\mathsf{poly}(n)}(w \mid x) \leq \log |W^x| + O(\log n) + \lambda(n) \tag{5}$$

for every $w \in W^x$. Roughly speaking, this means that every witness $w \in W^x$ can be compressed to just $v(n) := \log |W^x| + O(\log n) + \lambda(n)$ bits. Since each witness $w \in W^x$ admits its own compressed representation of this length, it is not hard to show that given x one can find a witness $w \in W^x$ with probability at least $1/(\mathsf{poly}(n) \cdot 2^{\lambda(n)})$ simply by running a randomly chosen program of length at most $v(n)$ for $\mathsf{poly}(n)$ steps (see Lemma 32). Consequently, repeating this procedure $\mathsf{poly}(n) \cdot 2^{\lambda(n)}$ times provides an algorithm that finds a witness for x with high probability, whenever one exists. This shows that $L \in \mathsf{BPP}$ if we can guarantee that $\lambda(n) = O(\log n)$.

Note that it is crucial for this argument to work that we obtain an upper bound on $\mathsf{pK}^{\mathsf{poly}(n)}$ complexity in Equation (5) instead of $\mathsf{pnK}^{\mathsf{poly}(n)}$ complexity. The latter does not provide an efficient algorithm that allows us to recover a string from its (nondeterministic) compressed representation.

Given the discussion above, in order to show that $\mathsf{NP} \subseteq \mathsf{BPP}$ it is sufficient to establish Equation (4) with $\lambda(n) = O(\log n)$, under the assumption that $\mathsf{MINnKT} \in \mathsf{BPP}$.

3. Easiness assumptions and collapses in time-bounded Kolmogorov complexity. It will be instructive to compare how different easiness assumptions allow us to relate pK and pnK complexities. For any large enough $t = \mathsf{poly}(n)$, the following implications can be established via a careful adaptation of existing techniques from meta-complexity to the setting of nondeterministic Kolmogorov complexity (see Lemma 55 and Lemma 52):

$$\begin{aligned} \mathsf{Gap\text{-}Cond\text{-}MINnKT} \in \mathsf{prBPP} \quad &\Longrightarrow \quad \mathsf{pK}^{\mathsf{poly}(t)}(y \mid x) \leq \mathsf{pnK}^t(y \mid x) + O(\log n). \\ \mathsf{Gap\text{-}MINnKT} \in \mathsf{prBPP} \quad &\Longrightarrow \quad \mathsf{pK}^{\mathsf{poly}(t)}(y \mid x) \leq \mathsf{pnK}^t(y \mid x) + \mathsf{pK}^t(x) - \mathsf{pK}^{\mathsf{poly}(t)}(x) + O(\log n). \end{aligned}$$

The conclusion in the first implication is precisely what we would like to show in Equation (4). However, it assumes the easiness of the problem of estimating *conditional* nondeterministic Kolmogorov complexity. On the other hand, while the assumption in the second implication is sufficient for our purposes (as it is implied by $\mathsf{MINnKT} \in \mathsf{BPP}$), the inequality in the conclusion has the extra term $\mathsf{pK}^t(x) - \mathsf{pK}^{\mathsf{poly}(t)}(x)$, known as the *time-bounded computational depth*. This quantity might be significantly larger than $O(\log n)$ for some strings x and choices of t. On the other hand, it is known to be at most $O(\log n)$ for an average string x, and at most $O(n/\log n)$ for every x for some moderately bounded choice of t. We remark that this notion of computational depth, with a polynomial gap in the time bounds, appears in many prior works in meta-complexity and has been an infamous obstacle that forces results to hold only in the average-case setting or leads to suboptimal worst-case running times of the form $2^{O(n/\log n)}$ rather than polynomial time (see, e.g., [Hir21, CHV22, GKLO22b, HKLO24]).

[4] Recall that $\mathsf{pK}^t(y)$ is the minimum k such that with probability at least $2/3$ over the choice of a random string $r \in \{0,1\}^t$, we have $\mathsf{K}^t(y \mid r) \leq k$ [LO22].

In order to achieve our goal, an important idea is to aim instead for an implication of the following form:

$$\mathsf{MINnKT} \in \mathsf{BPP} \implies \mathsf{pK}^{\mathsf{poly}(t)}(y \mid x) \leq \mathsf{pnK}^{t}(y \mid x) + \mathsf{pnK}^{t}(x) - \mathsf{pnK}^{t+p(n)}(x) + O(\log n), \quad (6)$$

where $p(n)$ is a polynomial independent of t. As before, assuming Equation (6), we can conclude the proof if we could show that $\mathsf{pnK}^{t}(x) - \mathsf{pnK}^{t+p(n)}(x) = O(\log n)$. While this might not hold for a particular value of t, it is possible to argue via a telescoping sum trick that this bound does hold for some t' that is not too large.[5] In more detail, for any fixed time bound t, let $t_i = t + i \cdot p(n)$, where $i \in \{0, 1, \ldots, n\}$. First, note that $\mathsf{pnK}^{t_0}(x) - \mathsf{pnK}^{t_n}(x) \leq 2n$, since any n-bit string x can be efficiently described with at most $2n$ bits. Therefore,

$$\sum_{i=0}^{n-1} \left(\mathsf{pnK}^{t_i}(x) - \mathsf{pnK}^{t_{i+1}}(x)\right) = \mathsf{pnK}^{t_0}(x) - \mathsf{pnK}^{t_n}(x) \leq 2n.$$

Consequently, there is an index i such that $\mathsf{pnK}^{t_i}(x) - \mathsf{pnK}^{t_{i+1}}(x) \leq 2$. In particular, for some time bound $t' \leq t + n \cdot p(n)$, we have $\mathsf{pnK}^{t'}(x) - \mathsf{pnK}^{t'+p(n)}(x) = O(\log n)$. It turns out that this is sufficient to establish Equation (4) with $\lambda(n) = O(\log n)$ under the assumption that $\mathsf{MINnKT} \in \mathsf{BPP}$, as desired.[6]

Unfortunately, it is not clear how to prove Equation (6) exactly. A key difficulty is that in many arguments involving meta-complexity, we typically make use of some form of time-bounded symmetry of information (SoI). However, for the highly efficient bounds we aim to achieve, we need symmetry of information with an *additive* polynomial overhead in the running time of the form $t + p(n)$, as opposed to an overhead of the form $\mathsf{poly}(t)$ (see, e.g., [Hir22c, GK22, HKLO24] and the weak symmetry of information for pK^t from [Ila23]). It is unclear to us how to establish such a result.

To illustrate the issue, let us briefly recall the approach of [Hir22c, GK22, HKLO24], which shows symmetry of information for pK^t under the assumption that MINKT is easy. Concretely,

$$\mathsf{pK}^{t}(x, y) \gtrsim \mathsf{pK}^{\mathsf{poly}(t)}(x) + \mathsf{pK}^{\mathsf{poly}(t)}(y \mid x).$$

Note the $\mathsf{poly}(t)$ time bound on the right-hand side of the inequality, particularly for the term $\mathsf{pK}^{\mathsf{poly}(t)}(x)$.

The argument compares the following three distributions over strings:

$$\begin{aligned}
\mathcal{D}_1 &:= \mathsf{DP}_k(x, \mathcal{U}_{nk}) \circ \mathsf{DP}_{k'}(y, \mathcal{U}_{nk'}) \circ \mathcal{U}_t, \\
\mathcal{D}_2 &:= \mathsf{DP}_k(x, \mathcal{U}_{nk}) \circ \mathcal{U}_{nk'+k'} \circ \mathcal{U}_t, \\
\mathcal{D}_3 &:= \mathcal{U}_{nk+k} \circ \mathcal{U}_{nk'+k'} \circ \mathcal{U}_t,
\end{aligned}$$

where $\mathsf{DP}_k \colon \{0,1\}^n \times \{0,1\}^{nk} \to \{0,1\}^{nk+k}$ is the (k-wise) direct product generator [Hir21] (see Definition 21), and $\mathcal{U}_m$ denotes the uniform distribution over $\{0,1\}^m$. One can then argue that if the K^t-complexities of these distributions are close, it follows that

$$\mathsf{pK}^{t}(x, y) \gtrsim k + k'.$$

(We omit the details here; see [HKLO24, Appendix A] for a full exposition.)

A useful property of the direct product generator is that if one can distinguish its output instantiated with a string x, namely $\mathsf{DP}_k(x, \mathcal{U}_{nk})$, from the uniform distribution $\mathcal{U}_{nk+k}$, then one can efficiently *reconstruct* x with only k bits of advice. By choosing $k \approx \mathsf{pK}^{\mathsf{poly}(t)}(x)$ and $k' \approx \mathsf{pK}^{\mathsf{poly}(t)}(y \mid x)$ for the above

[5] This idea is not new and a similar trick was employed by [Hir21] and subsequent papers. A key difference is that we are able to explore *additive* overheads in the running time, as opposed to polynomial and quasi-linear overheads.

[6] The benefits of obtaining improved bounds on the running time in applications of time-bounded Kolmogorov complexity have also been explored in [CHV22].

distributions, one can argue that, with high probability, the K^t-complexities of these distributions are indeed close. Otherwise, an efficient algorithm for MINKT would serve as a distinguisher, and the reconstruction property of the direct product generator would yield either $\mathsf{pK}^{\mathsf{poly}(t)}(x) \lesssim k$ or $\mathsf{pK}^{\mathsf{poly}(t)}(y \mid x) \lesssim k'$, contradicting the definitions of k and k'. Note that, since the reconstruction takes time $\mathsf{poly}(t)$, the $\mathsf{poly}(t)$ time bound in the choice $k \approx \mathsf{pK}^{\mathsf{poly}(t)}(x)$ seems unavoidable in this argument.

By adapting the above proof of symmetry of information with the assumption that MINnKT is easy, one can show that $\mathsf{pnK}^t(x, y) \gtrsim \mathsf{pK}^{\mathsf{poly}(t)}(x) + \mathsf{pK}^{\mathsf{poly}(t)}(y \mid x)$, which can then be used to obtain a weaker version of Equation (6) (see also Lemma 52):

$$\mathsf{MINnKT} \in \mathsf{BPP} \implies \mathsf{pK}^{\mathsf{poly}(t)}(y \mid x) \leq \mathsf{pnK}^t(y \mid x) + \mathsf{pK}^t(x) - \mathsf{pK}^{\mathsf{poly}(t)}(x) + O(\log n).$$

However, the polynomial time overhead remains inherent.

A potentially useful observation in this context is that, instead of the standard formulation of SoI, it is possible to work directly with a weaker variant—referred to as weak SoI—in which the SoI inequality holds for most choices of one of the strings. This approach can be used to avoid the overhead associated with the direct-product reconstruction of x. That said, existing proofs of weak SoI still seem to suffer from essentially the same overhead issue, typically stemming from the use of coding theorems or related arguments (see [Ila23]). Moreover, even an overhead of the form $O(t)$ is insufficient in our setting, which makes it particularly challenging.

Crucially, by defining a new notion of time-bounded Kolmogorov complexity, we manage to obtain an SoI-style result without incurring a large overhead in the time bound, which suffices for our purpose.

4. A new measure of complexity that achieves additive time overhead. We now give a simplified overview of our new approach, which overcomes this barrier and yields only an additive $\mathsf{poly}(n)$ overhead in the time bound. The key idea is to introduce a new complexity measure called ℓ-nK^t. Given a time bound t and a parameter $\ell \in \mathbb{N}$, we define ℓ-$\mathsf{nK}^t(x)$ as

$$\ell\text{-}\mathsf{nK}^t(x) = \min\ s \text{ s.t. } \Pr_{r \sim \mathcal{U}_\ell}\left[\mathsf{nK}^t(x \circ r) \leq s + \ell\right] \geq \frac{2}{3},$$

where $x \circ r$ denotes the concatenation of the strings x and r. In a sense, ℓ-$\mathsf{nK}^t(x)$ is related to $\mathsf{pnK}^t(x)$: while $\mathsf{pnK}^t(x)$ captures the nK^t-complexity of x *conditioned* on a random string, ℓ-$\mathsf{nK}^t(x)$ captures the nK^t-complexity of x *concatenated* with a random string of length ℓ.

First, one aspect of our new approach that helps avoid the undesired polynomial overhead in the time bound is that it avoids invoking the direct-product reconstruction of x. More specifically, we define two distributions of strings:

$$\mathcal{D}_1 := x \circ \mathsf{DP}_k(y, \mathcal{U}_{nk}) \circ \mathcal{U}_\ell,$$
$$\mathcal{D}_2 := x \circ \mathcal{U}_{nk+k} \circ \mathcal{U}_\ell,$$

where $\ell \geq \mathsf{poly}(n)$ is a parameter smaller than t. Note that, unlike in the earlier approach, we do not apply the direct product generator to x here.

Next, by unpacking the definition of ℓ-nK, we can upper bound the nK^t-complexity of the first distribution as follows:

$$\mathsf{nK}^t(x \circ \mathsf{DP}_k(y, \mathcal{U}_{nk}) \circ \mathcal{U}_\ell) \lesssim (\ell + nk)\text{-}\mathsf{nK}^{t - \mathsf{poly}(n)}(x \circ y) + \ell + nk.$$

Moreover, for the second distribution, we are able to show that, with high probability,

$$\mathsf{nK}^t\big(x \circ \mathcal{U}_{nk+k} \circ \mathcal{U}_\ell\big) \gtrsim (\ell + nk + k)\text{-}\mathsf{nK}^t(x) + \ell + nk + k.$$

(The formal statement appears in Lemma 43.[7]) Notably, the time bound appearing on the right-hand side remains t. In particular, this inequality can be viewed as a form of weak SoI that incurs no overhead in the time bound. This is essentially the key benefit of working with the more flexible notion of ℓ-nK^t, which allows us to avoid the large overhead present in traditional time-bounded SoI results.

Now, by choosing

$$k \approx (\ell + nk)\text{-}\mathsf{nK}^{t-\mathsf{poly}(n)}(x \circ y) - (\ell + nk + k)\text{-}\mathsf{nK}^t(x),$$

the algorithm for MINnKT distinguishes between these two distributions.[8] As a result, given x, we can distinguish $\mathsf{DP}_k(y, \mathcal{U}_{nk})$ from $\mathcal{U}_{nk+k}$. Hence, by the reconstruction property of the direct product generator, we obtain

$$\begin{aligned}\mathsf{pK}^{\mathsf{poly}(t)}(y \mid x) &\lesssim (\ell + nk)\text{-}\mathsf{nK}^{t-\mathsf{poly}(n)}(x \circ y) - (\ell + nk + k)\text{-}\mathsf{nK}^t(x) \\ &\lesssim \mathsf{pnK}^{\mathsf{poly}(n)}(y \mid x) + (\ell + nk)\text{-}\mathsf{nK}^{t-\mathsf{poly}(n)}(x) - (\ell + nk + k)\text{-}\mathsf{nK}^t(x),\end{aligned}$$

where the second inequality uses the fact that, to print x, y, and a random string, it suffices to first print x and the random string, and then print y given x and the random string; under our assumption on the computation model (see the discussion below), these two programs can be composed efficiently. Note that the difference between $(\ell + nk)\text{-}\mathsf{nK}^{t-\mathsf{poly}(n)}(x)$ and $(\ell + nk + k)\text{-}\mathsf{nK}^t(x)$ is only additive $\mathsf{poly}(n)$, both in the time bound and in the length of the random string.[9] Thus, we can apply the telescoping trick to average out their difference, as explained earlier.

Theorem 1 and Theorem 9 hold with respect to any computational model that behaves nicely with respect to the composition of programs. This is sometimes implicitly assumed in the literature on (time-bounded) Kolmogorov complexity. For instance, it is often used that if a function f on an input x is computed by a program Π_f in time t_f, and a function g on an input $y = f(x)$ is computed by a program Π_g in time t_g, then $g(f(x))$ can be computed in time $t_f + t_g +$ "lower order terms" by a program of description length $|\Pi_f| + |\Pi_g| +$"lower order terms".[10] Since our results explore a novel regime of parameters and depend in an important way on claims of this form, for precision and clarity of the presentation, we explicitly postulate in Section 2.2 the necessary properties of the computational model and encodings.

Acknowledgements. This work received support from the UKRI Frontier Research Guarantee Grant EP/Y007999/1 and the Centre for Discrete Mathematics and its Applications (DIMAP) at the University of Warwick.

2 Preliminaries

2.1 Basic Notation

We use ϵ to denote the empty string. The length of a string x is denoted $|x|$. Given strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, we let $x \circ y \in \{0,1\}^{n+m}$ denote their concatenation.

We write $x \sim \mathcal{D}$ to denote that x is sampled according to the distribution $\mathcal{D}$. For an element a in the support of $\mathcal{D}$, we let $\mathcal{D}(a)$ denote its probability. We use $\mathcal{U}_\ell$ to denote the uniform distribution over $\{0,1\}^\ell$.

[7] Also, in the formal proof, ℓ-nK is defined using different probability parameters rather than the fixed value $2/3$, which is omitted here for simplicity. The probability parameters in the two bounds differ by $1/\mathsf{poly}(n)$.

[8] For simplicity, k appears on the right-hand side of the definition here, so it is not immediately clear that such a k exists. In the actual proof, by observing that ℓ-nK^t is roughly non-increasing in ℓ, choosing $k \approx \ell\text{-}\mathsf{nK}^{t-\mathsf{poly}(n)}(x, y) - (\ell + \mathsf{poly}(n))\text{-}\mathsf{nK}^t(x)$ suffices.

[9] Also the difference in the probability parameters is only $1/\mathsf{poly}(n)$.

[10] As an illustrative example, think about the necessary properties required to establish the easy direction of symmetry information, which states that $\mathsf{K}^t(x, y) \leq \mathsf{K}^{t_1}(x) + \mathsf{K}^{t_2}(y \mid x) + O(\log(|x| + |y|))$ for $t \approx t_1 + t_2$.

2.2 Strongly Efficient Computational Models

Even though all standard machine models (single-tape, multi-tape, random-access, etc.) are equivalent up to a polynomial difference in running time, to prove Theorem 1 and Theorem 9 we require a computational model and a corresponding encoding of programs that behave well with respect to the time and description length overhead when composing computations.

In this section, we introduce in a semi-formal way the notion of *strongly efficient* computational model. This is sufficient for the purpose explained above. While the properties that we require are fairly natural and intuitively should hold for any carefully designed computational model and encoding, for concreteness we describe in Appendix A an explicit model that satisfies the relevant properties.

Strongly Efficient Computational Models. We assume an underlying machine model (e.g, Turing machines, RAM, etc.). A *program* Π is a partial function mapping three strings (corresponding to input, non-determinism, and randomness, respectively) to another string. Unless stated otherwise, we use r to denote the string corresponding to the randomness, and w to denote the string corresponding to the non-determinism. A program has two attributes: a non-negative integer *time* (think of the program computing over the empty string ϵ or over an explicitly described input), and a binary string *description*. The description of a program Π uniquely specifies the program. We abuse notation in that Π can refer to either the function or the description depending on the context. We define the *size* of a program Π to be the length of its description, written as $|\Pi|$.

We say that a program Π computes $(x; r) \to y$ *non-deterministically* in time t, if

- $\forall w \in \{0,1\}^t$, $\Pi(x; w; r)$ outputs either y or $\perp$ in time t, and
- $\exists w \in \{0,1\}^t$, $\Pi(x; w; r)$ outputs y in time t.

If Π has size s, we say $(x; r) \to y$ can be computed non-deterministically in time t by a program of size s, or in short, $(x; r) \to y$ in non-deterministic time t and size s. When $r = \epsilon$, we may omit it and use $x \to y$ as short for $(x; \epsilon) \to y$.[11]

We are ready to introduce the concept of strongly efficient computational model.

Definition 12 (Strongly Efficient Computational Model)**.** We say that a computational model and corresponding program encoding are *strongly efficient* if the following properties hold:[12]

(P1) Efficient Simulation: The computational model can be efficiently simulated by Turing machines. More precisely, there exists a polynomial p and a Turing machine M with four input tapes and one output tape, such that for any program Π, if $\Pi(x; w; r) = y$ in time t, then $M(\Pi; x; w; r)$ outputs y in $p(t + |\Pi|)$ steps.

(P2) Efficient Universality: The computational model can efficiently simulate Turing machines. More precisely, for every Turing machine M with four input tapes and one output tape, there is a polynomial p_M such that if $M(\alpha; x; w; r)$ outputs y in t steps, then there exists a program Π of size at most $|\alpha| + \log p_M(|\alpha|)$ such that $\Pi(x; w; r) = y$ in time $p_M(t)$.

(P3) Randomness and Non-Determinism Extension: For any program Π, suppose $\Pi(x; w; r) = y$ in time t. If $|w| \geq t$, then for any $w' \in \{0,1\}^*$, $\Pi(x; w \circ w'; r) = y$ in time t. Similarly, if $|r| \geq t$, then for any $r' \in \{0,1\}^*$, $\Pi(x; w; r \circ r') = y$ in time t.

[11] Note that, by design, the aforementioned notation using arrows coincide with the notion of conditional nK, which is formally introduced in Section 2.3.

[12] In fact, our results only need relaxed forms of these properties. However, we describe the properties in this way because they are natural and simplify the presentation.

(P4) Efficient Composition: There exists a polynomial p such that the following holds. For any $x, y, z, r \in \{0,1\}^*$, if $(x; r) \to y$ can be computed non-deterministically in time t_1 by a program of size s_1, and $y \to z$ can be computed non-deterministically in time t_2 by a program of size s_2, then $(x; r) \to z$ can be computed non-deterministically in time $t_1 + p(t_2) + p(|x| + |y| + |z|)$ by a program of size $s_1 + s_2 + \log p(|x| + |y| + |z|)$.

As alluded to above, for completeness, in Appendix A we provide an example of a computational model that satisfies the aforementioned properties.

How is this different from other papers in meta-complexity/time-bounded Kolmogorov complexity? First, we stress that properties (P1), (P2), (P3), and (P4) stated above are not stringent. This seems clear in the case of (P1), (P2), and (P3), so we comment on the case of (P4). In all our proofs, we have $|x|, |y|, |z| \leq \mathsf{poly}(n)$ and $t_1, t_2 \leq 2^n$ (indeed t_1 and t_2 are large enough polynomials in n in our arguments), where n is the relevant input length. Moreover, since under our definition the time-bounded Kolmogorov complexity of every string of length n is at most $n + O(\log n)$,[13] (P4) is actually weaker than the requirement that under composition the final running time is at most $t_1 + t_2 + \mathsf{poly}(|x| + |y| + |z| + \log t_1 + \log t_2)$ and the final encoding length is at most $s_1 + s_2 + O(\log(|x| + |y| + |z| + s_1 + s_2))$. We believe these are widely adopted conventions in the literature (even if implicit in proofs), which we make explicit in our exposition. (Formally, in (P4) we even allow the second computation to run in time $\mathsf{poly}(t_2)$ after composition, which is sufficient in our proofs. For this reason, one can think of it as a "weak" form of efficient composition with respect to the running time.)

The main difference in our case compared with some other works is that in order to have precise control over running times, we measure time complexity as the number of steps in the execution of a program Π, as opposed to the number of steps that a universal machine U takes to simulate Π for t steps (which could be for instance of order $t \cdot \mathsf{poly}(\log t)$). This distinction is almost always irrelevant, and perhaps for this reason it is often not explicitly noted in the literature,[14] but it is relevant in our arguments.

Of course, if a program Π expects a *description* of a program Π' as an input string, and Π' runs in time t', then there is a time overhead during the simulation of Π' by Π. This is also the reason why we developed the notion of $\ell\text{-}\mathsf{nK}^t(x)$ complexity discussed in Section 1.3, as it avoids the use of weak symmetry of information for the standard formulation of time-bounded Kolmogorov complexity, whose known proofs lead to a simulation overhead. (Note that, under (P4), there will be almost no overhead when composing two explicit programs, since in this case the programs are directly *executed* instead of *simulated*.)

Finally, we are not aware of results from the literature that would not remain valid under the conventions adopted above.

2.3 Time-Bounded Kolmogorov Complexity

Definition 13 (K^t)**.** For $x, y \in \{0,1\}^*$, $t \in \mathbb{N}$, and an oracle $\mathcal{O}$, we define the *t-time-bounded Kolmogorov complexity of x conditioning on y and given oracle $\mathcal{O}$* as

$$\mathsf{K}^{t,\mathcal{O}}(x \mid y) := \min_{\Pi \in \{0,1\}^*} \left\{ |\Pi| \;\middle|\; \Pi^{\mathcal{O}}(y; \epsilon; \epsilon) \text{ outputs } x \text{ within } t \text{ steps} \right\}.$$

We can consider the case where the oracle calls are *non-adaptive*. In this case, we will write $\mathsf{K}^{t,\|\mathcal{O}}(x \mid y)$.

Next, we will define other variants of the time-bounded Kolmogorov complexity measure. For simplicity, we will define only the basic versions without oracles. These can easily be generalized to settings where an oracle is present.

[13] This bound easily follows from (P2).

[14] We observe that some references do adopt a similar convention when defining time-bounded Kolmogorov complexity by using "time" as the number of steps of the program instead of the universal machine, such as [LP22, Section 2.3].

Definition 14 (pK^t)**.** For $x, y \in \{0,1\}^*$, $\lambda \in [0,1]$, $t \in \mathbb{N}$, the *probabilistic t-time-bounded Kolmogorov complexity of x conditioning on y* is defined as

$$\mathsf{pK}^t_\lambda(x \mid y) := \min\left\{ s \in \mathbb{N} \;\middle|\; \Pr_{r\sim\{0,1\}^t}[\exists \Pi \in \{0,1\}^s \text{ s.t. } \Pi(y;\epsilon;r) \text{ outputs } x \text{ within } t \text{ steps}] \geq \lambda \right\}.$$

We define $\mathsf{pK}^t_\lambda(x)$ as $\mathsf{pK}^t_\lambda(x \mid \epsilon)$. We omit the subscript λ when $\lambda = 2/3$.

Definition 15 (nK^t)**.** For $x, y, r \in \{0,1\}^*$ and $t \in \mathbb{N}$, the *nondeterministic t-time-bounded Kolmogorov complexity of x conditioning on $(y;r)$* is defined as

$$\mathsf{nK}^t(x \mid y; r) : \min_{\Pi \in \{0,1\}^*} \left\{ |\Pi| \;\middle|\; \begin{array}{l} \forall w \in \{0,1\}^t, \Pi(y;w;r) \text{ outputs } x \text{ or } \perp \text{ in } t \text{ steps;} \\ \exists w \in \{0,1\}^t, \Pi(y;w;r) \text{ outputs } x \text{ in } t \text{ steps} \end{array} \right\}.$$

We define $\mathsf{nK}^t(x \mid y)$ as $\mathsf{nK}^t(x \mid y; \epsilon)$, and $\mathsf{nK}^t(x)$ as $\mathsf{nK}^t(x \mid \epsilon)$.

Definition 16 (rnK^t)**.** For $x, y \in \{0,1\}^*$, $\lambda \in [0,1]$ and $t \in \mathbb{N}$, the *randomized nondeterministic t-time-bounded time-bounded Kolmogorov complexity of x conditioning on y* is defined as

$$\mathsf{rnK}^t_\lambda(x \mid y) : \min_{\Pi \in \{0,1\}^*} \left\{ |\Pi| \;\middle|\; \Pr_{r\sim\{0,1\}^t} \left[\begin{array}{l} \forall w \in \{0,1\}^t, \Pi(y;w;r) \text{ outputs } x \text{ or } \perp \text{ in } t \text{ steps;} \\ \exists w \in \{0,1\}^t, \Pi(y;w;r) \text{ outputs } x \text{ in } t \text{ steps} \end{array} \right] \geq \lambda \right\}.$$

We define $\mathsf{rnK}^t_\lambda(x)$ as $\mathsf{rnK}^t_\lambda(x \mid \epsilon)$. We omit the subscript λ when $\lambda = 2/3$.

Definition 17 (pnK^t)**.** For $x, y \in \{0,1\}^*$, $\lambda \in [0,1]$ and $t \in \mathbb{N}$, the *probabilistic nondeterministic t-time-bounded time-bounded Kolmogorov complexity of x conditioning on y* is defined as

$$\mathsf{pnK}^t_\lambda(x \mid y) := \min\left\{ s \in \mathbb{N} \;\middle|\; \Pr_{r\sim\{0,1\}^t} \left[\begin{array}{l} \exists \Pi \in \{0,1\}^{\leq s} \text{ such that the following conditions hold:} \\ \forall w \in \{0,1\}^t, \Pi(y;w;r) \text{ outputs } x \text{ or } \perp \text{ in } t \text{ steps;} \\ \exists w \in \{0,1\}^t, \Pi(y;w;r) \text{ outputs } x \text{ in } t \text{ steps} \end{array} \right] \geq \lambda \right\}.$$

Equivalently,

$$\mathsf{pnK}^t_\lambda(x \mid y) := \min\left\{ s \in \mathbb{N} \;\middle|\; \Pr_{r\sim\{0,1\}^t}\left[\mathsf{nK}^t(x \mid y; r) \leq s\right] \geq \lambda \right\}.$$

We define $\mathsf{pnK}^t_\lambda(x)$ as $\mathsf{pnK}^t_\lambda(x \mid \epsilon)$. We omit the subscript λ when $\lambda = 2/3$.

In the remainder of this subsection, we state some useful tools about Kolmogorov complexity.

Lemma 18 (Following [GKLO22a, Lemma 18])**.** *There exists a universal constant $c > 0$ such that for any $x \in \{0,1\}^*$ and time bound $t \in \mathbb{N}$, the following hold.*

- $\mathsf{K}(x) \leq \mathsf{pK}^t(x) + c \cdot \log t$.
- $\mathsf{K}(x) \leq \mathsf{pnK}^t(x) + c \cdot \log t$.

Lemma 19 (See [HIL+23, Lemma 9])**.** *There exists a universal constant $b > 0$ such that for any distribution family $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{D}_n$ is over $\{0,1\}^n$, and $\gamma \in \mathbb{N}$,*

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

Theorem 20 (Efficient Coding Theorem [LOZ22])**.** *For every polynomial-time samplable distribution family $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{D}_n$ is supported over $\{0,1\}^n$, there exists a polynomial p such that for every $x \in \mathsf{Support}(\mathcal{D}_n)$,*

$$\mathsf{pK}^{p(n)}(x) \;\leq\; \log \frac{1}{\mathcal{D}_n(x)} + \log p(n).$$

2.4 Direct Product Generator

For $x, y \in \{0,1\}^n$, we denote their inner product by $x \cdot y = \bigoplus_{i=1}^{n} x_i y_i$.

Definition 21 (Direct Product Generator (DPG) [Hir21])**.** For $n, k \in \mathbb{N}$, the k-wise direct product generator is the function

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

defined by

$$\mathsf{DP}_k(x, z_1 \circ \cdots \circ z_k) = (z_1 \circ \cdots \circ z_k \circ x \cdot z_1 \circ \cdots \circ x \cdot z_k).$$

The reconstruction lemma for the direct product generator states that, given access to an oracle distinguishing between $\mathsf{DP}_k(x, \mathcal{U}_{nk})$ and $\mathcal{U}_{nk+k}$, along with k bits of advice, we can recover x with $1/\mathsf{poly}(n)$ probability.

Lemma 22 (DPG Reconstruction [Hir21])**.** *There exists a pair of algorithms* $\mathsf{Recon}^{\mathcal{O}}$ *and* Ad*, and a polynomial* p*, such that for any* $n, m, k \in \mathbb{N}^+$*, we have the following.*

1. *For any* $r \in \{0,1\}^{p(nmk)}$*, any* $a \in \{0,1\}^k$*, and any* $D \colon \{0,1\}^{nk+k} \to \{0,1\}$*,* $\mathsf{Recon}^D(1^n, 1^m, 1^k, r, a)$ *runs in at most* $p(nmk)$ *steps, makes at most* $p(nmk)$ *queries to* D*, and outputs a string* $x \in \{0,1\}^n$*. Furthermore, the queries to* D *are non-adaptive; that is,* Recon *generates the full list of positions to query before receiving any answers.*
2. *For any* $x \in \{0,1\}^n$ *and any* $r \in \{0,1\}^{p(nmk)}$*,* $\mathsf{Ad}(1^n, 1^m, 1^k, x, r)$ *runs in at most* $p(nmk)$ *steps and outputs a string* $a \in \{0,1\}^k$*.*
3. *For any* $x \in \{0,1\}^n$ *and any function* D *that* $(1/m)$*-distinguishes between* $\mathsf{DP}_k(x, \mathcal{U}_{nk})$ *and* $\mathcal{U}_{nk+k}$*, we have*

$$\Pr_{r \sim \mathcal{U}_{p(nmk)}} \left[\mathsf{Recon}^D(1^n, 1^m, 1^k, r, \mathsf{Ad}(1^n, 1^m, 1^k, x, r)) = x \right] \geq \frac{1}{p(nmk)}.$$

Proof Sketch. The proof works by first converting the distinguisher into a next-bit predictor using Yao's hybrid argument [Yao82]. It then employs the Goldreich–Levin local list-decoding algorithm for the Hadamard code to obtain a polynomial-sized list containing x. We observe that both Yao's argument and the Goldreich–Levin algorithm can be implemented using only non-adaptive queries (see, e.g., [AB09, Sections 9.3.1 and 9.3.2]). Consequently, our final reconstruction algorithm also relies solely on non-adaptive queries. □

Lemma 23 (pK^t Reconstruction Lemma)**.** *There is a polynomial* p_{DP} *such that the following holds. For every* $n, m, k \in \mathbb{N}^+$*,* $x \in \{0,1\}^n$*, let* D *be a function that* $(1/m)$*-distinguishes* $\mathsf{DP}_k(x; \mathcal{U}_{nk})$ *from* $\mathcal{U}_{nk+k}$*. Then*

$$\mathsf{pK}^{p_{\mathsf{DP}}(nmk), \|D}(x) \leq k + \log p_{\mathsf{DP}}(nmk).$$

Proof. The lemma follows easily from the reconstruction property of the direct product generator (Lemma 22) and the success amplification property of pK^t (Lemma 29). □

Corollary 24 (pK^t Reconstruction with a Randomized Distinguisher)**.** *There is a polynomial* p *such that the following holds. For every* $n, m, k, t \in \mathbb{N}^+$*, and any string* $x \in \{0,1\}^n$*, let* D *be a function* $D \colon \{0,1\}^t \times \{0,1\}^{nk+k} \to \{0,1\}$ *satisfying*

$$\mathop{\mathbb{E}}_{\substack{w \sim \mathcal{U}_t \\ r \sim \mathcal{U}_{nk} \\ r' \sim \mathcal{U}_{nk+k}}} \left[D(w, \mathsf{DP}_k(x; r)) - D(w, r') \right] \geq \frac{1}{m}.$$

Then we have

$$\mathsf{pK}^{p(nmkt), \|D}(x) \leq k + \log p(nmkt).$$

Proof. Using Markov's inequality, one can argue that with probability at least $\frac{1}{2m}$ over the $w \sim \mathcal{U}_t$, $D(w,\cdot)$ $\frac{1}{2m}$-distinguishes $\mathsf{DP}_k(x;\mathcal{U}_{nk})$ from $\mathcal{U}_{nk+k}$. Therefore, we can first sample $w \sim \mathcal{U}_t$, then make all queries to $D(w,\cdot)$. By Lemma 23, there exists some large enough polynomial p_0 such that

$$\mathsf{pK}^{p_0(nmkt),\|D}_{1/(3m)}(x) \leq k + \log p_0(nmkt).$$

By Lemma 29, we can then amplify the success probability to be $2/3$. Thus, when p is a large enough polynomial, we obtain

$$\mathsf{pK}^{p(nmkt),\|D}(x) \leq k + \log p(nmkt),$$

as desired. □

2.5 Universal Time-Bounded Sampler

Definition 25 (Universal Time-Bounded Sampler)**.** Let $m, t \in \mathbb{N}$ and $x \in \{0,1\}^*$. The universal sampler $\mathsf{USamp}(1^m, 1^t, x)$ does the following.

1. Pick a uniformly random $k \sim [O(m)]$.
2. Pick a uniformly random $r \sim \{0,1\}^t$.
3. Pick a uniformly random $\Pi \sim \{0,1\}^k$.
4. Let y be the output of $\Pi(x;r)$ after running for t steps.
5. Output y.

Note that USamp runs in polynomial time. The following proposition follows easily from the definitions of pK^t and USamp.

Proposition 26. *For every $m, t, k \in \mathbb{N}$, $y \in \{0,1\}^m$ and $x \in \{0,1\}^*$, if $\mathsf{pK}^t(y \mid x) \leq k$, then $\mathsf{USamp}(1^m, 1^t, x)$ outputs y with probability $\Omega\big(1/(m \cdot 2^k)\big)$, where USamp is the universal sampler defined in Definition 25.*

2.6 Hash Functions

Definition 27 (Pairwise-Independent Hash Family)**.** For any $n, m \in \mathbb{N}$, a set of functions $H \subseteq \{0,1\}^n \to \{0,1\}^m$ is called a *pairwise-independent hash family* if for any $x, y \in \{0,1\}^n$ such that $x \neq y$ and any $r, s \in \{0,1\}^m$,

$$\Pr_{h \sim H}[h(x) = r \wedge h(y) = s] = 2^{-2m}.$$

Proposition 28. *There exists a polynomial p and a randomized algorithm A such that, for every $n, m \in \mathbb{N}$ and every random string $r \in \{0,1\}^{p(nm)}$, $A(1^n, 1^m, r, \cdot)$ computes a function $\{0,1\}^n \to \{0,1\}^m$ in time $p(nm)$. Furthermore, $H := \big\{A(1^n, 1^m, r, \cdot) \mid r \in \{0,1\}^{p(nm)}\big\}$ is a pairwise-independent hash family.*

Proof. We can construct pairwise-independent hash families using finite fields $\mathsf{GF}(2^n)$. When $n = m$, for any $x \in \{0,1\}^n$, we sample $a, b \sim \mathcal{U}_n$, and compute $h_{a,b}(x) = a \cdot x + b$, where addition and multiplication are done over $\mathsf{GF}(2^n)$. Such a computation can be done in time $\mathsf{poly}(n)$, see e.g. [AB09, Appendix A.4]. Then for every $y \in \{0,1\}^n \setminus \{x\}$ and any $r, s \in \{0,1\}^n$, there exists exactly one pair of strings (a, b) satisfying $a \cdot x + b = r$ and $a \cdot y + b = s$. In other words, $\{h_{a,b} \mid a, b \in \{0,1\}^n\}$ is a pairwise-independent hash family. For $m < n$, we can simply discard the first $(n - m)$ bits of $h_{a,b}(x)$. For $m > n$, we can define $h'_{a,b}(x) = a \cdot (0^{m-n} \circ x) + b$. □

2.7 Average-Case Complexity

Recall that a pair $(L, \mathcal{D})$ is a *distributional problem* if $L \subseteq \{0,1\}^*$ and $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ is a distribution family, where each $\mathcal{D}_n$ is a distribution over $\{0,1\}^*$.

We let DistNP denote the set of distributional problems $(L, \mathcal{D})$ such that $L \in \mathsf{NP}$ and $\mathcal{D}$ is polynomial-time samplable. Here, a distribution family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ is *polynomial-time samplable* if there exists a randomized polynomial-time algorithm A such that for every $n \geq 1$, the output of $A(1^n)$ is distributed according to $\mathcal{D}_n$.

A distributional problem $(L, \mathcal{D})$ is said to admit a (errorless) *heuristic scheme* if there exists a probabilistic polynomial-time algorithm A such that the following holds for every $n, k \in \mathbb{N}$:

1. For every $x \in \mathsf{Support}(\mathcal{D}_n)$,
$$\Pr_A\left[A(x, 1^n, 1^k) \in \{L(x), \perp\}\right] \geq \frac{4}{5},$$
2. and
$$\mathop{\mathbf{Pr}}_{x\sim\mathcal{D}_n}\left[\Pr_A[A(x, 1^n, 1^k) = \perp] < \frac{1}{5}\right] \geq 1 - \frac{1}{k}.$$

We let AvgBPP denote the set of distributional problems that admit a heuristic scheme. For more information about average-case complexity, we refer to [BT06].

Also, by slightly abusing notation, we say that MINnKT is average-case easy under the uniform distribution, or $(\mathsf{MINnKT}, \mathcal{U}) \in \mathsf{AvgBPP}$, if the following holds.

> There exist a polynomial-time algorithm A and a polynomial ρ such that the following holds for all $n, s, k \in \mathbb{N}$, and all $t \geq \rho(n)$.
>
> 1. For all $x \in \{0,1\}^n$,
> $$\mathop{\mathbf{Pr}}_{A}\left[A(x, 1^s, 1^t, 1^k) \in \{\mathsf{MINnKT}(x, 1^s, 1^t), \perp\}\right] \geq \frac{4}{5},$$
> 2. and
> $$\mathop{\mathbf{Pr}}_{x\sim\{0,1\}^n}\left[\mathop{\mathbf{Pr}}_{A}[A(x, 1^s, 1^t, 1^k) = \perp] < 1/5\right] \geq 1 - \frac{1}{k}.$$

3 Properties of pnK^t

3.1 Basic Properties of pnK^t

Lemma 29 (Success Amplification)**.** *For any string $x \in \{0,1\}^n$, time bound $t \in \mathbb{N}$, any $0 < \alpha < \beta < 1$ and any oracle $\mathcal{O}$, we have*
$$\mathsf{pnK}_\beta^{O(kt),\mathcal{O}}(x) \leq \mathsf{pnK}_\alpha^{t,\mathcal{O}}(x) + O(\log k),$$
where $k := \left\lceil \frac{\log(1-\beta)}{\log(1-\alpha)} \right\rceil$. Moreover, the above also holds in the setting where the oracle access is non-adaptive. That is,
$$\mathsf{pnK}_\beta^{O(kt),\|\mathcal{O}}(x) \leq \mathsf{pnK}_\alpha^{t,\|\mathcal{O}}(x) + O(\log k).$$

Proof. Let $l := \mathsf{pnK}_\alpha^{t,\mathcal{O}}(x)$. We say that a string $r \in \{0,1\}^t$ is *good* if there exists a program $\Pi \in \{0,1\}^l$ such that

- For every $w \in \{0,1\}^t$, $\Pi^{\mathcal{O}}(w;r)$ outputs either x or $\perp$ in t steps;
- There exists some $w \in \{0,1\}^t$ such that $\Pi^{\mathcal{O}}(w;r)$ outputs x in t steps.

By the definition of pnK and l, we have

$$\Pr_{r\sim\{0,1\}^t}[r \text{ is good}] \geq \alpha.$$

We can then take $k = \left\lceil \frac{\log(1-\beta)}{\log(1-\alpha)} \right\rceil$ independent samples $r_1, \ldots, r_k$ from $\mathcal{U}_t$. The probability of getting at least one good sample among $r_1, \ldots, r_k$ is at least $1-(1-\alpha)^k \geq \beta$. If we have a good sample, then we can store the index i such that r_i is good using $\lceil \log k \rceil$ bits, and there exists a program Π such that $\Pi^{\mathcal{O}}(w; r_i)$ outputs x. Hence, we conclude that $\mathsf{pnK}_{\beta}^{O(kt),\mathcal{O}}(x) \leq l + O(\log k)$. Notice that if the original program makes parallel queries to $\mathcal{O}$, then so does the new program.

The above proof can be easily extended to the setting of non-adaptive oracle access, which shows the "moreover" part of the lemma. □

Since for any $0 < \alpha < 1$, $-1/\log(1-\alpha) \leq 1/\alpha$ holds, we have the following corollary:

Corollary 30. *For any string $x \in \{0,1\}^n$, time bound $t \in \mathbb{N}$, any $0 < \alpha < 1$ and any oracle $\mathcal{O}$, we have*

$$\mathsf{pnK}^{O(2t/\alpha),\mathcal{O}}(x) \leq \mathsf{pnK}_{\alpha}^{t,\mathcal{O}}(x) + O(\log(1/\alpha)).$$

Moreover, the above also holds in the setting where the oracle access is non-adaptive, i.e.

$$\mathsf{pnK}^{O(2t/\alpha),\|\mathcal{O}}(x) \leq \mathsf{pnK}_{\alpha}^{t,\|\mathcal{O}}(x) + O(\log(1/\alpha)).$$

Lemma 31. *For any string $x, y \in \{0,1\}^*$, time bound $t \in \mathbb{N}$, any oracle $\mathcal{O}$, we have*

$$\mathsf{pnK}^{\mathsf{poly}(t),\mathcal{O}}(x \mid y) \leq \mathsf{pK}^{t,\|\mathsf{NP}^{\mathcal{O}}}(x \mid y) + O(\log t).$$

Proof. Suppose $\mathsf{pK}^{t,\|\mathsf{NP}^{\mathcal{O}}}(x \mid y) = s$. Then, with probability at least $2/3$ over the randomness $r \sim \{0,1\}^t$, there exists a program of size s that, given r, y and *non-adaptive* access to some oracle $A \in \mathsf{NP}^{\mathcal{O}}$, runs in t steps and outputs x. Suppose A is the following oracle:

$$A(q) = 1 \iff \exists y \in \{0,1\}^{\mathsf{poly}(|q|)} \text{ such that } V^{\mathcal{O}}(q,w) = 1,$$

where V is some polynomial-time verifier. We will replace oracle access to A with access only to $\mathcal{O}$ by guessing the answers to the (non-adaptive) queries to A and verifying them using $\mathcal{O}$, with the help of an additional small amount of advice.

More specifically, fix a randomness r and a program Π (which may depend on r). Let $q_1, q_2, \ldots, q_\ell$, where $\ell \leq t$, be the (non-adaptive) queries made by $\Pi(r)$ to the oracle A. Also, let p be the number of positive answers to these queries, i.e., $p := \sum_{i=1}^{\ell} A(q_i)$. We then non-deterministically guess ℓ witnesses $(w_1, w_2, \ldots, w_\ell) =: y$ and count the number of indices i for which $V^{\mathcal{O}}(q_i, w_i) = 1$. Denote this number by u_w. If u_w is not equal to p, we output $\perp$. Otherwise, for each w_i that satisfies the verifier V, we set the answer to the query q_i to be 1, and we let the rest of the answers be 0. We then run $\Pi^A(r)$ while simulating the oracle A using the answers obtained as described above.

Note that, given the number p, which can be specified using $\lceil \log \ell \rceil \leq \lceil \log t \rceil$ bits, the above procedure can be implemented to run in time $\mathsf{poly}(t)$ with only oracle access to $\mathcal{O}$. Also, it is easy to observe that there exists some guess of $w = (w_1, w_2, \ldots, w_\ell)$ that satisfies $u_w = p$. Moreover, for any guess of w that satisfies $u_w = p$, the answers obtained are the correct answers to the queries $(q_1, q_2, \ldots, q_\ell)$ for A. It follows that, with high probability, there exists a program of size at most $s + O(\log t)$ that, given y and oracle access to $\mathcal{O}$, runs in time $\mathsf{poly}(t)$ and outputs x non-deterministically. □

3.2 Language Compression for pnK^t

3.2.1 Proof of Theorem 4

We provide two different proofs for the optimal language compression property of pnK^t. The first proof, given below, is based on an approach from [BLvM05] and relies on the direct product generator (Definition 21).

First Proof of Theorem 4. Let A_n be any non-empty subset of $\{0,1\}^n$. We set $k := \lceil \log |A_n| \rceil + 1$. We define

$$B_n := \left\{ y \in \{0,1\}^{nk+k} \mid \exists x \in A_n, z \in \{0,1\}^{nk} \text{ such that } y = \mathsf{DP}_k(x,z) \right\}.$$

By a simple counting argument, we have that $|B_n| \leq |A_n| \cdot 2^{nk}$. Also, by our definition of k, $|A_n| \leq 2^{k-1}$, and hence $|B_n| \leq \frac{1}{2} \cdot 2^{nk+k}$. By viewing $B_n \colon \{0,1\}^{nk+k} \to \{0,1\}$ as the characteristic function of the set B_n, we have

$$\mathbf{Pr}[B_n(\mathcal{U}_{nk+k}) = 1] \leq \frac{1}{2}.$$

On the other hand, from the definition of B_n, we have that for any $x \in A_n$,

$$\mathbf{Pr}[B_n(\mathsf{DP}_k(x, \mathcal{U}_{nk})) = 1] = 1.$$

Therefore, B_n $\frac{1}{2}$-distinguishes $\mathsf{DP}_k(x, \mathcal{U}_{nk})$ and $\mathcal{U}_{nk+k}$.

By the pK^t reconstruction lemma (Lemma 23), we get that there exists some polynomial p such that

$$\mathsf{pK}^{p(n),\|B_n}(x) \leq k + \log p(n). \tag{7}$$

Now, observe that the membership of B_n can be decided in NP^{A_n}. Indeed, given $y \in \{0,1\}^{nk+k}$, we can guess $x \in \{0,1\}^n$, $z \in \{0,1\}^{nk}$, and then check whether both $x \in A_n$ and $y = \mathsf{DP}_k(x,z)$. [15] Now, by applying Lemma 31 to Equation (7), we get that

$$\mathsf{pnK}^{\mathsf{poly}(n),A_n}(x) \leq k + O(\log n) = \lceil \log |A_n| \rceil + 1 + O(\log n).$$

It follows that there exists a polynomial p_{LC} such that

$$\mathsf{pnK}^{p_{\mathsf{LC}}(n),A_n}(x) \leq \log |A_n| + \log p_{\mathsf{LC}}(n),$$

as desired. □

Next, we give a second proof that relies on hash functions.

Second Proof of Theorem 4. Let A_n be any non-empty subset of $\{0,1\}^n$, and let x be any string in A_n. We set $k := \lceil \log |A_n| \rceil + 2$. Let $H \subseteq \{0,1\}^n \to \{0,1\}^k$ be a family of pairwise-independent hash functions. Since H is pairwise-independent, we have

$$\Pr_{h \sim H}[h(x) = h(y)] = 2^{-k}$$

[15] Strictly speaking, we cannot say that B_n is decided in NP^{A_n} since checking the membership of a given string in B_n requires knowing the numbers n and k. One way to address this issue is to consider the set $B := \left\{ (y, n', k') \mid |y| = n'k' + k' \text{ and } \exists x \in \{0,1\}^{n'}, z \in \{0,1\}^{n'k'} \text{ s.t. } x \in A_n \text{ and } y = \mathsf{DP}_{k'}(x,z) \right\}$. We can then replace the oracle B_n with B while replacing every query y with (y, n, k), where n and k are fixed integers as specified in the proof and can be hardcoded in the description. For simplicity of presentation, we ignore this technicality.

for any $y \in A_n \setminus \{x\}$. Thus, by a union bound, we obtain

$$\Pr_{h \sim H}[\exists y \in A_n \setminus \{x\} \text{ such that } h(x) = h(y)] \leq 2^{-k} \cdot |A_n| \leq \frac{1}{4}. \tag{8}$$

For $h \sim H$, we include $v := h(x)$ as part of the description used to output x non-deterministically. Note that v can be specified using $\log |A_n| + 2$ bits. We can then guess $x' \in \{0,1\}^n$ and check whether $h(x') = v$ and $x' \in A_n$. If so, we output x'; otherwise, we output $\perp$.

By Equation (8), with probability at least $3/4$, we sample a good h such that $h(x)$ is unique in A_n, ensuring that the above procedure outputs x non-deterministically. It follows that

$$\mathsf{pnK}^{p_{\mathsf{LC}}(n), A_n}(x) \leq \log |A_n| + \log p_{\mathsf{LC}}(n)$$

for a sufficiently large polynomial p_{LC}. □

3.2.2 Proof of Theorem 5

Before presenting the proof of Theorem 5, we need the following lemma.

Lemma 32. *Let $m, t, \theta \in \mathbb{N}$, $A \subseteq \{0,1\}^m$, and $x \in \{0,1\}^*$. Suppose for every $y \in A$, it holds that*

$$\mathsf{pK}^t(y \mid x) \leq \log |A| + \theta.$$

Then $\mathsf{USamp}(1^m, 1^t, x)$ outputs some string in A with probability at least $\frac{1}{\mathsf{poly}(m) \cdot 2^\theta}$.

Proof. By Proposition 26, for *each* $y \in A$, $\mathsf{USamp}(1^m, 1^t, x)$ outputs y with probability at least

$$\frac{1}{O(m) \cdot 2^{\log |A| + \theta}} = \frac{1}{O(m) \cdot |A| \cdot 2^\theta}.$$

Hence the probability that $\mathsf{USamp}(1^m, 1^t, x)$ outputs *some* $y \in A$ is at least

$$|A| \cdot \frac{1}{O(m) \cdot |A| \cdot 2^\theta} \geq \frac{1}{O(m) \cdot 2^\theta},$$

as desired. □

Proof of Theorem 5. Suppose we have average-case pK^t language compression for P. We show how to solve any unary NP problem in randomized polynomial time.

Let $L \in \mathsf{Unary\text{-}NP}$. Without loss of generality, we assume that for every instance 1^n, the set of L-witnesses of 1^n (with respect to some fixed verifier) has a length of exactly m, where $m := m(n) = \mathsf{poly}(n)$. Let A denote the set of L-witnesses. Note that the membership of A is decidable in polynomial time. Then by assumption on average-case pK^t language compression, it holds that for every m, with probability at least $1/m^c$ over $y \sim A_m$,

$$\mathsf{pK}^{m^c}(x) \leq \log |A_m| + c \cdot \log m.$$

Let A'_m be the set of strings in A_m for which the above condition holds. Note that

$$|A'_m| \geq |A_m| / m^c.$$

Then for each $y \in A'_m$, we have

$$\begin{aligned} \mathsf{pK}^{n^c}(y) &\leq \log |A_m| + c \cdot \log m \\ &\leq \log |A'_m| + c \cdot \log m + c \cdot \log m \\ &= \log |A'_m| + 2c \cdot \log m. \end{aligned}$$

Now consider the algorithm that, given as input 1^n, simply runs $\mathsf{USamp}\big(1^{m(n)}, 1^{m(n)^c}, \epsilon\big)$. By Lemma 32, this algorithm outputs some string in A'_m, which consists of L-witnesses and is non-empty if $1^n \in L$, with probability at least $1/\mathsf{poly}(n)$. By standard amplification, this yields an efficient randomized algorithm for solving L with high probability. □

3.3 Conditional Coding for pnK^t: Proof of Theorem 6

Proof of Theorem 6. Let $\{\mathcal{D}_n\}$ be a polynomial-time samplable distribution family. Fix $n \in \mathbb{N}$ and $(x, y) \in \mathsf{Support}(\mathcal{D}_n)$.

Let $\mathcal{E} := \mathcal{D}_n(x \mid y)$. Define

$$B := \left\{ z \in \{0,1\}^n : \mathcal{D}_n(z \mid y) \geq \frac{\mathcal{E}}{16} \right\}.$$

Note that $|B| \leq 16/\mathcal{E}$. Now consider a family of pairwise-independent hash functions $H \subseteq \{0,1\}^n \to \{0,1\}^k$, where $k := \lceil \log \frac{1}{\mathcal{E}} \rceil + 7$. By a union bound, with probability at least $7/8$ over $h \sim H$, we have $h(x) \neq h(z)$ for any $z \in B \setminus \{x\}$. Hence, we can try to include $v := h(x)$ in our description for outputting x. Then we guess x' and output it if and only if $h(x') = v$.

In order for the above to work, we also need to make sure that the guessed string x' is not in B, since otherwise we could have $h(x') = h(x)$ but $x' \neq x$. Next, we describe how to exclude strings outside of B. The argument is similar to a construction from [GS86].

Let A be a sampler for $\{\mathcal{D}_n\}$ that has a running time $q(n)$, where q is some polynomial. For any $a, b \in \{0,1\}^n$, define $R_{a,b} := \{r \in \{0,1\}^{q(n)} : A(1^n; r) = (a, b)\}$. Now let $k' = \lfloor \log |R_{x,y}| \rfloor - 1$. Then we define another pairwise-independent hash family $H' \subseteq \{0,1\}^{q(n)} \to \{0,1\}^{k'}$. Then by Chebyshev's inequality,

$$\Pr_{h' \sim H'} \left[\exists r \in R_{x,y} \text{ such that } h'(r) = 0^{k'} \right] \geq \frac{1}{2}.$$

However, for any $z \notin B$, by definition, we have $|R_{z,y}| \leq |R_{x,y}|/16$, implying that $|R_{z,y}| \leq 2^{k'}/4$. Therefore, by a union bound, we have

$$\Pr_{h' \sim H'} \left[\exists r \in R_{z,y} \text{ such that } h'(r) = 0^{k'} \right] \leq \frac{1}{4}.$$

We can see that there is a gap between the cases of $z = x$ and of $z \notin B$, so our next step is to amplify this gap: We set $l = C \cdot n$ for some large enough constant C, and take $l = C \cdot n$ samples from H' to obtain $h'_1, \ldots, h'_l$, and use non-determinism to guess $r_1, \ldots, r_l \in \{0,1\}^{q(n)}$. We also use non-determinism to guess $z \in \{0,1\}^n$. Then we do the following checks. If any of these checks fail, halt and output $\perp$; otherwise, output z:

1. $A(1^n; r_1) = A(1^n; r_2) = \cdots = A(1^n; r_l) = (z, y)$.
2. The number of $i \in [l]$ such that $h'_i(r_i) = 0^{k'}$ is at least $l/3$.
3. $h(z)$ is the same as the string $v := h(x)$ specified in the description.

By the Chernoff bound and a union bound, when C is a large enough constant, with probability at least $7/8$ over $h'_1, \ldots, h'_l$, for any $z \notin B$, no sequence of $r_1, \ldots, r_l$ can pass the first and second checks. While with probability at least $1 - 2^{-n}$, there exists some sequence of $r_1, \ldots, r_l$ such that x can pass the check. Also, with probability at least $7/8$ over h, any $z \in B \setminus \{x\}$ cannot pass the third check. Therefore, by a union bound, with probability at least $1 - 1/8 - 1/8 - 2^{-n} \geq 2/3$, only x can pass the three checks. It is not hard to see that the entire reconstruction procedure runs in time polynomial in n, and we only need to specify $h(x)$ in this reconstruction algorithm, as well as the description of the sampler A. Hence, we conclude that there exists some polynomial p such that $\mathsf{pnK}^{p(n)}(x \mid y) \leq \log \frac{1}{\mathcal{D}_n(x|y)} + \log p(n)$. □

3.4 Symmetry of Information for pnK^t: Proofs of Theorem 7 and Theorem 8

In this section, we prove the results related to symmetry of information for pnK^t (Theorem 7 and Theorem 8).

Proof Sketch of Theorem 7. The proof follows a similar approach to that used in [GK22, Hir22c] to establish symmetry of information for pK^t under the assumption that MINKT is easy (see also [HKLO24, Appendix A]). This approach employs the direct product generator and its reconstruction property (Lemmas 22 and 23) while using MINKT (which is in NP) as a distinguisher. In fact, by adapting this approach and using the fact that the reconstruction procedure of the direct product generator makes only non-adapative queries, one can show that

$$\mathsf{pK}^t(x,y) \geq \mathsf{pK}^{\mathsf{poly}(t),\|\mathsf{NP}}(y \mid x) + \mathsf{pK}^{\mathsf{poly}(t),\|\mathsf{NP}}(x) + O(\log t).$$

Now note that the semi-symmetry of information property for pnK^t follows by combining the above with Lemma 31. □

Proof of Theorem 8. Let $\mathcal{D} = \{\mathcal{D}_n\}$ be any polynomial-time samplable distribution family and q be any polynomial. Let $p$and r be sufficiently large polynomials specified later. Fix $n \in \mathbb{N}$ and $t \geq p(n)$.

By Theorem 7, we have that for every $x, y \in \{0,1\}^n$, we have

$$\mathsf{pK}^{r(n)}(x,y) \geq \mathsf{pnK}^{p_{\mathsf{Sol}}(r(n))}(y \mid x) + \mathsf{pnK}^{p_{\mathsf{Sol}}(r(n))}(x) - \log p_{\mathsf{Sol}}(r(n)), \tag{9}$$

where p_{Sol} is the polynomial in Theorem 7.

First, by the coding theorem for pK^t (Theorem 20) and by letting r be a sufficiently large polynomial, we have

$$\mathsf{pK}^{r(n)}(x,y) \leq \log \frac{1}{\mathcal{D}_n(x,y)} + \log r(n). \tag{10}$$

On the other hand, by Lemma 19, we have

$$\Pr_{x \sim \mathcal{D}_n}\left[\mathsf{K}(x) \geq \log \frac{1}{\mathcal{D}_n(x)} - O(\log n) - \log q(n)\right] \geq 1 - \frac{1}{q(n)}. \tag{11}$$

It follows that with probability at least $1 - 1/q(n)$ over $(x,y) \sim \mathcal{D}_n$, we get

$$\begin{aligned}
\mathsf{pK}^{r(n)}(x,y) &\leq \log \frac{1}{\mathcal{D}_n(x,y)} + \log r(n) && \text{(by Equation (10))}\\
&\leq \mathsf{K}(x,y) + O(\log n) + \log q(n) + \log r(n) && \text{(by Equation (11))}\\
&\leq \mathsf{pnK}^t(x,y) + O(\log n) + \log q(n) + c' \cdot \log t, && \text{(by Lemma 18)}
\end{aligned}$$

where $c' > 0$ is a universal constant. Combining the above with Equation (9), we get

$$\mathsf{pnK}^t(x,y) \geq \mathsf{pnK}^{p_{\mathsf{Sol}}(r(n))}(y \mid x) + \mathsf{pnK}^{p_{\mathsf{Sol}}(r(n))}(x) - \log p_{\mathsf{Sol}}(r(n)) - O(\log n) - \log q(n) - c' \cdot \log t.$$

By letting p be a sufficiently large polynomial, the above yields

$$\mathsf{pnK}^t(x,y) \geq \mathsf{pnK}^t(y \mid x) + \mathsf{pnK}^t(x) - c \cdot \log t,$$

for some universal constant $c > 0$. □

4 Worst-Case Hardness of MINnKT: Proof of Theorem 1

In this section we prove Theorem 1.

Proof of Theorem 1. The theorem follows directly from Theorem 33 and Theorem 34, stated and proved in Section 4.1 and Section 4.2, respectively. □

4.1 The Easy Direction

Here we prove the following.

Theorem 33. *If* $\mathsf{NP} \subseteq \mathsf{BPP}$*, then* $\mathsf{MINnKT} \in \mathsf{BPP}$.

Proof. By [Ko82], we know that $\mathsf{NP} \subseteq \mathsf{BPP}$ implies $\mathsf{PH} \subseteq \mathsf{BPP}$. Therefore, it suffices to prove that $\mathsf{MINnKT} \in \mathsf{PH}$. By definition, we have

$$\begin{aligned}(x, 1^s, 1^t) \in \mathsf{MINnKT} &\iff \mathsf{nK}^t(x) \leq s \\ &\iff \exists \Pi, |\Pi| \leq s, \begin{cases} \forall w \in \{0,1\}^t, \Pi(\epsilon; w; \epsilon) \text{ outputs } x \text{ or } \perp \text{ in } t \text{ steps}, \\ \exists w \in \{0,1\}^t, \Pi(\epsilon; w; \epsilon) \text{ outputs } x \text{ in } t \text{ steps}. \end{cases}\end{aligned}$$

The last condition can be checked by a Σ_2^{p}-machine in time $\mathsf{poly}(s,t)$. Therefore, $\mathsf{MINnKT} \in \Sigma_2^{\mathsf{p}} \subseteq \mathsf{PH} \subseteq \mathsf{BPP}$. □

4.2 The Hard Direction

Throughout this subsection, we will assume that the underlying computational model is *strongly efficient*, as defined in Section 2.2. We will prove the hard direction of Theorem 1.

Theorem 34. *If* $\mathsf{MINnKT} \in \mathsf{BPP}$*, then* $\mathsf{NP} \subseteq \mathsf{BPP}$.

We will first define a technical measure of Kolmogorov complexity, namely ℓ-nK. We then prove several properties of ℓ-nK. Finally, we use the measure and its properties to establish Theorem 34.

4.2.1 Technical Tool: ℓ-nK^t_γ

Definition 35 (ℓ-nK^t_γ)**.** For any $t, \ell \in \mathbb{N}$, any $\gamma \in (0,1)$, and any string $x \in \{0,1\}^*$, we define ℓ-$\mathsf{nK}^t_\gamma(x)$ as

$$\ell\text{-}\mathsf{nK}^t_\gamma(x) = \min\left\{ s \in \mathbb{Z} \;\middle|\; \Pr_{r \sim \mathcal{U}_\ell}\left[\mathsf{nK}^t(x \circ r) \leq s + \ell\right] \geq \gamma \right\}.$$

From the definition, it is not immediately clear whether ℓ-$\mathsf{nK}^t_\gamma(x)$ is non-negative.[16] Using a simple counting argument, we can bound its value from below by $(-1 - \lceil \log 1/\gamma \rceil)$, as stated in the following lemma.[17]

Lemma 36 (Lower bound for ℓ-nK^t_γ)**.** *For any $\ell, t \in \mathbb{N}$, any string $x \in \{0,1\}^*$ and any $\gamma \in (0,1)$, we have*

$$\ell\text{-}\mathsf{nK}^t_\gamma(x) \geq -\left\lceil \log \frac{1}{\gamma} \right\rceil - 1.$$

Proof. For the sake of contradiction, suppose that ℓ-$\mathsf{nK}^t_\gamma(x) \leq -\lceil \log 1/\gamma \rceil - 2$. By the definition of ℓ-nK, we have

$$\Pr_{r \sim \mathcal{U}_\ell}\left[\mathsf{nK}^t(x \circ r) \leq \ell - \left\lceil \log \frac{1}{\gamma} \right\rceil - 2\right] \geq \gamma.$$

[16] Consider for instance this definition when γ is exponentially small.

[17] Jumping ahead, in the proof of Theorem 34 we only need to consider ℓ-$\mathsf{nK}^t_\gamma(x)$ for $\gamma = \Omega(1)$, and additive constants do not affect the argument.

Now define the set A as

$$A = \left\{ x \circ r \;\middle|\; r \in \{0,1\}^\ell,\ \mathsf{nK}^t(x \circ r) \leq \ell - \lceil \log 1/\gamma \rceil - 2 \right\}.$$

Then $|A| \geq \gamma \cdot 2^\ell$. However, the number of programs of length at most $(\ell - \lceil \log 1/\gamma \rceil - 2)$ is bounded by $2^0 + \cdots + 2^{\ell - \lceil \log 1/\gamma \rceil - 2} \leq 2^{\ell - \lceil \log 1/\gamma \rceil - 1} \leq \gamma \cdot 2^{\ell-1}$. Since $|A| \geq \gamma \cdot 2^\ell > \gamma \cdot 2^{\ell-1}$, by the pigeonhole principle, there must be a program computing two strings in A, which is a contradiction. □

Lemma 37 (Upper Bound for ℓ-nK^t_γ)**.** *There exists a polynomial p such that the following holds. For any $\ell \in \mathbb{N}$, any string $x \in \{0,1\}^*$ and any $\gamma \in (0,1)$, we have*

$$\ell\text{-}\mathsf{nK}^{p(|x|+\ell)}_\gamma(x) \leq |x| + \log p(|x| + \ell).$$

Proof. Let M be the Turing machine satisfying the following.

For any strings $y, w \in \{0,1\}^*$, $M(y; \epsilon; w; \epsilon) = y$ in $O(|y|)$ steps.

Hence by efficient universality (Definition 12), when p is a large enough polynomial, for any $r \in \{0,1\}^\ell$, $\epsilon \to (x \circ r)$ is in time $p(|x| + \ell)$ and size $(|x| + \ell + \log p(|x| + \ell))$. Therefore, we get

$$\Pr_{r \sim \mathcal{U}_\ell}\left[\mathsf{nK}^{p(|x|+\ell)}(x \circ r) \leq |x| + \ell + \log p(|x| + \ell) \right] = 1.$$

By the definition of ℓ-nK, we conclude that for any $\gamma \in (0,1)$,

$$\ell\text{-}\mathsf{nK}^{p(|x|+\ell)}_\gamma(x) \leq |x| + \log p(|x| + \ell).$$

This completes the proof. □

Lemma 38 (Monotonicity of ℓ-$\mathsf{nK}^t_\gamma(x)$ on γ and t)**.** *For any $\gamma_1, \gamma_2 \in (0,1)$ satisfying $\gamma_1 \leq \gamma_2$, any $\ell, t_1, t_2 \in \mathbb{N}^+$ satisfying $t_1 \geq t_2$, and any string $x \in \{0,1\}^*$, we have*

$$\ell\text{-}\mathsf{nK}^{t_1}_{\gamma_1}(x) \leq \ell\text{-}\mathsf{nK}^{t_2}_{\gamma_2}(x).$$

Proof. Let $s = \ell\text{-}\mathsf{nK}^{t_2}_{\gamma_2}(x)$. Then we have

$$\Pr_{r \sim \mathcal{U}_\ell}\left[\mathsf{nK}^{t_2}(x \circ r) \leq s \right] \geq \gamma_2.$$

By non-determinism extension (Definition 12), for any $r \in \{0,1\}^\ell$, we have $\mathsf{nK}^{t_1}(x \circ r) \leq \mathsf{nK}^{t_2}(x \circ r)$. Hence we get

$$\Pr_{r \sim \mathcal{U}_\ell}\left[\mathsf{nK}^{t_1}(x \circ r) \leq s \right] \geq \gamma_2 \geq \gamma_1.$$

Hence by the definition of ℓ-nK, $\ell\text{-}\mathsf{nK}^{t_1}_{\gamma_1}(x) \leq s$. Substituting s by $\ell\text{-}\mathsf{nK}^{t_2}_{\gamma_2}(x)$ finishes the proof. □

Lemma 39 (Monotonicity of ℓ-$\mathsf{nK}^t_\gamma(x)$ on ℓ and x)**.** *There exists a polynomial p such that the following holds. For any $\ell, \ell' \in \mathbb{N}$ with $\ell \leq \ell'$, any $\gamma \in (0,1)$, any strings $x, y \in \{0,1\}^*$, and any $t \in \mathbb{N}$, we have*

$$\ell'\text{-}\mathsf{nK}^{t+p(|x|+|y|+\ell')}_\gamma(x) \leq \ell\text{-}\mathsf{nK}^t_\gamma(x \circ y) + \log p(|x| + |y| + \ell')$$

Proof. Let $s = \ell\text{-}\mathsf{nK}^t_\gamma(x \circ y)$. For a string $r \in \{0,1\}^\ell$, we say r is good if $\mathsf{nK}^t(x \circ y \circ r) \leq s + \ell$. Then by the definition of ℓ-nK, with probability at least γ over $r \sim \mathcal{U}_\ell$, r is good. For such a good r, $\epsilon \to (x \circ y \circ r)$ can be non-deterministically computed in time t and size $s + \ell$. For any integer i, let $\widetilde{i}$ denote the binary encoding of i with each bit duplicated, concatenated with 01. Let M be the Turing machine satisfying the following property.

For strings $x, y, r, r', w \in \{0,1\}^*$,

$$M(\widetilde{|x|} \circ \widetilde{|y|} \circ r'; x \circ y \circ r; w; \epsilon) = x \circ r \circ r'$$

in $\mathsf{poly}(|x|, |y|, |r|, |r'|)$ steps.

Hence by efficient universality (Definition 12), for any $r' \in \{0,1\}^{\ell'-\ell}$, $(x \circ y \circ r) \to (x \circ r \circ r')$ is in non-deterministic time $\mathsf{poly}(|x| + |y| + \ell')$ and size $(\ell' - \ell + O(\log(|x| + |y| + \ell')))$. By efficient composition (Definition 12), when p is some large enough polynomial, for any good r, $\epsilon \to (x \circ r \circ r')$ is in non-deterministic time $t + p(|x| + |y| + \ell')$ and size $(s + \ell' + \log p(|x| + |y| + \ell'))$. Therefore, we have

$$\Pr_{\substack{r \sim \mathcal{U}_\ell \\ r' \sim \mathcal{U}_{\ell'-\ell}}} \left[\mathsf{nK}^{t+p(|x|+|y|+\ell')}(x \circ r \circ r') \leq s + \ell' + \log p(|x| + |y| + \ell') \right] \geq \Pr_{r \sim \mathcal{U}_\ell}[r \text{ is good}] \geq \gamma$$

In other words, $\ell'\text{-}\mathsf{nK}_\gamma^{t+p(|x|+|y|+\ell')}(x) \leq s + \log p(|x| + |y| + \ell')$. Substituting s by $\ell\text{-}\mathsf{nK}_\gamma^t(x \circ y)$ finishes the proof. □

Similarly to MINnKT, we can define the MINℓ-nKT problem for ℓ-nK.

Definition 40 (MINℓ-nKT)**.** We define MINℓ-nKT as the promise problem $(\mathcal{YES}, \mathcal{NO})$ such that

$$\begin{cases} \ell\text{-}\mathsf{nK}_{b/a}^t(x) \leq s \implies (x, 1^s, 1^t, 1^\ell, 1^a, 1^b) \in \mathcal{YES}, \\ \ell\text{-}\mathsf{nK}_{(b-1)/a}^t(x) > s \implies (x, 1^s, 1^t, 1^\ell, 1^a, 1^b) \in \mathcal{NO}. \end{cases}$$

A basic property of MINℓ-nKT is that its easiness follows from the easiness of MINnKT.

Lemma 41. *If* MINnKT $\in$ BPP*, then* MINℓ-nKT $\in$ prBPP.

Proof. Suppose we are given an instance $(x, 1^s, 1^t, 1^\ell, 1^a, 1^b)$ that is in the promised set of inputs for MINℓ-nKT, and we want to decide whether it is in $\mathcal{YES}$ or $\mathcal{NO}$. If it is in $\mathcal{YES}$, then we have $\ell\text{-}\mathsf{nK}_{b/a}^t(x) \leq s$, and by the definition of ℓ-nK, we get

$$\Pr_{r \sim \mathcal{U}_\ell} \left[\mathsf{nK}^t(x \circ r) \leq s + \ell \right] \geq \frac{b}{a}.$$

On the other hand, if the instance is in $\mathcal{NO}$, then we have $\ell\text{-}\mathsf{nK}_{(b-1)/a}^t(x) > s$, and by the definition of ℓ-nK, we get

$$\Pr_{r \sim \mathcal{U}_\ell} \left[\mathsf{nK}^t(x \circ r) \leq s + \ell \right] < \frac{b-1}{a}.$$

Therefore, in order to solve MINℓ-nKT it suffices to use an algorithm for MINnKT to distinguish the two cases described above. While this is a standard argument, for completeness, we describe the reduction and its analysis below.

Let A be a probabilistic polynomial-time algorithm that decides MINnKT with error $1/3$. Let A^m denote the algorithm that runs A for m times and takes a majority vote. By Chernoff bound, the error of A^m on any given input is bounded by $2^{-m/18}$. Our algorithm B for deciding MINℓ-nKT works as follows: it first defines $k = 18a^2$, then it takes k independent samples $R_1, \ldots, R_k \sim \mathcal{U}_\ell$. Next, for each $i \in [k]$, it computes $X_i = A^{\lceil 36 \log k \rceil}(x \circ R_i, 1^s, 1^t)$. Finally, it computes $V = (\sum_{i=1}^k X_i)/k$, and accepts if and only if $V \geq (b - 1/2)/a$.

It is not hard to see that B runs in time $\mathsf{poly}(|x|, t, s, \ell, a)$. We claim that B computes MINℓ-nKT with error at most $1/3$. Let $Y_i = \mathsf{MINnKT}(x \circ R_i, 1^s, 1^t)$. First, since the error of A^m is bounded by $2^{m/18}$, by a union bound, we have

$$\mathbf{Pr}[\exists i \in [k], X_i \neq Y_i] \leq k \cdot 2^{-36 \log k/18} = \frac{1}{k}. \tag{12}$$

That is, with probability $(1 - 1/k)$, all $\mathsf{MINnKT}(x \circ R_i, 1^s, 1^t)$ are computed correctly by $A^{\lceil 36 \log k \rceil}$. Next, if the input is in $\mathcal{YES}$, then by the definition of $\mathsf{MIN}\ell\text{-}\mathsf{nKT}$, we have $\mathbb{E}[\sum_{i=1}^{k} Y_i] \geq k \cdot b/a$. Also, Y_i are independent identically distributed random variables in $\{0, 1\}$. Therefore, using Chernoff bound, we get

$$\mathbf{Pr}\left[\sum_{i=1}^{k} Y_i < k \cdot \frac{b - 1/3}{a}\right] \leq \exp\left(-2 \cdot \frac{1}{9a^2} \cdot k\right) = e^{-4}. \tag{13}$$

Combining Equations (12) and (13) with a union bound, we have

$$\mathbf{Pr}\left[\sum_{i=1}^{k} X_i < k \cdot \frac{b - 1/3}{a}\right] \leq e^{-4} + \frac{1}{k} \leq \frac{1}{3}.$$

That is, the probability of B rejecting an input from $\mathcal{YES}$ is at most $1/3$. Similarly, one can show that the probability of B accepting an input from $\mathcal{NO}$ is at most $1/3$. Hence we conclude that $\mathsf{MIN}\ell\text{-}\mathsf{nKT} = (\mathcal{YES}, \mathcal{NO}) \in \mathsf{prBPP}$. □

4.2.2 Useful Bounds for $\ell\text{-}\mathsf{nK}^t_\gamma$

In this section, we state and prove the following lemmas for $\ell\text{-}\mathsf{nK}$, which are important ingredients in the proof of the hard direction of Theorem 1.

Lemma 42 (DP computation overhead)**.** *There exists a polynomial p such that the following holds. For any $n, m, k, t, \ell \in \mathbb{N}^+$, any $\gamma \in (0, 1)$ and any strings $x \in \{0,1\}^n, y \in \{0,1\}^m, z \in \{0,1\}^{mk}$, we have*

$$\ell\text{-}\mathsf{nK}^{t+p(nmk\ell)}_\gamma(x \circ \mathsf{DP}_k(y; z)) \leq \ell\text{-}\mathsf{nK}^t_\gamma(x \circ y) + mk + \log p(nmk\ell).$$

Proof. Let $s = \ell\text{-}\mathsf{nK}^t_\gamma(x \circ y)$. We say that a given $r \in \{0,1\}^\ell$ is good if $\mathsf{nK}^t(x \circ y \circ r) \leq s + \ell$. Then by the definition of $\ell\text{-}\mathsf{nK}$, with probability at least γ over $r \sim \mathcal{U}_l$, r is good. For a good r, $\epsilon \to (x \circ y \circ r)$ is in non-deterministic time t and size $(s + \ell)$. For any integer i, let $\widetilde{i}$ denote the binary encoding of i with each bit duplicated, concatenated with 01. Let M be the Turing machine satisfying the following.

> For any $n, m, k, \ell \in \mathbb{N}^+$, and for any strings $x \in \{0,1\}^n, y \in \{0,1\}^m, z \in \{0,1\}^{mk}, r \in \{0,1\}^\ell, w \in \{0,1\}^*$,
>
> $$M(\widetilde{n} \circ \widetilde{m} \circ \widetilde{k} \circ \widetilde{\ell} \circ z; x \circ y \circ r; w; \epsilon) = (x \circ \mathsf{DP}_k(y; z) \circ r)$$
>
> in $\mathsf{poly}(nmk\ell)$ steps.

By efficient universality (Definition 12), $(x \circ y \circ r) \to (x \circ \mathsf{DP}_k(y; a) \circ r)$ is in non-deterministic time $\mathsf{poly}(nmk\ell)$ and size $(mk + O(\log(nmk\ell)))$. Then by efficient composition (Definition 12), when p is some large enough polynomial, $\epsilon \to (x \circ \mathsf{DP}_k(y; a) \circ r)$ is in non-deterministic time $(t + p(nmk\ell))$ and size $(s + \ell + mk + \log p(nmk\ell))$. Therefore, for any good $r \in \{0,1\}^\ell$, we have

$$\mathsf{nK}^{t+p(nmk\ell)}(x \circ \mathsf{DP}_k(y; a) \circ r) \leq s + \ell + mk + \log p(nmk\ell).$$

Because the fraction of good $r \sim \mathcal{U}_\ell$ is at least γ, we conclude that

$$\ell\text{-}\mathsf{nK}^{t+p(nmk\ell)}_\gamma(x \circ \mathsf{DP}_k(y; a)) \leq s + mk + \log p(nmk\ell).$$

Substituting s by $\ell\text{-}\mathsf{nK}^t_\gamma(x \circ y)$ completes the proof. □

Lemma 43 (Highly efficient weak symmetry of information surrogate for ℓ-nK)**.** *For any $n, m, t, \ell \in \mathbb{N}^+$, any $0 < \alpha < \gamma < 1$, and any string $x \in \{0,1\}^n$, we have*

$$\Pr_{w \sim \mathcal{U}_m}\left[\ell\text{-nK}^t_\gamma(x \circ w) \geq (\ell+m)\text{-nK}^t_{\gamma-\alpha}(x) + m\right] \geq \alpha.$$

Proof. Let s be the smallest integer satisfying

$$\Pr_{w \sim \mathcal{U}_m}\left[\ell\text{-nK}^t_\gamma(x \circ w) \leq s\right] \geq 1 - \alpha.$$

Then by the definition of ℓ-nK, we have

$$\Pr_{w \sim \mathcal{U}_m}\left[\Pr_{r \sim \mathcal{U}_\ell}\left[\text{nK}^t(x \circ w \circ r) \leq s + \ell\right] \geq \gamma\right] \geq 1 - \alpha.$$

Therefore, we have

$$\Pr_{\substack{w \sim \mathcal{U}_m \\ r \sim \mathcal{U}_\ell}}\left[\text{nK}^t(x \circ w \circ r) \leq s + \ell\right] \geq \gamma(1-\alpha) \geq \gamma - \alpha.$$

By the definition of ℓ-nK, we get

$$(\ell+m)\text{-nK}^t_{\gamma-\alpha}(x) \leq s - m.$$

In other words, $s \geq (\ell+m)\text{-nK}^t_{\gamma-\alpha}(x) + m$. Now by minimality of s, we get

$$\Pr_{w \sim \mathcal{U}_m}\left[\ell\text{-nK}^t_\gamma(x \circ w) \leq s - 1\right] < 1 - \alpha.$$

In other words, $\mathbf{Pr}_{w \sim \mathcal{U}_m}[\ell\text{-nK}^t_\gamma(x \circ w) \geq s] > \alpha$. Substituting s, we get

$$\Pr_{w \sim \mathcal{U}_m}\left[\ell\text{-nK}^t_\gamma(x \circ w) \geq (\ell+m)\text{-nK}^t_{\gamma-\alpha}(x) + m\right] \geq \alpha.$$

This completes the proof. □

Lemma 44 (Chain-rule style relation between ℓ-nK and pnK)**.** *There exists a polynomial p such that the following holds. For strings $x, y \in \{0,1\}^*$ and parameters $\ell, t, t_1, t_2 \in \mathbb{N}^+$ and $\gamma, \gamma', \lambda \in (0,1)$ satisfying the following constraints*:

- $t_2 \leq \ell$,
- $t \geq t_1 + p(|x| + |y| + \ell)$,
- $(1-\gamma) \geq (1-\gamma') + (1-\lambda)$,

we have

$$\ell\text{-nK}^t_\gamma(x \circ y) \leq \ell\text{-nK}^{t_1}_{\gamma'}(x) + \text{pnK}^{t_2}_\lambda(y \mid x) + \log p(|x| + |y| + \ell).$$

Proof. The general idea is that, if $t_2 \leq \ell$, then to non-deterministically compute $x \circ y$ we can use the random ℓ-bit string generated along with x via $\ell\text{-nK}^{t_1}_{\gamma'}(x)$ to obtain y from $\text{pnK}^\ell_\lambda(y \mid x)$. Let $\ell\text{-nK}^{t_1}_{\gamma'}(x) = s_1$, and $\text{pnK}^\ell_\lambda(y \mid x) = s_2$. Without loss of generality, we assume that $s_2 \leq |x| + |y|$, because otherwise, by Lemma 37, when p is a large enough polynomial, we have $\ell\text{-nK}^{p(|x|+|y|+\ell)}_\gamma(x \circ y) \leq |x| + |y| + \log p(|x| + |y| + \ell)$, which finishes the proof. Now for any string $r \in \{0,1\}^\ell$, we define the following conditions:

(C1) r satisfies $\text{nK}^{t_1}(x \circ r) \leq s_1 + \ell$.

(C2) r satisfies $\text{nK}^\ell(y \mid x; r) \leq s_2$.

Then we have the following claim:

Claim 45. *If r satisfies both (C1) and (C2), then for a large enough polynomial p, we have* $\mathsf{nK}^t(x \circ y \circ r) \leq s_1 + s_2 + \ell + \log p(|x| + |y| + \ell)$.

Proof of Claim 45. By (C1), we have

$$\epsilon \to (x \circ r) \text{ in non-deterministic time } t_1 \text{ and size } s_1 + \ell. \tag{14}$$

By (C2), we have

$$(x; r) \to y \text{ in non-deterministic time } \ell \text{ and size } s_2. \tag{15}$$

Let Π be the program of Equation (15). Let M be the Turing machine which gives efficient simulation for the computational model (Definition 12). Then, for a large enough polynomial p_0, since $s_2 \leq |x| + |y|$, we have

- $\forall w \in \{0,1\}^\ell$, $M(\Pi; x; w; r)$ outputs either y or $\perp$ in $p_0(|x| + |y| + \ell)$ steps,
- $\exists w \in \{0,1\}^\ell$, $M(\Pi; x; w; r)$ outputs y in $p_0(|x| + |y| + \ell)$ steps.

For any integer i, let $\widetilde{i}$ denote the binary encoding of i with each bit duplicated, concatenated with 01. Let M' be the Turing machine satisfying the following condition.

> For any $\Pi, x, w, r \in \{0,1\}^*$ and any $\ell \in \mathbb{N}$ satisfying $\ell \leq |w|$, on input $(\widetilde{|x|} \circ \widetilde{\ell} \circ \Pi; x \circ r; w; \epsilon)$, M' runs $M(\Pi; x; w_\ell; r)$ to obtain y, where w_ℓ denotes the length-ℓ prefix of w. If $y \neq \perp$, it outputs $x \circ y \circ r$; otherwise it outputs $\perp$.

Then for a large enough polynomial p_1, for any $t' \geq t$ we have

- $\forall w \in \{0,1\}^{t'}$, $M'(\widetilde{|x|} \circ \widetilde{\ell} \circ \Pi; x \circ r; w; \epsilon)$ outputs either $(x \circ y \circ r)$ or $\perp$ in $p_1(|x| + |y| + \ell)$ steps,
- $\exists w \in \{0,1\}^{t'}$, $M'(\widetilde{|x|} \circ \widetilde{\ell} \circ \Pi; x \circ r; w; \epsilon)$ outputs $(x \circ y \circ r)$ in $p_1(|x| + |y| + \ell)$ steps.

By efficient universality (Definition 12), when p_2 is a large enough polynomial, we get

$$(x \circ r) \to (x \circ y \circ r) \text{ in non-deterministic time } p_2(|x| + |y| + \ell) \text{ and size } (s_2 + \log p_2(|x| + |y| + \ell)). \tag{16}$$

By applying efficient composition (Definition 12) to Equations 14 and 16, when p is a large enough polynomial, we get

$$\epsilon \to (x \circ y \circ r) \text{ in non-deterministic time } (t_1 + p(|x| + |y| + \ell)) \text{ and size } (s_1 + s_2 + \ell + \log p(|x| + |y| + \ell)).$$

Since $t \geq t_1 + p(|x| + |y| + \ell)$, we have

$$\mathsf{nK}^t(x \circ y \circ r) \leq s_1 + s_2 + \ell + \log p(|x| + |y| + \ell).$$

This completes the proof of the claim. ◇

By the definition of ℓ-$\mathsf{nK}_{\gamma'}$ and pnK_λ, for $r \sim \mathcal{U}_\ell$, r satisfies (C1) with probability at least γ', and r satisfies (C2) with probability at least λ. By a union bound, with probability at least $1-(1-\gamma')-(1-\lambda) \geq \gamma$, r satisfies both (C1) and (C2). Hence by the definition of ℓ-nK and Claim 45, we have

$$\ell\text{-}\mathsf{nK}^t_\gamma(x \circ y) \leq \ell\text{-}\mathsf{nK}^{t_1}_{\gamma'}(x) + \mathsf{pnK}^\ell_\lambda(y \mid x) + \log p(|x| + |y| + \ell).$$

Since $t_2 \leq \ell$, by randomness and non-determinism extension (Definition 12), we have $\mathsf{pnK}^\ell_\lambda(x) \leq \mathsf{pnK}^{t_2}_\lambda(x)$. Hence we get

$$\ell\text{-}\mathsf{nK}^t_\gamma(x \circ y) \leq \ell\text{-}\mathsf{nK}^{t_1}_{\gamma'}(x) + \mathsf{pnK}^{t_2}_\lambda(y \mid x) + \log p(|x| + |y| + \ell),$$

as desired. □

4.2.3 Main Lemma: Bounding Conditional pK

Lemma 46 (Main Lemma)**.** *Suppose* $\mathsf{MINnKT} \in \mathsf{BPP}$. *Then there exists a polynomial* p *and an integer* $N_0 \in \mathbb{N}^+$ *such that the following holds. For any* $n, m, \tau, \ell, t, b \in \mathbb{N}^+$ *and any string* $x \in \{0,1\}^n, y \in \{0,1\}^m$ *satisfying the following constraints*:

- $2/3 < b/n^3 < 1$,
- $N_0 \leq n \leq m \leq \tau$.
- $\tau \leq \ell \leq \tau^4$,
- $t \geq 2p(\tau)$,

we have

$$\mathsf{pK}^{p(t)}(y \mid x) \leq \mathsf{pnK}^\tau_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{nK}^{t-p(\tau)}_{(b+1)/n^3}(x) - (\ell+\tau^3)\text{-}\mathsf{nK}^{t+p(\tau)}_{(b-2)/n^3}(x) + \log p(t).$$

Proof. Let p_1, p_2, p_3, p_4 be the polynomials defined in Lemmas 37, 39, 42 and 44 respectively. Without loss of generality, we assume they are all monotone. We define k as

$$k := \ell\text{-}\mathsf{nK}^{t-p_3(\tau^8)}_{b/n^3}(x \circ y) - (\ell+\tau^3)\text{-}\mathsf{nK}^{t+p_2(\tau^6)}_{(b-2)/n^3}(x) + \lceil \log p_3(\tau^8) \rceil + \lceil \log p_2(\tau^6) \rceil + 1. \tag{17}$$

First, we argue that k is a positive integer. By our assumptions on the parameters, when N_0 is large enough, we have $n + m + \ell + \tau^3 \leq \tau^6$. Hence we have

$$\begin{aligned}
k &\geq \ell\text{-}\mathsf{nK}^{t-p_3(\tau^8)}_{b/n^3}(x \circ y) - (\ell+\tau^3)\text{-}\mathsf{nK}^{t+p_2(\tau^6)}_{(b-2)/n^3}(x) + \log p_2(\tau^6) + 1 \\
&\geq \ell\text{-}\mathsf{nK}^{t-p_3(\tau^8)}_{b/n^3}(x \circ y) - \ell\text{-}\mathsf{nK}^{t}_{(b-2)/n^3}(x \circ y) + 1 && \text{(By Lemma 39)} \\
&\geq \ell\text{-}\mathsf{nK}^{t-p_3(\tau^8)}_{b/n^3}(x \circ y) - \ell\text{-}\mathsf{nK}^{t-p_3(\tau^8)}_{b/n^3}(x \circ y) + 1 && \text{(By Lemma 38)} \\
&= 1. && (18)
\end{aligned}$$

Next, we upper bound and lower bound the value of k. We will then use the DPG reconstruction lemma to finish the proof.

Upper Bound for k. Since $n \le m \le \tau \le \ell$, by Lemma 44, we get

$$\ell\text{-}\mathsf{nK}^{t-p_3(\tau^8)}_{b/n^3}(x \circ y) \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{nK}^{t-p_3(\tau^8)-p_4(\tau^6)}_{(b+1)/n^3}(x) + \log p_4(\tau^6). \tag{19}$$

Therefore, by combining Equations (17) and (19), we have

$$k \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{nK}^{t-p_3(\tau^8)-p_4(\tau^6)}_{(b+1)/n^3}(x) - (\ell+\tau^3)\text{-}\mathsf{nK}^{t+p_2(\tau^6)}_{(b-2)/n^3}(x) + \log p_3(\tau^8) + \log p_2(\tau^6) + \log p_4(\tau^6) + 3.$$

Therefore, when q_1 is a large enough polynomial, we get

$$k \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{nK}^{t-q_1(\tau)}_{(b+1)/n^3}(x) - (\ell+\tau^3)\text{-}\mathsf{nK}^{t+q_1(\tau)}_{(b-2)/n^3}(x) + \log q_1(\tau). \tag{20}$$

Lower Bound for k. We first show a loose upper bound on k when p and N_0 are large enough. Based on this, we give a lower bound for k. When p is a large enough polynomial such that $p(z) \ge p_3(z^8) + p_1(z^6)$, $t \ge p(\tau) \ge p_3(\tau^8) + p_1(\tau^6)$. By Lemma 37, $\ell\text{-}\mathsf{nK}^{t-p_3(\tau^8)}_{b/n^3}(x \circ y) \le n + m + \log p_1(n+m+\ell) \le 2\tau + \log p_1(\tau^6)$. By Lemma 36, we also have $(\ell+\tau^3)\text{-}\mathsf{nK}^{t+p_2(\tau^6)}_{(b-2)/n^3}(x) \ge -2$. Therefore, we have a loose upper bound for k:

$$k \le 2\tau + \log p_1(\tau^6) + \log p_3(\tau^8) + \log p_2(\tau^6) + 5.$$

Since $\tau \ge N_0$, when N_0 is a large enough constant, we have $k \le 3\tau$. Therefore, we get $mk + k \le \tau^3$, and $\tau^6 \ge \ell + \tau^3 + n$. By Lemma 39, we get

$$\begin{aligned}(\ell+\tau^3)\text{-}\mathsf{nK}^{t+p_2(\tau^6)}_{(b-2)/n^3}(x) - \log p_2(\tau^6) &\le (\ell+\tau^3)\text{-}\mathsf{nK}^{t+p_2(\ell+\tau^3+n)}_{(b-2)/n^3}(x) - \log p_2(\ell+\tau^3+n)\\ &\le (\ell+mk+k)\text{-}\mathsf{nK}^{t}_{(b-2)/n^3}(x).\end{aligned}$$

We also have $\tau^8 \ge nmk\ell$, which gives us

$$\ell\text{-}\mathsf{nK}^{t-p_3(\tau^8)}_{b/n^3}(x \circ y) + \log p_3(\tau^8) \ge \ell\text{-}\mathsf{nK}^{t-p_3(nmk\ell)}_{b/n^3}(x \circ y) + \log p_3(nmk\ell).$$

Therefore, we can now lower bound k:

$$\begin{aligned}k &= \ell\text{-}\mathsf{nK}^{t-p_3(\tau^8)}_{b/n^3}(x \circ y) - (\ell+\tau^3)\text{-}\mathsf{nK}^{t+p_2(\tau^6)}_{(b-2)/n^3}(x) + \lceil \log p_3(\tau^8) \rceil + \lceil \log p_2(\tau^6) \rceil + 1\\ &\ge \ell\text{-}\mathsf{nK}^{t-p_3(nmk\ell)}_{b/n^3}(x \circ y) + \log p_3(nmk\ell) - (\ell+mk+k)\text{-}\mathsf{nK}^{t}_{(b-2)/n^3}(x) + 1.\end{aligned} \tag{21}$$

Distinguisher for the Direct Product Generator. By Equation (18), when N_0 is large enough, $k \ge 1$ always holds. Here we define an algorithm D to distinguish $\mathsf{DP}_k(y; \mathcal{U}_{mk})$ from $\mathcal{U}_{mk+k}$. We set the threshold

$$s = \ell\text{-}\mathsf{nK}^{t-p_3(nmk\ell)}_{b/n^3}(x \circ y) + mk + \lfloor \log p_3(nmk\ell) \rfloor. \tag{22}$$

By Lemma 41, $\mathsf{MINnKT} \in \mathsf{BPP}$ implies $\mathsf{MIN}\ell\text{-}\mathsf{nKT} \in \mathsf{prBPP}$. Let B be the polynomial-time algorithm that decides $\mathsf{MIN}\ell\text{-}\mathsf{nKT}$ with error $1/n^4$. We define the distinguisher D, which depends on x, as follows:

1. D gets input $z \in \{0,1\}^{mk+k}$.
2. D outputs $B(x \circ z, 1^s, 1^t, 1^\ell, 1^{n^3}, 1^b)$.

In fact, if $z \sim \mathsf{DP}_k(y; \mathcal{U}_{mk})$, then by Lemma 42 and Equation (22), we get

$$\Pr_{w \sim \mathcal{U}_{mk}} \left[\ell\text{-}\mathsf{nK}^t_{b/n^3}(x \circ \mathsf{DP}_k(y; w)) \le s \right] = 1.$$

Therefore, we have

$$\Pr_{\substack{w \sim \mathcal{U}_{mk} \\ \text{Randomness of } D}} [D(\mathsf{DP}_k(y; w)) = 1] \ge 1 - \frac{1}{n^4}. \tag{23}$$

On the other hand, if $z \sim \mathcal{U}_{mk+k}$, then by Lemma 43, we get

$$\Pr_{w \sim \mathcal{U}_{mk+k}} \left[\ell\text{-}\mathsf{nK}^t_{(b-1)/n^3}(x \circ w) \ge (\ell + mk + k)\text{-}\mathsf{nK}^t_{(b-2)/n^3}(x) + mk + k \right] \ge \frac{1}{n^3}.$$

But by Equation (21) and Equation (22), we have

$$(\ell + mk + k)\text{-}\mathsf{nK}^t_{(b-2)/n^3}(x) + mk + k \ge s + 1.$$

Combining these two inequalities, we get

$$\Pr_{w \sim \mathcal{U}_{mk+k}} \left[\ell\text{-}\mathsf{nK}^t_{(b-1)/n^3}(x \circ w) \ge s + 1 \right] \ge \frac{1}{n^3}.$$

Therefore, we have

$$\Pr_{\substack{w \sim \mathcal{U}_{mk+k} \\ \text{Randomness of } D}} [D(w) = 1] \le 1 - \frac{1}{n^3} + \frac{1}{n^4}. \tag{24}$$

By comparing Equation (23) and Equation (24), we see that D distinguishes $\mathsf{DP}_k(y; \mathcal{U}_{mk})$ from $\mathcal{U}_{mk+k}$ with probability $1/n^3 - 2/n^4 \ge 1/n^4$. Then by Corollary 24, there exists some polynomial p_{DP} such that

$$\mathsf{pK}^{p_{\mathsf{DP}}(t), \|D}(y) \le k + \log p_{\mathsf{DP}}(t).$$

We can store the program of D in the description, as well as t, n, m, k, s, ℓ, b, which takes no more than $O(\log t)$ bits in total. If we have x, then we can simulate D, which runs in time $\mathsf{poly}(t)$, and answer the oracle queries. Therefore, when q_2 is a large enough polynomial, we have

$$\mathsf{pK}^{q_2(t)}(y \mid x) \le k + \log q_2(t). \tag{25}$$

Putting It All Together. To summarize the previous paragraphs, when p is a large enough polynomial and N_0 is a large enough constant, by Equation (20) there exists some polynomial q_1 such that

$$k \le \mathsf{pnK}^\tau_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{nK}^{t-q_1(\tau)}_{(b+1)/n^3}(x) - (\ell + \tau^3)\text{-}\mathsf{nK}^{t+q_1(\tau)}_{(b-2)/n^3}(x) + \log q_1(\tau).$$

Also, by Equation (25), there exists some polynomial q_2 such that

$$k \ge \mathsf{pK}^{q_2(t)}(y \mid x) - \log q_2(t).$$

Combining these two inequalities, we get

$$\mathsf{pK}^{q_2(t)}(y \mid x) \le \mathsf{pnK}^\tau_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{nK}^{t-q_1(\tau)}_{(b+1)/n^3}(x) - (\ell + \tau^3)\text{-}\mathsf{nK}^{t+q_1(\tau)}_{(b-2)/n^3}(x) + \log q_1(\tau) + \log q_2(t).$$

Therefore, when p is a large enough polynomial, we get

$$\mathsf{pK}^{p(t)}(y \mid x) \le \mathsf{pnK}^\tau_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{nK}^{t-p(\tau)}_{(b+1)/n^3}(x) - (\ell + \tau^3)\text{-}\mathsf{nK}^{t+p(\tau)}_{(b-2)/n^3}(x) + \log p(t),$$

as desired. □

4.2.4 Proof of Theorem 34

We first state and prove the following lemma, which is then used to establish Theorem 34.

Lemma 47 (Witness compression). *Let $L = \mathsf{SAT}$, and define $L_n = L \cap \{0,1\}^n$. Suppose $L \in \mathsf{NTIME}[n^c]$. For a given $x \in \{0,1\}^n$, define $A_x \subseteq \{0,1\}^{n^c}$ to be the set of witnesses for x. If $\mathsf{MINnKT} \in \mathsf{BPP}$, then there exists a polynomial p such that for any n, any $x \in L_n$ and any $y \in A_x$, we have*

$$\mathsf{pK}^{p(n)}(y \mid x) \leq \log |A_x| + \log p(n).$$

Proof. If $x \in L_n$, then by Theorem 4, there exists some polynomial p_{LC} such that for any $y \in A_x$, $\mathsf{pnK}^{p_{\mathsf{LC}}(n^c),A_x}(y \mid x) \leq \log |A_x| + \log p_{\mathsf{LC}}(n^c)$. Since A_x is decidable in time n^c given x, and by Lemma 29 we can amplify the success probability, we get that there exists some polynomial p_1 such that $p_1(z) \geq z^d$, for a large enough constant $d > 0$, and

$$\mathsf{pnK}^{p_1(n)}_{1-1/n^3}(y \mid x) \leq \log |A_x| + \log p_1(n). \tag{26}$$

Let p_2 be the polynomial stated in Lemma 46. Let $m = n^c, \tau = p_1(n)$. Then by Lemma 46, when n is large enough, for any $\ell, t, b \in \mathbb{N}^+$ satisfying $2/3 < b/n^3 < 1$, $\tau \leq \ell \leq \tau^4$, $t \geq p_2(\tau)$, and any string $x \in \{0,1\}^n$, $y \in \{0,1\}^{n^c}$, we have

$$\mathsf{pK}^{p_2(t)}(y \mid x) \leq \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{nK}^{t-p_2(\tau)}_{(b+1)/n^3}(x) - (\ell+\tau^3)\text{-}\mathsf{nK}^{t+p_2(\tau)}_{(b-2)/n^3}(x) + \log p_2(t).$$

Now, setting t to be some polynomial of n, if we could bound $\left(\ell\text{-}\mathsf{nK}^{t-p_2(\tau)}_{(b+1)/n^3}(x) - (\ell+\tau^3)\text{-}\mathsf{nK}^{t+p_2(\tau)}_{(b-2)/n^3}(x)\right)$ by a constant, then using Equation (26) we would be done. Even though we don't know how to do this directly, we can use a telescoping sum trick. More specifically, let p_3 be the polynomial stated in Lemma 37. We define three sequences $\{\ell_i\}_{i=0}^n$, $\{t_i\}_{i=0}^n$ and $\{b_i\}_{i=0}^n$ recursively:

- $\ell_0 = \tau^3$; $\ell_i = \ell_{i-1} + \tau^3$, $\forall i \in [n]$.
- $t_0 = 2p_2(\tau) + p_3(\tau^6)$; $t_i = t_{i-1} + 2p_2(\tau)$, $\forall i \in [n]$.
- $b_0 = \lfloor 3n^3/4 \rfloor$; $b_i = b_{i-1} - 3$, $\forall i \in [n]$.

It is not hard to verify that for any $i \in \{0, \ldots, n\}$, ℓ_i, t_i, b_i satisfy the conditions of Lemma 46. Hence we get

$$\mathsf{pK}^{p_2(t_i)}(y \mid x) \leq \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \ell_i\text{-}\mathsf{nK}^{t_i-p_2(\tau)}_{(b_i+1)/n^3}(x) - (\ell_i+\tau^3)\text{-}\mathsf{nK}^{t_i+p_2(\tau)}_{(b_i-2)/n^3}(x) + \log p_2(t_i). \tag{27}$$

Now using a telescoping argument, we get

$$\begin{aligned}
\ell_0\text{-}\mathsf{nK}^{t_0-p_2(\tau)}_{(b_0+1)/n^3}(x) - \ell_n\text{-}\mathsf{nK}^{t_n-p_2(\tau)}_{(b_n+1)/n^3}(x) &= \sum_{i=0}^{n-1}\left(\ell_i\text{-}\mathsf{nK}^{t_i-p_2(\tau)}_{(b_i+1)/n^3}(x) - \ell_{i+1}\text{-}\mathsf{nK}^{t_{i+1}-p_2(\tau)}_{(b_{i+1}+1)/n^3}(x)\right) \\
&= \sum_{i=0}^{n-1}\left(\ell_i\text{-}\mathsf{nK}^{t_i-p_2(\tau)}_{(b_i+1)/n^3}(x) - (\ell_i+\tau^3)\text{-}\mathsf{nK}^{t_i+p_2(\tau)}_{(b_i-2)/n^3}(x)\right)
\end{aligned}$$

By Lemma 37, we have

$$\ell_0\text{-}\mathsf{nK}^{t_0-p_2(\tau)}_{(b_0+1)/n^3}(x) \leq (\tau^3)\text{-}\mathsf{nK}^{p_3(\tau^6)}_{(b_0+1)/n}(x) \leq n + \log p_3(\tau^6),$$

which is at most $1.5n$ for all large enough n. On the other hand, it follows from Lemma 36 and the definition of b_n that $-\ell_n\text{-}\mathsf{nK}^{t_n-p_2(\tau)}_{(b_n+1)/n^3}(x) = O(1)$. This implies that

$$\ell_0\text{-}\mathsf{nK}^{t_0-p_2(\tau)}_{(b_0+1)/n^3}(x) - \ell_n\text{-}\mathsf{nK}^{t_n-p_2(\tau)}_{(b_n+1)/n^3}(x) \le 2n,$$

provided that n is large enough. By averaging, there must be some $i \in \{0,\dots,n-1\}$ such that

$$\left(\ell_i\text{-}\mathsf{nK}^{t_i-p_2(\tau)}_{(b_i+1)/n^3}(x) - (\ell_i+\tau^3)\text{-}\mathsf{nK}^{t_i+p_2(\tau)}_{(b_i-2)/n^3}(x)\right) \le 2.$$

For such an i, by Equation (27) we have

$$\mathsf{pK}^{p_2(t_i)}(y \mid x) \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \log p_2(t_i) + 2.$$

Note that $t_i \le t_n$. Then the above yields

$$\mathsf{pK}^{p_2(t_n)}(y \mid x) \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \log p_2(t_n) + 2.$$

Combining this with Equation (26) and using that $\tau = p_1(n)$, we get that for all large enough n, for any $x \in L_n$ and $y \in A_x$,

$$\mathsf{pK}^{p_2(t_n)}(y \mid x) \le \log|A_x| + \log p_1(n) + \log p_2(t_n) + 2.$$

Notice that $t_n \le (2n+2)p_2(\tau)+p_3(\tau^6) = (2n+2)p_2(p_1(n))+p_3(p_1^6(n))$, which is polynomially bounded by n. Hence for a large enough polynomial p, we have

$$\mathsf{pK}^{p(n)}(y \mid x) \le \log|A_x| + \log p(n),$$

which concludes the proof. □

Proof of Theorem 34 from Lemma 47. Our randomized polynomial-time algorithm B for deciding $L = \mathsf{SAT}$ works as follows:

1. B gets $x \in \{0,1\}^n$ as input.
2. B takes a sample y from $\mathsf{USamp}(1^{n^c}, 1^{p(n)}, x)$.
3. B checks if y is a witness for x.
4. B repeats steps 2 and 3 for $O(n^c \cdot p(n))$ rounds. If at any round, y is indeed a witness for x, B accepts; otherwise B rejects.

If $x \notin L_n$, then $A_x = \varnothing$, so B never accepts x. If $x \in L_n$, then by Lemma 47 and Proposition 26, for any $y \in A_x$, the probability that $\mathsf{USamp}(1^{n^c}, 1^{p(n)}, x) = y$ is at least $\Omega(1/(n^c \cdot p(n) \cdot |A_x|))$. Adding up the probabilities for each $y \in A_x$, we get

$$\Pr_{y\sim\mathsf{USamp}(1^{n^c},1^{p(n)},x)}[y \in A_x] \ge \Omega\left(\frac{1}{n^c \cdot p(n)}\right).$$

Therefore, if we sample $O(n^c \cdot p(n))$ times, we can obtain some $y \in A_x$ with probability $\Omega(1)$. Hence B accepts x with probability $\Omega(1)$. Since this probability can be easily amplified, we conclude that $L \in \mathsf{BPP}$. Finally, since $L = \mathsf{SAT}$ is NP-complete, we conclude that $\mathsf{NP} \subseteq \mathsf{BPP}$. □

5 Average-Case Hardness of MINnKT

In this section, we show Theorem 2 and Theorem 3. We begin with some useful tools.

5.1 Technical Tools

We will in fact consider an assumption that is weaker than MINnKT being average-case easy. More specifically, let "$(\mathsf{coMINnKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$" denote the following statement.

> There exists a constant $c > 0$, a polynomial ρ, and a probabilistic polynomial-time algorithm A such that the following hold for all sufficiently large n, all $t \geq \rho(n)$, and all $s \leq n - c \cdot \log t$:
>
> 1. For every $x \in \{0,1\}^n$ with $\mathsf{nK}^t(x) \leq s$, we have $\mathbf{Pr}_A[A(x, 1^s, 1^t) = 1] \geq \frac{2}{3}$.
> 2. With probability at least $1 - 1/t$ over $x \sim \{0,1\}^n$, we have $\mathbf{Pr}_A[A(x, 1^s, 1^t) = 0] \geq \frac{2}{3}$.

Also, let "$(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$" denote the analogous statement where nK^t is replaced by K^t.

We first state some results that are implicit in prior work, e.g., [Hir18, Hir20b, GK22, Hir22c, HKLO24]. We omit the details of the proofs since no new ideas are needed.

Lemma 48. *The following holds.*

$$(\mathsf{MINnKT}, \mathcal{U}) \in \mathsf{AvgBPP} \implies (\mathsf{coMINnKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP} \implies (\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}.$$

Proof Sketch. The proof of the first implication can be easily adapted from that of [HKLO24, Proposition 11]. The high-level idea is that if there exists an errorless heuristic scheme that computes MINnKT, then we can replace $\perp$ with 1 to obtain an algorithm that still accepts all strings with small nK^t-complexity while rejecting a large fraction of random strings. This follows from the fact that a random string has large nK^t-complexity, and the new algorithm will only err on a small fraction of these strings.

The second implication follows from the fact that $\mathsf{nK}^t(x) \leq \mathsf{K}^t(x)$ for every x and t. □

Lemma 49 (Symmetry of Information for pK^t; See, e.g., [HKLO24, Lemma 36])**.** *If $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$, then there exist polynomials p_{Sol} and p_0 such that for all sufficiently large $x, y \in \{0,1\}^*$ and every $t \geq p_0(|x| + |y|)$,*

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

Lemma 50. *If $(\mathsf{coMINKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$, then there exist a constant $c > 0$, a polynomial p_{dth} and an algorithm $\mathsf{Approx\text{-}depth}$ that, on input $(x, 1^{t_1}, 1^{t_2}, 1^k)$, where $x \in \{0,1\}^n$, $t_1, t_2, k \in \mathbb{N}$ with $t_1, t_2 \geq cn$, runs in time $\mathsf{poly}(n, t_1, t_2, k)$ and with probability $1 - 2^{-k}$ outputs an integer s such that*

$$\mathsf{pK}^{p_{\mathsf{dth}}(t_1)}(x) - \mathsf{pK}^{t_2}(x) \leq s \leq \mathsf{pK}^{t_1}(x) - \mathsf{pK}^{p_{\mathsf{dth}}(t_2)}(x) + \log p_{\mathsf{dth}}(t_1) + \log p_{\mathsf{dth}}(t_2).$$

Proof Sketch. The proof ideas are similar to those in [HKLO24, Lemma 47]. The main difference is that, instead of using a generator with a rK^t-style reconstruction and sub-optimal advice complexity as in the proof of [HKLO24, Lemma 47], we employ the direct product generator, which has a pK^t-style reconstruction and optimal advice complexity. □

Lemma 51. *If $(\mathsf{coMINnKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$, then there exists a polynomial p_1 such that for all $z \in \{0,1\}^*$ and $t \in \mathbb{N}$,*

$$\mathsf{pK}^{p_1(t)}(z) \leq \mathsf{pnK}^t(z) + \log p_1(t).$$

Proof Sketch. The proof can be easily adapted from that of Lemma 56, by considering x to be the empty string in the statement of Lemma 56. □

The following is the main technical lemma for this section.

Lemma 52. *If* $(\mathsf{coMINnKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$*, then there exist polynomials* p *and* q *such that for all* $x, y \in \{0,1\}^*$ *and all* $t \geq q(|x|, |y|)$*,*

$$\mathsf{pK}^{p(t)}(y \mid x) \leq \mathsf{pnK}^t(y \mid x) + \mathsf{pK}^t(x) - \mathsf{pK}^{p(t)}(x) + \log p(t).$$

Proof. Assume $(\mathsf{coMINnKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$. Fix $x, y \in \{0,1\}^*$ and $t \geq q(|x|, |y|)$, where q is a sufficiently large polynomial.

First of all, note that by Lemma 48 and Lemma 49, we get symmetry of information for pK^t. Now let p_1 be the polynomial from Lemma 51. We have

$$\begin{aligned}
\mathsf{pK}^{p_{\mathsf{Sol}}(p_1(2t))}(y \mid x) &\leq \mathsf{pK}^{p_1(2t)}(x, y) - \mathsf{pK}^{p_{\mathsf{Sol}}(p_1(2t))}(x) + \log p_{\mathsf{Sol}}(p_1(2t)) \\
&\leq \mathsf{pnK}^{2t}(x, y) - \mathsf{pK}^{p_{\mathsf{Sol}}(p_1(2t))}(x) + \log p_{\mathsf{Sol}}(p_1(2t)) && \text{(by Lemma 51)} \\
&\leq \mathsf{pnK}^t(y \mid x) + \mathsf{pnK}^t(x) - \mathsf{pK}^{p_{\mathsf{Sol}}(p_1(2t))}(x) + \log p_{\mathsf{Sol}}(p_1(2t)) \\
&\leq \mathsf{pnK}^t(y \mid x) + \mathsf{pK}^t(x) - \mathsf{pK}^{p_{\mathsf{Sol}}(p_1(2t))}(x) + \log p_{\mathsf{Sol}}(p_1(2t)).
\end{aligned}$$

Let p be a sufficiently large polynomial, the above yields

$$\mathsf{pK}^{p(t)}(y \mid x) \leq \mathsf{pnK}^t(y \mid x) + \mathsf{pK}^t(x) - \mathsf{pK}^{p(t)}(x) + \log p(t),$$

as desired. □

5.2 Proof of Theorem 2

In this subsection, we prove Theorem 2.

Proof of Theorem 2. Let $L \in \mathsf{NP}$. For an instance $x \in \{0,1\}^n$, let A_x denote the set of witnesses of x (with respect to some fixed verifier). Without loss of generality, we assume that every string in A_x has a length of exactly m, where $m = \mathsf{poly}(n)$. Let $\mathcal{D} := \{\mathcal{D}_n\}_{n \in \mathbb{N}}$ be a polynomial-time samplable distribution family.

Assuming $(\mathsf{MINnKT}, \mathcal{U}) \in \mathsf{AvgBPP}$, we show an errorless heuristic scheme for solving L over $\mathcal{D}$. The idea is to employ Lemma 52 and show that $\mathsf{pK}^{p(t)}(y \mid x) \leq \mathsf{pnK}^t(y \mid x) + O(\log n)$ *for most instances* $x \sim \mathcal{D}$. Then we can use language compression for pnK^t (Theorem 4) and follow the argument as shown in the proof of Theorem 10 to obtain an witness for those instances, using the universal sampler. For this, we show that computational depth, $\mathsf{pK}^t(x) - \mathsf{pK}^{p(t)}(x)$ as appear in Lemma 52, is small for most x. Moreover, in order to make the algorithm errorless, we need to be able to recognize the "bad" x's whose computational depth is not small. More specifically, we need the following depth certifying algorithm.

Claim 53. *Let* p *be the polynomial in Lemma 52. There exists a probabilistic polynomial-time algorithm* $\mathsf{Certify}$*, a polynomial* p_1*, and a constant* $d > 0$ *such that for every* $n, k \in \mathbb{N}$ *and* $t \geq p_1(n)$*, the following holds with probability* $1 - 2^{-k}$ *over the internal randomness of* $\mathsf{Certify}$*:*

1. *For every* $x \in \{0,1\}^n$*, if* $\mathsf{pK}^t(x) - \mathsf{pK}^{p(t)}(x) > d \cdot \log t + \log k$*, then* $\mathsf{Certify}(x, 1^t, 1^k) = 0$*.*
2. *Also,*

$$\Pr_{x \sim \mathcal{D}_n}[\mathsf{Certify}(x, 1^t, 1^k) = 1] \geq 1 - \frac{1}{2k}.$$

Proof of Claim 53. Let Approx-depth be the algorithm in Lemma 50, and let $d > 0$ be a constant specified later. We define Certify as follows.

> On input $(x, 1^t, 1^k)$, compute $s := \mathsf{Approx\text{-}depth}(x, 1^{p_{\mathsf{dth}}^{-1}(t)}, 1^{p(t)}, 1^{n+k})$. If $s \le d \cdot \log t + \log k$, accept; otherwise, reject.

By the correctness of the algorithm Approx-depth (Lemma 50) and a union bound over the set of n-bit strings, it is easy to see that with probability at least $1 - 2^{-k}$, it holds that

$$\mathsf{pK}^t(x) - \mathsf{pK}^{p(t)}(x) \le s \le \mathsf{pK}^{p_{\mathsf{dth}}^{-1}(t)}(x) - \mathsf{pK}^{p_{\mathsf{dth}}(p(t))}(x) + \log(t) + \log p_{\mathsf{dth}}(p(t)).$$

The first property of Certify stated in the claim follows directly from the lower bound of s above.

For the second property, we use the upper bound of s and show that with probability at least $1 - 1/k$ over $x \sim \mathcal{D}_n$, it holds that the quantify

$$\mathsf{pK}^{p_{\mathsf{dth}}^{-1}(t)}(x) - \mathsf{pK}^{p_{\mathsf{dth}}(p(t))}(x) + \log(t) + \log p_{\mathsf{dth}}(p(t)) \tag{28}$$

is at most $\le d \cdot \log t + \log k$.

First of all, by the coding theorem for pK^t (Theorem 20), we have for every $x \in \mathsf{Support}(\mathcal{D}_n)$

$$\mathsf{pK}^{p_{\mathsf{dth}}^{-1}(t)}(x) \le \log \frac{1}{\mathcal{D}_n(x)} + O(\log n), \tag{29}$$

provided that $t \ge p_1(n)$ for some sufficiently large polynomial p_1.

On the other hand, by Lemma 19, we have

$$\Pr_{x \sim \mathcal{D}_n}\left[\mathsf{K}(x) \ge \log \frac{1}{\mathcal{D}_n(x)} - O(\log n) - \log 2k\right] \ge 1 - \frac{1}{2k}. \tag{30}$$

It follows that with probability at least $1 - 1/(2k)$ over $x \sim \mathcal{D}_n$, we get

$$\begin{aligned}
&\mathsf{pK}^{p_{\mathsf{dth}}^{-1}(t)}(x) - \mathsf{pK}^{p_{\mathsf{dth}}(p(t))}(x) \\
&\le \mathsf{pK}^{p_{\mathsf{dth}}^{-1}(t)}(x) - \mathsf{K}(x) + O(\log p_{\mathsf{dth}}(p(t))) && \text{(by Lemma 18)} \\
&\le \left(\log \frac{1}{\mathcal{D}_n} + O(\log n)\right) - \left(\log \frac{1}{\mathcal{D}_n(x)} - O(\log n) - \log 2k\right) + O(\log p_{\mathsf{dth}}(p(t))) \\
& && \text{(by Equations (29) and (30))} \\
&\le (d/2) \cdot \log t + \log k,
\end{aligned}$$

provided that d is sufficiently large. Finally, by plugging the above into Equation (28), we get that Equation (28) (and hence s) is upper bounded by $d \cdot \log t + \log k$. This completes the proof of the second property of Certify and hence the claim. ◇

Before describing our final heuristic scheme for solving L, we show how we can solve L for those x whose computational depth is small. Define the following procedure B.

> On input x, run $\mathsf{USamp}(1^m, 1^{p(p_2(n))}, x)$, where $p_2 \ge \max\{p_{\mathsf{LC}}, q\}$, p_{LC} is the polynomial in Theorem 4 and p, q are the polynomials in Lemma 52. Let y be the output of USamp, if y is an L-witness of x, accept. Otherwise, reject.

Consider the case when the following condition holds:

$$\mathsf{pK}^{p_2(n)}(x) - \mathsf{pK}^{p_1(p_2(n))}(x) + \log p(t) \leq d \cdot \log p_2(n) + \log k. \tag{31}$$

Then we have

$$\begin{aligned}
\mathsf{pK}^{p(p_2(n))}(y \mid x) &\leq \mathsf{pnK}^{p_2(n)}(y \mid x) + \mathsf{pK}^{p_2(n)}(x) - \mathsf{pK}^{p(p_2(n))}(x) + \log p(p_2(n)) && \text{(by Lemma 52)} \\
&\leq \log |A_x| + \log p_2(n) + \mathsf{pK}^{p_2(n)}(x) - \mathsf{pK}^{p(p_2(n))}(x) + \log p(p_2(n)) && \text{(by Theorem 4)} \\
&\leq \log |A_x| + \log p_2(n) + d \cdot \log p_2(n) + \log k + \log p(p_2(n)) && \text{(by Equation (31))} \\
&\leq \log |A_x| + d' \cdot \log n + \log k,
\end{aligned}$$

where $d' > 0$ is a sufficiently large constant. Then by Lemma 32, we get that A solves L on x with probability at least $1/\mathsf{poly}(n)$. Using standard techniques, we can amply the success probability of B to be at least $9/10$.

We now describe an errorless average-case algorithm for solving L. Let A be the following algorithm.

> On input $(x, 1^n, 1^k)$, we first run $\mathsf{Certify}(x, 1^{p_2(n)}, 1^k)$. If it rejects, we output $\perp$. Otherwise, we run $B(x)$ and output its returned answer.

To see that the above algorithm is heuristic scheme for solving L, first note that with probability at least $1 - 2^{-k}$ over the internal randomness of Certify, we either have that Certify rejects, in which case we output $\perp$, or Certify accepts and in the latter case, by the first property of Certify stated in Claim 53, the condition as specified in Equation (31) holds, and the algorithm B will correct decide L on x with probability at least $9/10$. It follows that with probability at least $4/5$, the above algorithm A outputs either $\perp$ or the $L(x)$.

Secondly, by the second property of Certify stated in Claim 53, with probability at least $1 - 2^{-k}$ over the internal randomness of the our algorithm, we output a value that is not $\perp$ with probability at least $1 - 1/(2k)$ over $x \sim \mathcal{D}_n$. By a simple averaging argument, we get that

$$\mathop{\mathbf{Pr}}_{x \sim \mathcal{D}_n}\left[\mathop{\mathbf{Pr}}_{A}[A(x, 1^n, 1^k) = \perp] < 1/5\right] \geq 1 - 1/k.$$

This completes the proof of Theorem 2. □

5.3 Proof of Theorem 3

In this subsection, we show Theorem 3.

Proof of Theorem 3. Let $L \in \mathsf{NP}$. For an instance $x \in \{0,1\}^n$, let A_x denote the set of witnesses of x (with respect to some fixed verifier). Without loss of generality, we assume that every string in A_x has a length of exactly m, where $m = \mathsf{poly}(n)$. Let $\mathcal{D} : \{\mathcal{D}_n\}_{n \in \mathbb{N}}$ be a polynomial-time samplable distribution family.

Assuming $\mathsf{MINnKT}, \mathcal{U} \in \mathsf{AvgBPP}$, we show an randomized algorithm for solving L in time $2^{O(n/\log n)}$. The idea is to employ Lemma 52 and show that for every instance x, $\mathsf{pK}^{p(t)}(y \mid x) \leq \mathsf{pnK}^t(y \mid x) + O(n/\log n)$ *for some* $t \leq 2^{O(n/\log n)}$. Then we can use language compression pnK^t (Theorem 4) and follow the argument as shown in the proof of Theorem 10 to obtain an witness within the claimed time.

Claim 54 ([Hir21])**.** *For every $\varepsilon > 0$, every non-decreasing polynomials β and p, and every large enough $x \in \{0,1\}^n$, there exists a time bound t such that $\beta(n) \leq t \leq 2^{n^\varepsilon}$ and*

$$\mathsf{pK}^t(x) - \mathsf{pK}^{p(t)}(x) \leq O\left(\frac{n}{\log n}\right).$$

Proof Sketch of Claim 54. The proof can be easily adapted from that of [HKLO24, Lemma 12], which follows from a simple argument using telescoping sum. ◇

Consider the following algorithm $\mathcal{A}$ for solving L.

> On input $x \in \{0,1\}^n$, run $\mathsf{USamp}(1^m, 1^{2^{c\cdot n/\log n}}, x)$, where $c > 0$ is a constant specified later. Let y be the output of USamp, if y is an L-witness of x, accept. Otherwise, reject.

It is easy to see that the above algorithm runs in time $2^{O(n/\log n)}$. Next, we argue its correctness.

Let p_{LC} be the polynomial in Theorem 4 and p, q be the polynomials in Lemma 52. By Claim 54, there exists some $\max\{p_{\mathsf{LC}}(n), q(n,m)\} \le t \le 2^{\sqrt{n}}$ such that

$$\mathsf{pK}^t(x) - \mathsf{pK}^{p(t)}(x) \le O\left(\frac{n}{\log n}\right). \tag{32}$$

In this case, we have

$$\begin{aligned}
\mathsf{pK}^{p(t)}(y \mid x) &\le \mathsf{pnK}^t(y \mid x) + \mathsf{pK}^t(x) - \mathsf{pK}^{p(t)}(x) + \log p(t) && \text{(by Lemma 52)}\\
&\le \log |A_x| + \log t + \mathsf{pK}^t(x) - \mathsf{pK}^{p(t)}(x) + \log p(t) && \text{(by Theorem 4)}\\
&\le \log |A_x| + \log t + O\left(\frac{n}{\log n}\right) + \log p(t) && \text{(by Equation (32))}\\
&\le \log |A_x| + \frac{d' \cdot n}{\log n},
\end{aligned}$$

where $d' > 0$ is a sufficiently large constant. Then by Lemma 32 and letting c be a sufficiently large constant, we get that $\mathsf{USamp}(1^m, 1^{2^{c\cdot n/\log n}}, x)$ outputs a L-witness of x (if exists) with probability at least $1/2^{O(n/\log n)}$. It follows the algorithm $\mathcal{A}$ will reject with probability 1 on a no-instance and accept with probability at least $1/2^{O(n/\log n)}$ on a yes-instance. Using standard techniques, we can amply the success probability of such an algorithm to be at least $2/3$. □

6 Worst-Case Hardness of Gap-Cond-MINnKT: Proofs of Theorem 10 and Corollary 11

In this section, we prove Theorem 10 and Corollary 11. We first need the following lemma.

Lemma 55. *If* $\mathsf{Gap\text{-}Cond\text{-}MINnKT} \in \mathsf{prBPP}$*, then there exists a polynomial* $p(\cdot)$ *such that for all* $x, y \in \{0,1\}^*$ *and* $t \in \mathbb{N}$*,*

$$\mathsf{pK}^{p(t+|x|)}(y \mid x) \le \mathsf{pnK}^t(y \mid x) + \log p(t + |x|).$$

For the convenience of presenting the proofs in some later results, we will prove the following stronger lemma.

Lemma 56. *Assume there exists a constant* $c > 0$*, a polynomial* ρ*, and a probabilistic polynomial-time algorithm* A *such that the following hold for all sufficiently large* $n, m' \in \mathbb{N}$*, all* $x \in \{0,1\}^n$*, all* $t' \ge \rho(n + m')$*, and all* $s' \le m' - c \cdot \log(t' + n)$*:*

1. *For every* $y' \in \{0,1\}^{m'}$ *with* $\mathsf{nK}^t(y' \mid x) \le s'$*, we have* $\mathbf{Pr}_A[A(x, y', 1^{s'}, 1^{t'}) = 1] \ge \frac{2}{3}$.
2. *With probability at least* $1 - 1/t'$ *over* $y' \sim \{0,1\}^{m'}$*, we have* $\mathbf{Pr}_A[A(x, y', 1^{s'}, 1^{t'}) = 0] \ge \frac{2}{3}$.

Then there exists a polynomial $p(\cdot)$ such that for all $x, y \in \{0,1\}^*$ and $t \in \mathbb{N}$,

$$\mathsf{pK}^{p(t+|x|)}(y \mid x) \leq \mathsf{pnK}^t(y \mid x) + \log p(t + |x|).$$

Proof. Fix $x \in \{0,1\}^n$, $y \in \{0,1\}^m$ and $t \in \mathbb{N}$. Without loss generality, we assume that $t \geq |y|$. Let DP_k be the direct generator, where $k > 0$ is specified later.

Let $c > 0$ be a constant, ρ be a polynomial and A be an algorithm as stated in the assumption of the lemma.

Let $s := \mathsf{pnK}^t(y \mid x)$ and $k := s + d \cdot \log t$ for a sufficiently large constant $d > c$. Then with probability at least $2/3$ over $r \sim \{0,1\}^t$, there is a program $\Pi \in \{0,1\}^s$ such that

- $\forall w \in \{0,1\}^t$, $\Pi(x; w; r)$ outputs y or $\perp$ in t steps, and
- $\exists w \in \{0,1\}^t$, $\Pi(x; w; r)$ outputs y in t steps.

Note that the above implies that with probability at least $2/3$ over $r \sim \{0,1\}^t$,

$$\mathsf{nK}^{\mathsf{poly}(t)}(y \circ r \mid x) \leq s + |r| + O(\log t).$$

Also, for every $z \in \{0,1\}^{mk}$, $\mathsf{DP}_k(y; z)$ is computable in time $\mathsf{poly}(m)$ given x. It follows that with probability at least $2/3$ over $r \sim \{0,1\}^t$,

$$\mathsf{nK}^{t'}(\mathsf{DP}_k(y; z) \circ r \mid x) \leq s + mk + t + O(\log t) := s',$$

where $t' := \max\{\rho(n + m'), \mathsf{poly}(t)\}$ and $m' := mk + k$. By the property of the algorithm A (Item 1 in the lemma), the above implies that for any choice of $z \in \{0,1\}^{mk}$,

$$\Pr_{\substack{r \sim \{0,1\}^t \\ A}}\left[A\left(\mathsf{DP}_k(y; z) \circ r, x, 1^{s'}, 1^{t'}\right) = 1\right] \geq \frac{2}{3} \cdot \frac{2}{3} = \frac{4}{9}. \tag{33}$$

On the other hand, by the property of the algorithm A (Item 2 in the lemma), we get that

$$\Pr_{\substack{u \sim \{0,1\}^{nk+k} \\ r \sim \{0,1\}^t \\ A}}\left[A\left(u \circ r, x, 1^{s'}, 1^{t'}\right) = 0\right] > \frac{2}{3} \cdot (1 - 1/t') > \frac{4}{9}. \tag{34}$$

Comparing Equation (33) and Equation (34), we get a randomized distinguisher for $\mathsf{DP}_k(y; \mathcal{U}_{nk})$ with advantage $1/12$, defined by sampling $r \sim \{0,1\}^t$, and outputting $A(- \circ r, x, 1^{s'}, 1^{t'})$. By the reconstruction property of DP_k (Lemma 23), there exists some polynomial q such that

$$\mathsf{pK}^{q(t'+|x|)}(y \mid x) \leq k + \log q(t' + |x|).$$

Recall that $k = \mathsf{pnK}^t(y \mid x) + O(\log t)$. It follows that

$$\mathsf{pK}^{p(t+|x|)}(y \mid x) \leq \mathsf{pnK}^t(y \mid x) + \log p(t + |x|),$$

where p is some polynomial. □

Proof of Lemma 55. Given Lemma 56, it suffices to show that if Gap-Cond-MINnKT $\in$ prBPP, then the assumption in Lemma 56 is true.

Assuming Cond-MINnKT $\in$ prBPP, let B be a probabilistic polynomial-time algorithm and τ be a polynomial such that, given $(x, y', 1^{s'}, 1^{t'})$, B accepts with probability at least $2/3$ if $\mathsf{nK}^t(y' \mid x) \leq s'$ and rejects with probability at least $2/3$ if $\mathsf{nK}^{\tau(t+|x|)}(y' \mid x) > s' + \log \tau(t' + |x|)$.

We claim that B satisfies the conditions of an algorithm in Lemma 56. The first condition is immediate. For the second condition, note that by a simple counting argument, we have that with probability at least $1 - 1/t'$ over $y' \sim \{0,1\}^{m'}$,

$$\begin{aligned} \mathsf{nK}^{\tau(t')}(y' \mid x) &\geq \mathsf{K}(y' \mid x) \\ &\geq m' - O(\log t') \\ &> s' + \log \tau(t' + |x|), \end{aligned}$$

where the last inequality holds as long as $s' \leq m' - c \cdot \log t'$ for some sufficiently large constant c. In this case, we have that B rejects with probability at least $2/3$. □

We are now ready to prove Theorem 10.

Proof of Theorem 10. It is easy to see that Cond-MINnKT can be solved in NP using two (non-adaptive) calls to an NP oracle. Therefore, if NP $\subseteq$ BPP, then PH $\subseteq$ BPP [Ko82], which implies Cond-MINnKT $\in$ BPP. This also implies Gap-Cond-MINnKT $\in$ prBPP.

Next, we show that if Gap-Cond-MINnKT $\in$ prBPP, then NP $\subseteq$ BPP. Let $L \in$ NP. For an instance $x \in \{0,1\}^n$, let A_x denote the set of L-witnesses of x (with respect to some fixed verifier). Without loss of generality, we assume that every string in A_x has a length of exactly m, where $m = \mathsf{poly}(n)$.

By language compression for pnK^t (Theorem 4), we get that there is a polynomial p_{LC} such that for every string $y \in A_x$,

$$\mathsf{pnK}^{p_{\mathsf{LC}}(n),A_x}(y) \leq \log |A_x| + \log p_{\mathsf{LC}}(n).$$

Note that given x, the membership of A_x is decidable in polynomial time. Hence, we can efficiently answer oracle calls to A_x given x. This implies that there exists some polynomial p_1 such that for every $y \in A_x$,

$$\mathsf{pnK}^{p_1(n)}(y \mid x) \leq \mathsf{pnK}^{p_{\mathsf{LC}}(n),A_x}(y).$$

Furthermore, by Lemma 55, we have

$$\mathsf{pK}^{p(n)}(y \mid x) \leq \mathsf{pnK}^{p_1(n)}(y \mid x)$$

for some polynomial p. It follows that

$$\mathsf{pK}^{p(n)}(y \mid x) \leq \log |A_x| + \log p(n). \tag{35}$$

By Lemma 32, $\mathsf{USamp}(1^m, 1^{p(n)}, x)$ outputs a L-witness of x with probability at least $1/\mathsf{poly}(n)$. By standard amplification, this yields an efficient randomized algorithm for solving L with high probability. □

Proof of Corollary 11. First, by Theorem 1 and Theorem 10, the two statements "MINnKT $\in$ BPP" and "Gap-Cond-MINnKT $\in$ prBPP" are equivalent to "NP $\subseteq$ BPP". To include "Cond-MINnKT $\in$ prBPP" in this equivalence, observe that Cond-MINnKT $\in \Sigma_2^{\mathsf{P}}$. Since NP $\subseteq$ BPP implies $\Sigma_2^{\mathsf{P}} \subseteq$ BPP, it follows from the easiness of Gap-Cond-MINnKT that Cond-MINnKT $\in$ BPP. Finally, if Cond-MINnKT $\in$ BPP, then it trivially follows that Gap-Cond-MINnKT $\in$ prBPP. □

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[ACM+21] Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and a conditional variant of MKTP. In *Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 7:1–7:19, 2021.

[AFPS12] Luis Filipe Coelho Antunes, Lance Fortnow, Alexandre Pinto, and Andre Souto. Low-depth witnesses are easy to find. *Comput. Complex.*, 21(3):479–497, 2012.

[BFL01] Harry Buhrman, Lance Fortnow, and Sophie Laplante. Resource-bounded Kolmogorov complexity revisited. *SIAM Journal on Computing*, 31(3):887–905, 2001.

[BLvM05] Harry Buhrman, Troy Lee, and Dieter van Melkebeek. Language compression and pseudorandom generators. *Comput. Complex.*, 14(3):228–255, 2005.

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1), 2006.

[CHV22] Lijie Chen, Shuichi Hirahara, and Neekon Vafa. Average-Case Hardness of NP and PH from Worst-Case Fine-Grained Assumptions. In *Innovations in Theoretical Computer Science Conference* (ITCS), volume 215, pages 45:1–45:16, 2022.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Conference on Computational Complexity* (CCC), pages 10:1–10:24, 2016.

[GK22] Halley Goldberg and Valentine Kabanets. A simpler proof of the worst-case to average-case reduction for polynomial hierarchy via symmetry of information. *Electron. Colloquium Comput. Complex.*, 7:1–14, 2022.

[GKLO22a] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Probabilistic Kolmogorov complexity with applications to average-case complexity. In *Computational Complexity Conference* (CCC), pages 16:1–16:60, 2022.

[GKLO22b] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Probabilistic Kolmogorov complexity with applications to average-case complexity. In *Computational Complexity Conference* (CCC), pages 16:1–16:60, 2022.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Symposium on Theory of Computing* (STOC), pages 59–68, 1986.

[HIL+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 *Symposium on Theory of Computing* (STOC), pages 1039–1050, 2023.

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Symposium on Foundations of Computer Science* (FOCS), pages 247–258, 2018.

[Hir20a] Shuichi Hirahara. Characterizing average-case complexity of PH by worst-case meta-complexity. In *Symposium on Foundations of Computer Science* (FOCS), pages 50–60, 2020.

[Hir20b] Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In *Conference on Computational Complexity* (CCC), 2020.

[Hir20c] Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Symposium on Theory of Computing* (STOC), pages 1038–1051, 2020.

[Hir21] Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *Symposium on Theory of Computing* (STOC), pages 292–302, 2021.

[Hir22a] Shuichi Hirahara. Meta-computational average-case complexity: A new paradigm toward excluding Heuristica. *Bull. EATCS*, 136, 2022.

[Hir22b] Shuichi Hirahara. NP-hardness of learning programs and partial MCSP. In *Symposium on Foundations of Computer Science* (FOCS), pages 968–979, 2022.

[Hir22c] Shuichi Hirahara. Symmetry of information from meta-complexity. In *Computational Complexity Conference* (CCC), pages 26:1–26:41, 2022.

[HIR23] Yizhi Huang, Rahul Ilango, and Hanlin Ren. NP-hardness of approximating meta-complexity: A cryptographic approach. In *Symposium on Theory of Computing* (STOC), pages 1067–1075, 2023.

[HKLO24] Shuichi Hirahara, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Exact search-to-decision reductions for time-bounded Kolmogorov complexity. In *Computational Complexity Conference* (CCC), pages 29:1–29:56, 2024.

[Ila23] Rahul Ilango. SAT reduces to the minimum circuit size problem with a random oracle. In *Symposium on Foundations of Computer Science* (FOCS), pages 733–742, 2023.

[KC00] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Symposium on Theory of Computing* (STOC), pages 73–79, 2000.

[Ko82] Ker-I Ko. Some observations on the probabilistic algorithms and NP-hard problems. *Inf. Process. Lett.*, 14(1):39–43, 1982.

[Ko91] Ker-I Ko. On the complexity of learning minimum time-bounded Turing machines. *SIAM J. Comput.*, 20(5):962–986, 1991.

[Lee06] Troy Lee. *Kolmogorov complexity and formula lower bounds*. PhD thesis, University of Amsterdam, 2006.

[LO22] Zhenjian Lu and Igor C. Oliveira. Theory and applications of probabilistic Kolmogorov complexity. *Bull. EATCS*, 137, 2022.

[LOZ22] Zhenjian Lu, Igor Carboni Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 92:1–92:14, 2022.

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *Symposium on Foundations of Computer Science* (FOCS), pages 1243–1254, 2020.

[LP22] Yanyi Liu and Rafael Pass. On one-way functions from NP-complete problems. In *Conference on Computational Complexity* (CCC), pages 36:1–36:24, 2022.

[LR05] Troy Lee and Andrei E. Romashchenko. Resource bounded symmetry of information revisited. *Theor. Comput. Sci.*, 345(2-3):386–405, 2005.

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Symposium on Theory of Computing* (STOC), pages 330–335, 1983.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *Symposium on Foundations of Computer Science* (FOCS), pages 80–91, 1982.

A An Explicit Computational Model

In this section, we describe one possible way of defining a computational model and a modified encoding of programs that satisfy the properties stated in Section 2.2.

We consider multi-tape Turing machines over the alphabet $\Sigma = \{0, 1, \perp\}$. In addition to the usual input tape and work tapes, we allow a machine to access three additional tapes: one holding an auxiliary string, one holding a non-deterministic guess, and one holding the randomness. We assume that the machine has *sequential access* (i.e., no random access) to the randomness tape and the non-deterministic tape.[18] This assumption will be useful to guarantee "Randomness and Non-Determinism Extension". The remaining details of the model can be arbitrary.

We also assume an encoding function $\langle\cdot\rangle$ for machines (which we omit for simplicity of notation) such that there is a time-efficient universal machine U which, given (the encoding of) a machine M, an input α, an auxiliary string x, a non-deterministic guess w, and the random string r, runs in time $\mathsf{poly}(|M|, t)$ and outputs the string produced by $M(\alpha; x; w; r)$, where $|M|$ denote the encoding length of M and t is the running time of M on $(\alpha; x; w; r)$. This assumption will be useful to guarantee "Efficient Simulation".

In several previous papers, a program is defined as a pair (M, α), where M is a machine and $\alpha \in \{0,1\}^*$ is an "advice" string. However, this definition of programs alone, which does not provide further details about the encoding of programs, might not satisfy the conditions for being "strongly efficient", as stated in Section 2.2. Instead, we present a generic way of obtaining the desired properties given an existing encoding, which will be useful to guarantee "Efficient Composition". We consider *a sequence of machine-advice pairs*. More formally, a program Π is defined as

$$\Pi := ((M_1, \alpha_1), \ldots, (M_\ell, \alpha_\ell)),$$

where each M_i is a valid encoding of a machine and $\alpha_i \in \{0,1\}^*$. The output of Π on $(x; w; r)$ (which corresponds to the auxiliary input, the non-deterministic guess, and the randomness), denoted as $\Pi(x; w; r)$, is obtained as follows:[19]

- Let y_1 be the output of $M_1(\alpha_1; x; w; r)$. Also, let t_1 be the number of steps $M_1(\alpha_1; x; w; r)$ takes to produce y_1.
- Let y_2 be the output of $M_2\big(\alpha_2; x \circ y_1; w_{[t_1+1:|w|]}; r_{[t_1+1:|r|]}\big)$ and t_2 be the corresponding running time.

$$\vdots$$

- Let y_ℓ be the output of $M_\ell\big(\alpha_\ell; x \circ y_{\ell-1}; w_{[t_{\ell-1}+1:|w|]}; r_{[t_{\ell-1}+1:|r|]}\big)$ and t_ℓ be the corresponding running time.
- Output y_ℓ.

[18] In order to avoid confusion, sequential access means that the tape head can remain at a given tape cell or move left or right in each time step of the computation. In particular, we do not require read-only access or one-way access to the tape.

[19] For a string z of length ℓ and a subset $S \subseteq [\ell]$, where $[\ell] = \{1, 2, \ldots, \ell\}$, we let w_S denote the substring of z obtained by concatenating the bits of z corresponding to indexes in S. For $a < b$, we let $[a : b]$ denote the set $\{a, a+1, \ldots, b\}$.

Additionally, in each of the above steps, if $\perp$ is produced, Π outputs $\perp$ and halts.

The running time of the program Π on input (x, w, r) is given by $\sum_{i=1}^{\ell} t_i$.

We now describe how the program Π is encoded. To encode a machine-advice pair (M, α), we use the following format:

$$s_1\, 01\, \langle M \rangle\, s_2\, 01\, \alpha,$$

where $\langle M \rangle$ is an encoding of the machine M,[20] and s_1 is (resp. s_2) the binary representation of the integer specifying the length of $\langle M \rangle$ (resp. α), with each bit duplicated. The full description of Π is obtained by concatenating the encodings of all pairs (M_i, α_i) for $i \in [\ell]$. The size of Π, denoted $|\Pi|$, is the total bit length of this description.

Proposition 57. *The computational model described above is strongly efficient, as specified by Definition 12.*

Proof. The "Efficient Simulation" property follows from the fact that we use a time-efficient universal machine U that can simulate any machine with only a polynomial overhead.

The "Randomness Extension" property follows from the fact that a machine only has sequential access to the randomness tape. Indeed, if a program has a running time of at most t on a random string r with $|r| \geq t$, then since it only has sequential access to r, it will never reach the end of r during its execution, and appending any string to r will result in exactly the same output behavior. Similarly, the model also admits "Non-Determinism Extension".

For the "Efficient Universality" property, suppose we have a machine M and $\alpha_0, x, w, r \in \{0,1\}^*$ such that $M(\alpha_0; x; w; r)$ outputs y in t steps. Consider the program $\Pi := (M', \alpha)$, where M' and α are defined as follows:

> $\alpha := (M, \alpha_0)$, and M' is a machine that performs the following operation: on input $(\alpha = (M, \alpha_0); x; w; r)$, it simulates $M(\alpha_0; x; w; r)$.

It is straightforward to verify that the description length of Π is at most $|\alpha_0| + c_M \cdot \log |\alpha_0|$ for some constant $c_M > 0$ that depends only on M. Moreover, it follows from the construction that $\Pi(x; w; r)$ outputs $M(\alpha_0; x; w; r) = y$ in time t^{c_M}, provided that c_M is chosen to be sufficiently large.

Finally, we argue that the "Efficient Composition" property also holds. Let $x, y, z, r \in \{0,1\}^*$. Suppose there is a program $\Pi_1 = ((M_1, \alpha_1), \ldots, (M_\ell, \alpha_\ell))$ of size s_1 that computes $(x; r) \to y$ non-deterministically in time t_1, and a program Π_2 of size s_2 that computes $y \to z$ non-deterministically in time t_2. We first define the following machine-advice pair (M, α):

> $\alpha := (\Pi_2, m)$, where $m \in \mathbb{N}$ is the length of y, and M is a machine that does the following: given $(\alpha; x \circ y; w'; r')$, simulate $\Pi_2(y; w'; \epsilon)$.

Our final program Π for computing $(x; r) \to z$ is defined as Π_1 extended by (M, α), i.e.,

$$\Pi := ((M_1, \alpha_1), \ldots, (M_\ell, \alpha_\ell), (M, \alpha))\,.$$

First of all, it is easy to verify that the description of Π has length at most $s_1 + s_2 + \log p(|y| + |z|)$ for some sufficiently large polynomial p. To see that Π computes $(x; r) \to z$, note that after the execution of Π_1 on $(x; r)$, we have non-deterministically computed y, and the running time up to this point is at most t_1. After that, $M(\alpha, x \circ y; w'; r')$ will be run, where w' is the remaining part of the string on the non-deterministic tape and r' is the remaining part on the randomness tape. By construction, the (sub-)program (M, α) on $x \circ y$ will (non-deterministically) output z, as desired. Also, note that due to efficient simulation, $M(\alpha, x \circ y; w'; r')$ runs in time at most $p(t_2) + p(|x| + |y| + |z|)$. Therefore, the running time of the program Π on $(x; r)$ is at most $t_1 + p(t_2) + p(|x| + |y| + |z|)$. □

[20] As noted above, we assume a fixed standard encoding for Turing machines.

B Worst-Case to Average-Case Reductions and Gap-MINpnKT

For $\tau\colon \mathbb{N} \to \mathbb{N}$, let Gap_τ–MINpnKT be the following problem: Given $(x, 1^s, 1^t)$, where $x \in \{0,1\}^*$ and $s, t \in \mathbb{N}$, decide whether $\mathsf{pnK}^t(x) \le s$ or $\mathsf{pnK}^{\tau(t)}(x) > s + \log \tau(t)$. We say that Gap-MINpnKT $\in$ prBPP if there is a probabilistic polynomial-time algorithm that solves Gap_τ-MINpnKT for some polynomial τ.

Proposition 58. *The following holds.*

$$(\mathsf{MINnKT}, \mathcal{U}) \in \mathsf{AvgBPP} \implies \mathsf{Gap\text{-}MINpnKT} \in \mathsf{prBPP}.$$

Proof. First of all, suppose $(\mathsf{MINnKT}, \mathcal{U}) \in \mathsf{AvgBPP}$. Then by Lemma 48, we have that $(\mathsf{coMINnKT}, \mathcal{U}) \in \mathsf{Avg}^1\mathsf{BPP}$. In other words, there exist a constant $c > 0$, a polynomial ρ, and a probabilistic polynomial-time algorithm A such that the following hold for all sufficiently large n', all $t' \ge \rho(n')$, and all $s' \le n' - c \cdot \log t'$:

1. For every $x \in \{0,1\}^{n'}$ with $\mathsf{nK}^t(x) \le s'$, we have $\mathbf{Pr}_A[A(x, 1^{s'}, 1^{t'}) = 1] \ge 1 - \frac{1}{10t'}$.
2. With probability at least $1 - 1/t'$ over $x \sim \{0,1\}^{n'}$, we have $\mathbf{Pr}_A[A(x, 1^{s'}, 1^{t'}) = 0] \ge 1 - \frac{1}{10t'}$.

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

- $k := s + d^3 \cdot \log t$.
- $n' := k + t^d$.
- $t' := t^{2d}$.
- $s' := s + O(\log\log t') + t^d + nk + d^2 \log t$.

Define an algorithm A' as follows:

> On input $(x, 1^s, 1^t)$ with $x \in \{0,1\}^n$, sample $z \sim \{0,1\}^{nk}$ and $w \sim \{0,1\}^{t^d}$, and then output $A\Big(\mathsf{DP}_k(x; z) \circ w, 1^{s'}, 1^{t'}\Big)$.

Below, we show that A' solves Gap_τ-MINpnKT correctly with high probability, where τ is a polynomial specified later.

First, consider the case that $(x, 1^s, 1^t)$ is a yes-instance of Gap_τ-MINpnKT, i.e., $\mathsf{pnK}^t(x) \le s$. By success amplification (Lemma 29), we get that

$$\mathsf{pnK}^{t^d}_{1-1/10n'}(x) \le s + O(\log\log t').$$

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

$$\mathop{\mathbf{Pr}}_{w \sim \{0,1\}^{t^d}}\Big[\mathsf{nK}^{t^{2d}}(\mathsf{DP}_k(y; z) \circ w) \le s + O(\log\log t') + t^d + nk + O(d \log t)\Big] \ge 1 - \frac{1}{10n'}. \tag{36}$$

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

$$s + O(\log\log t') + t^d + nk + O(d \log t) \le s'. \tag{37}$$

Equation (36) and Equation (37) together imply that

$$\mathop{\mathbf{Pr}}_{w \sim \{0,1\}^{t^d}}\Big[\mathsf{nK}^{t^{2d}}(\mathsf{DP}_k(y; z) \circ w \mid x) \le s'\Big] \ge 1 - \frac{1}{10t'}.$$

Then by the property of A and a union bound, we have

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

and so

$$\mathop{\mathbf{Pr}}_{A'}\big[A'(x,1^s,1^t)=1\big] \geq 1-\frac{1}{5t'}. \tag{38}$$

Now consider any $(x,1^s,1^t)$ that is a no-instance of Gap_τ-MINpnKT, i.e., $\mathsf{pK}^{\tau(t)}(X) > s+\log\tau(t)$. We will show that

$$\mathop{\mathbf{Pr}}_{A'}\big[A'(x,1^s,1^t)=1\big] \leq 1-\frac{2}{5t'}. \tag{39}$$

Note that by combining Equation (38) and Equation (39), A' yields a polynomial-time algorithm for solving Gap_τ-MINpnKT via standard success amplification techniques.

Suppose, for the sake of contradiction, Equation (39) is not true. Then by the definition of A', we have

$$\mathop{\mathbf{Pr}}_{w,z,A}\Big[A(\mathsf{DP}_k(x;z)\circ w, x, 1^{s'}, 1^{t'})=1\Big] > 1-\frac{2}{5t'}. \tag{40}$$

On the other hand, by the property of A, we get

$$\mathop{\mathbf{Pr}}_{u,w,A}\Big[A(u\circ w, 1^{s'}, 1^{t'})=0\Big] \geq 1-\frac{1}{2t'}. \tag{41}$$

Comparing Equation (40) and Equation (41), we get a randomized procedure defined as $B(-\circ\mathcal{U}_{t^d}, 1^{s'}, 1^{t'})$ that distinguishes $\mathsf{DP}_k(x;\mathcal{U}_{O(nk)})$ from $\mathcal{U}_m$ with advantage $1/(10t')$. By Lemma 23 and by letting τ be a sufficiently large polynomial, we get

$$\begin{aligned}\mathsf{pK}^{\tau(t)}(x) &\leq k + O(\log t')\\ &\leq s + d^3\cdot\log t + O(d\log t)\\ &\leq s+\log\tau(t).\end{aligned}$$

This means $(x,1^s,1^t)$ is *not* a no-instance of Gap_τ-MINpnKT, which gives the desired contradiction. □

Corollary 59. *Suppose*

$$\mathsf{Gap\text{-}MINpnKT}\in\mathsf{prBPP} \implies \mathsf{NP}\subseteq\mathsf{BPP}.$$

Then

$$\mathsf{Dist}\Sigma_2^{\mathsf{P}}\subseteq\mathsf{AvgBPP} \implies \mathsf{NP}\subseteq\mathsf{BPP}.$$

Proof. If $\mathsf{Dist}\Sigma_2^{\mathsf{P}}\subseteq\mathsf{AvgBPP}$, then $(\mathsf{MINnKT},\mathcal{U})\in\mathsf{AvgBPP}$, since $\mathsf{MINnKT}\in\Sigma_2^{\mathsf{p}}$. By Proposition 58, we get $\mathsf{Gap\text{-}MINpnKT}\in\mathsf{prBPP}$. Finally, by the assumption, we get $\mathsf{NP}\subseteq\mathsf{BPP}$. □

Consequently, if Gap-MINpnKT also captures the hardness of NP (extending Theorem 9), a breakthrough result relating the average-case and worst-case complexities of the polynomial hierarchy would follow.

C Worst-Case Hardness of MINpnKT: Proof of Theorem 9

Definition 60 (MINpnKT)**.** We define MINpnKT as a promise problem $(\mathcal{YES}, \mathcal{NO})$, where

$$\mathsf{pnK}^t_{b/a}(x) \leq s \implies (x, 1^s, 1^t, 1^a, 1^b) \in \mathcal{YES},$$
$$\mathsf{pnK}^t_{(b-1)/a}(x) > s \implies (x, 1^s, 1^t, 1^a, 1^b) \in \mathcal{NO}.$$

In this section we prove Theorem 9.

Proof of Theorem 9. The theorem follows directly from Theorem 61 and Theorem 62, stated and proved in Appendix C.1 and Appendix C.2 respectively. □

C.1 The Easy Direction

Here we prove the following.

Theorem 61. *If* NP $\subseteq$ BPP*, then* MINpnKT $\in$ BPP.

Proof. By [Ko82], we know that NP $\subseteq$ BPP implies PH $\subseteq$ BPP. We define a partial function f such that for $z = r_1 \circ \cdots \circ r_k$ where each $f_i \in \{0,1\}^t$, $f(x, 1^s, 1^t, 1^k, z)$ is defined as the number of $i \in [k]$ such that $\mathsf{nK}^t(x \mid \epsilon; r_i) \leq s$. We also define a language L, such that $(x, 1^s, 1^t, 1^k, 1^v, z) \in L$ iff $f(x, 1^s, 1^t, 1^k, z) \geq v$. It is not hard to see that $L \in$ PH, because we can use alternation of quantifiers to simulate every non-deterministic program of bounded length, and count the number of "good" r_i in z. By our assumption, $L \in$ BPP, and by a simple search-to-decision reduction, $f \in$ FBPP. Let A be the randomized polynomial-time algorithm computing f with $1/10$ error.

Now suppose we are given $(x, 1^s, 1^t, 1^a, 1^b)$ which is in the promised inputs of MINpnKT. Our randomized algorithm for deciding $(x, 1^s, 1^t, 1^a, 1^b)$ works as follows: we set $k = 18a^2$, and we take k independent samples $r_1, \ldots, r_k \sim \mathcal{U}_t$. Then we compute $\tilde{v} = A(x, 1^s, 1^t, 1^k, r_1 \circ \cdots \circ r_k)$. We accept iff $\tilde{v} \geq \frac{b-1/2}{a}k$. Clearly this algorithm runs in polynomial time, so we only need to bound its error. In fact, by the definition of MINpnKT and pnK, if the input is in $\mathcal{YES}$, then we have

$$\Pr_{r \sim \mathcal{U}_t}\left[\mathsf{nK}^t(x \mid \epsilon; r) \leq s\right] \geq \frac{b}{a}.$$

On the other hand, if the input is in $\mathcal{NO}$, then we have

$$\Pr_{r \sim \mathcal{U}_t}\left[\mathsf{nK}^t(x \mid \epsilon; r) \leq s\right] < \frac{b-1}{a}.$$

Let $v = f(x, 1^s, 1^t, 1^k, r_1 \circ \cdots \circ r_k)$. By Chernoff bound, when input is in $\mathcal{YES}$, with probability at least $(1 - \mathrm{e}^{-4})$, $v \geq \frac{b-1/3}{a}k$; and when the input is in $\mathcal{NO}$, with probability at least $(1 - \mathrm{e}^{-4})$, $v \leq \frac{b-2/3}{a}k$. Since the error probability of A is bounded by $1/10$, we have $\mathbf{Pr}[v \neq \tilde{v}] \leq 1/10$. Hence by union bound, the probability of our algorithm making an error on promised inputs is at most $\mathrm{e}^{-4} + 1/10 \leq 1/3$. □

C.2 The Hard Direction

Throughout this subsection, we will assume that the underlying computational model is *strongly efficient*, as defined in Section 2.2. We establish the hard direction of Theorem 9.

Theorem 62. *If* MINpnKT $\in$ prBPP*, then* NP $\subseteq$ BPP.

Comparison with the proof of Theorem 34 in Section 4.2. The proof of Theorem 62 has the same structure as that of Theorem 34, and requires essentially no new ideas. In this paragraph, we sketch the modifications needed for the proof of Theorem 62. (Given the relevance of this result for the discussion in Section 1.2.3, for completeness and convenience of the reader, below we also include a detailed proof of Theorem 62.) First, analogously to Definition 35, we define a technical variant of Kolmogorov complexity called ℓ-pnK (see Definition 63). Then, similarly to Lemma 36, we prove a lower bound for ℓ-pnK by a counting argument in Lemma 64, but now we also take into account the randomness of pnK. From Definition 40 and Lemma 41, we get Definition 68 and Lemma 69 by changing nK into pnK. In order to adapt Lemmas 37 to 39 and 42 to 44, in addition to changing nK into pnK, we need to change the arguments "$\epsilon \to x \circ r$ in non-deterministic time t and size s" into "$(\epsilon; r) \to x \circ \hat{r}$ in non-deterministic time t and size s", and we also need to use randomness extension (Definition 12) to extend the length of the random string w. These modifications give Lemmas 65 to 67 and 70 to 72. Finally, with respect to Lemmas 46 and 47, we just change nK into pnK to get Lemmas 47 and 74. The remaining steps of the proof are unchanged.

C.2.1 Technical Tool: $\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}$

Definition 63 ($\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}$)**.** For any $t, \ell \in \mathbb{N}$, any $\lambda, \gamma \in (0,1)$, and any string $x \in \{0,1\}^*$, we define $\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x)$ as

$$\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x) = \min\left\{ s \in \mathbb{N} \;\middle|\; \Pr_{\hat{r}\sim\mathcal{U}_\ell}\left[\mathsf{pnK}^t_\lambda(x \circ \hat{r}) \le s + \ell\right] \ge \gamma \right\}.$$

As before, from the definition, it is not immediately clear that $\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x)$ is non-negative. However, we can bound its value from below by $(-1 - \lceil \log 1/(\lambda\gamma) \rceil)$, as stated in the following lemma:

Lemma 64 (Lower bound for $\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}$)**.** *For any $\ell, t \in \mathbb{N}$, any string $x \in \{0,1\}^*$ and any $\lambda, \gamma \in (0,1)$, we have*

$$\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x) \ge -\left\lceil \log \frac{1}{\lambda\gamma} \right\rceil - 1.$$

Proof. For the sake of contradiction, suppose that $\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x) \le -\lceil \log 1/(\lambda\gamma) \rceil - 2$. By the definitions of ℓ-pnK and pnK, we have

$$\begin{aligned}
&\Pr_{\hat{r}\sim\mathcal{U}_\ell}\left[\mathsf{pnK}^t_\lambda(x \circ \hat{r}) \le \ell - \left\lceil \log \frac{1}{\lambda\gamma} \right\rceil - 2\right] \ge \gamma \\
\implies &\Pr_{\hat{r}\sim\mathcal{U}_\ell}\left[\Pr_{r\sim\mathcal{U}_t}\left[\mathsf{nK}^t(x \circ \hat{r} \mid \epsilon; r) \le \ell - \left\lceil \log \frac{1}{\lambda\gamma} \right\rceil - 2\right] \ge \lambda\right] \ge \gamma \\
\implies &\Pr_{\substack{\hat{r}\sim\mathcal{U}_\ell \\ r\sim\mathcal{U}_t}}\left[\mathsf{nK}^t(x \circ \hat{r} \mid \epsilon; r) \le \ell - \left\lceil \log \frac{1}{\lambda\gamma} \right\rceil - 2\right] \ge \lambda\gamma.
\end{aligned}$$

Now define the set A as

$$A = \left\{ (\hat{r}, r) \in \{0,1\}^\ell \times \{0,1\}^t \;\middle|\; \mathsf{nK}^t(x \circ \hat{r} \mid \epsilon; r) \le \ell - \left\lceil \log \frac{1}{\lambda\gamma} \right\rceil - 2 \right\}.$$

Then $|A| \ge \lambda\gamma \cdot 2^{\ell+t}$. However, the number of programs of length at most $(\ell - \lceil \log 1/(\lambda\gamma) \rceil - 2)$ is bounded by $2^0 + \cdots + 2^{\ell - \lceil \log 1/(\lambda\gamma) \rceil - 2} \le 2^{\ell - \lceil \log 1/(\lambda\gamma) \rceil - 1} \le \lambda\gamma \cdot 2^{\ell-1}$. Therefore, there are at most $\lambda\gamma \cdot 2^{\ell+t-1}$ pairs (Π, r) where $|\Pi| \le \ell - \lceil \log 1/(\lambda\gamma) \rceil - 2$ and $|r| = t$. Since $|A| \ge \lambda\gamma \cdot 2^{\ell+t} > \lambda\gamma \cdot 2^{\ell+t-1}$, by pigeon hole principle, there must exists a pair (Π, w) and two strings $\hat{r}_1 \ne \hat{r}_2$, such that Π non-deterministically computes both $(\epsilon; r) \to (x \circ \hat{r}_1)$ and $(\epsilon; r) \to (x \circ \hat{r}_2)$, a contradiction. □

Lemma 65 (Upper Bound for ℓ-$\mathsf{pnK}^t_{\lambda,\gamma}$)**.** *There exists a polynomial p such that the following holds. For any $\ell \in \mathbb{N}$, any string $x \in \{0,1\}^*$ and any $\lambda, \gamma \in (0,1)$, we have*

$$\ell\text{-}\mathsf{pnK}^{p(|x|+\ell)}_{\lambda,\gamma}(x) \leq |x| + \log p(|x|+\ell).$$

Proof. Let M be the Turing machine satisfying the following.

For any strings $x, w, r \in \{0,1\}^*$, $M(x;\epsilon;w;r) = x$ in $O(|x|)$ steps.

Hence by efficient universality (Definition 12), when p is a large enough polynomial, for any string $r \in \{0,1\}^{p(|x|+\ell)}$ and $\hat{r} \in \{0,1\}^\ell$, $(\epsilon; r) \to (x \circ \hat{r})$ is in non-deterministic time $p(|x|+\ell)$ and size $(|x| + \ell + \log p(|x|+\ell))$. Therefore, for any $\hat{r} \in \{0,1\}^\ell$, we have

$$\Pr_{r \sim \mathcal{U}_{p(|x|+\ell)}}\left[\mathsf{nK}^{p(|x|+\ell)}(x \circ \hat{r} \mid \epsilon; r) \leq |x| + \ell + \log p(|x|+\ell)\right] = 1.$$

Which gives us $\mathsf{pnK}^{p(|x|+\ell)}_{\lambda}(x \circ \hat{r}) \leq |x| + \ell + \log p(|x|+\ell)$ for any $\lambda \in (0,1)$. Since this holds for any $\hat{r}$, we get

$$\Pr_{\hat{r} \sim \mathcal{U}_\ell}\left[\mathsf{pnK}^{p(|x|+\ell)}_{\lambda}(x \circ \hat{r}) \leq |x| + \ell + \log p(|x|+\ell)\right] = 1.$$

By the definition of ℓ-pnK, we conclude that for any $\gamma \in (0,1)$,

$$\ell\text{-}\mathsf{pnK}^{p(|x|+\ell)}_{\lambda,\gamma}(x) \leq |x| + \log p(|x|+\ell),$$

which completes the proof. □

Lemma 66 (Monotonicity of ℓ-$\mathsf{pnK}^t_{\lambda,\gamma}(x)$ on λ, γ, t)**.** *For any $\ell, t_1, t_2 \in \mathbb{N}^+$ satisfying $t_1 \geq t_2$, any $\lambda_1, \lambda_2, \gamma_1, \gamma_2$ satisfying $\lambda_1 \leq \lambda_2$ and $\gamma_1 \leq \gamma_2$, and any string $x \in \{0,1\}^*$, we have*

$$\ell\text{-}\mathsf{pnK}^{t_1}_{\lambda_1,\gamma_1}(x) \leq \ell\text{-}\mathsf{pnK}^{t_2}_{\lambda_2,\gamma_2}(x).$$

Proof. Let $s = \ell$-$\mathsf{pnK}^{t_2}_{\lambda_2,\gamma_2}(x)$. Then we have

$$\Pr_{\hat{r} \sim \mathcal{U}_\ell}\left[\mathsf{pnK}^{t_2}_{\lambda_2}(x \circ \hat{r}) \leq s\right] \geq \gamma_2.$$

By randomness & non-determinism extension (Definition 12), for any $\hat{r} \in \{0,1\}^\ell$, we have $\mathsf{pnK}^{t_1}_{\lambda_1}(x \circ \hat{r}) \leq \mathsf{pnK}^{t_2}_{\lambda_2}(x \circ \hat{r})$. Hence we get

$$\Pr_{\hat{r} \sim \mathcal{U}_\ell}\left[\mathsf{pnK}^{t_1}_{\lambda_1}(x \circ \hat{r}) \leq s\right] \geq \gamma_2 \geq \gamma_1.$$

Hence by the definition of ℓ-pnK, we have ℓ-$\mathsf{pnK}^{t_1}_{\lambda_1,\gamma_1}(x) \leq s$. Substituting s by ℓ-$\mathsf{pnK}^{t_2}_{\lambda_2,\gamma_2}(x)$ finishes the proof. □

Lemma 67 (Monotonicity of ℓ-$\mathsf{pnK}^t_{\lambda,\gamma}(x)$ on ℓ, x)**.** *There exists a polynomial p such that the following holds. For any $\ell, \ell' \in \mathbb{N}$ with $\ell < \ell'$, any $\lambda, \gamma \in (0,1)$, any strings $x, y \in \{0,1\}^*$ and any $t \in \mathbb{N}$, we have*

$$\ell'\text{-}\mathsf{pnK}^{t+p(|x|+|y|+\ell')}_{\lambda,\gamma}(x) \leq \ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x \circ y) + \log p(|x| + |y| + \ell')$$

Proof. Let $s = \ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x \circ y)$. For some string $\hat{r} \in \{0,1\}^\ell$, we say $\hat{r}$ is good if $\mathsf{pnK}^t_\lambda(x \circ y \circ \hat{r}) \le s + \ell$. Then by the definition of $\ell\text{-}\mathsf{pnK}$, with probability at least γ over $\hat{r} \sim \mathcal{U}_\ell$, r is good. For such a good $\hat{r}$, by the definition of pnK, we have

$$\Pr_{r \sim \mathcal{U}_t}\left[\mathsf{nK}^t(x \circ y \circ \hat{r} \mid \epsilon; r) \le s + \ell\right] \ge \lambda.$$

By randomness extension (Definition 12), for any $t' \ge t$, we have

$$\Pr_{r' \sim \mathcal{U}_{t'}}\left[\mathsf{nK}^t(x \circ y \circ \hat{r} \mid \epsilon; r') \le s + \ell\right] \ge \lambda.$$

In other words, with probability λ over $r' \sim \mathcal{U}_{t'}$, $(\epsilon; r') \to (x \circ y \circ \hat{r})$ is in non-deterministic time t and size $(s + \ell)$. Now for any integer i, let $\widetilde{i}$ denote the binary encoding of i with each bit duplicated, concatenated with 01. Let M be the Turing machine satisfying the following.

For any strings $x, y, \hat{r}, \hat{r}', w \in \{0,1\}^*$,

$$M(\widetilde{|x|} \circ \widetilde{|y|} \circ \hat{r}'; x \circ y \circ \hat{r}; w; \epsilon) = x \circ \hat{r} \circ \hat{r}'$$

in $\mathsf{poly}(|x|, |y|, |\hat{r}|, |\hat{r}'|)$ steps.

Hence by efficient universality (Definition 12), for any $\hat{r}' \in \{0,1\}^{\ell' - \ell}$, $(x \circ y \circ \hat{r}) \to (x \circ \hat{r} \circ \hat{r}')$ is in non-deterministic time $\mathsf{poly}(|x| + |y| + \ell')$ and size $(\ell' - \ell + O(\log(|x| + |y| + \ell')))$. Applying efficient composition (Definition 12), when p is some large enough polynomial,

$$\Pr_{r' \sim \mathcal{U}_{t + p(|x|+|y|+\ell')}}\left[\mathsf{nK}^{t+p(|x|+\ell')}(x \circ \hat{r} \circ \hat{r}' \mid \epsilon; r') \le s + \ell' + \log p(|x| + |y| + \ell')\right] \ge \lambda.$$

By the definition of pnK, this is equivalent to

$$\mathsf{pnK}^{t+p(|x|+\ell')}_\lambda(x \circ \hat{r} \circ \hat{r}') \le s + \ell' + \log p(|x| + |y| + \ell').$$

Since the above inequality works for any good $\hat{r} \sim \mathcal{U}_\ell$ and any $\hat{r}' \in \{0,1\}^{\ell' - \ell}$, we get

$$\Pr_{\substack{\hat{r} \sim \mathcal{U}_\ell \\ \hat{r}' \sim \mathcal{U}_{\ell' - \ell}}}\left[\mathsf{pnK}^{t+p(|x|+\ell')}_\lambda(x \circ \hat{r} \circ \hat{r}') \le s + \ell' + \log p(|x| + |y| + \ell')\right] \ge \gamma.$$

In other words, $\ell'\text{-}\mathsf{pnK}^{t+p(|x|+\ell')}_{\lambda,\gamma}(x) \le s + \log p(|x| + |y| + \ell')$. Substituting s by $\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x \circ y)$ finishes the proof. □

Similar to $\mathsf{MIN}\ell\text{-}\mathsf{nKT}$, we can define the $\mathsf{MIN}\ell\text{-}\mathsf{pnKT}$ problem for $\ell\text{-}\mathsf{pnK}$.

Definition 68 ($\mathsf{MIN}\ell\text{-}\mathsf{pnKT}$)**.** We define $\mathsf{MIN}\ell\text{-}\mathsf{pnKT}$ as the promise problem $(\mathcal{YES}, \mathcal{NO})$ such that

$$\begin{cases} \ell\text{-}\mathsf{pnK}^t_{b/a,d/c}(x) \le s \implies (x, 1^s, 1^t, 1^\ell, 1^a, 1^b, 1^c, 1^d) \in \mathcal{YES}, \\ \ell\text{-}\mathsf{pnK}^t_{(b-1)/a,(d-1)/c}(x) > s \implies (x, 1^s, 1^t, 1^\ell, 1^a, 1^b, 1^c, 1^d) \in \mathcal{NO}. \end{cases}$$

A basic property of $\mathsf{MIN}\ell\text{-}\mathsf{pnKT}$ is that its easiness follows from the easiness of $\mathsf{MINpnKT}$.

Lemma 69. *If* $\mathsf{MINpnKT} \in \mathsf{prBPP}$*, then* $\mathsf{MIN}\ell\text{-}\mathsf{pnKT} \in \mathsf{prBPP}$.

Proof. Suppose we are given an instance $(x, 1^s, 1^t, 1^\ell, 1^a, 1^b, 1^c, 1^d)$ that is in the promised inputs of $\mathsf{MIN}\ell\text{-}\mathsf{pnKT}$, and we want to decide whether it is in $\mathcal{YES}$ or $\mathcal{NO}$. If it is in $\mathcal{YES}$, then we have $\ell\text{-}\mathsf{pnK}^t_{b/a,d/c}(x) \leq s$, and by the definition of $\ell\text{-}\mathsf{pnK}$, we get

$$\mathop{\mathbf{Pr}}_{\hat{r}\sim\mathcal{U}_\ell}\left[\mathsf{pnK}^t_{b/a}(x \circ \hat{r}) \leq s + \ell\right] \geq \frac{d}{c}.$$

On the other hand, if the instance is in $\mathcal{NO}$, then we have $\ell\text{-}\mathsf{pnK}^t_{(b-1)/a,(d-1)/c}(x) > s$, and by the definition of $\ell\text{-}\mathsf{pnK}$, we get

$$\mathop{\mathbf{Pr}}_{\hat{r}\sim\mathcal{U}_\ell}\left[\mathsf{pnK}^t_{(b-1)/a}(x \circ \hat{r}) \leq s + \ell\right] < \frac{d-1}{c}.$$

Now let A be the probabilistic polynomial-time algorithm that decides $\mathsf{MINpnKT}$ with error $1/3$. Let A^m denote the algorithm that runs A for m times and takes the majority. By Chernoff bound, the error of A^m is bounded by $2^{-m/18}$. Our algorithm B for deciding $\mathsf{MIN}\ell\text{-}\mathsf{pnKT}$ works as follows: it first defines $k = 18c^2$, then it takes k independent samples $\hat{R}_1, \ldots, \hat{R}_k \sim \mathcal{U}_\ell$. Then for each $i \in [k]$, it computes $X_i = A^{\lceil 36 \log k \rceil}(x \circ \hat{R}_i, 1^s, 1^t, 1^a, 1^b)$. Finally, it computes $V = (\sum_{i=1}^k X_i)/k$, and accepts iff $V \geq (d - 1/2)/c$.

It is not hard to see that B runs in time $\mathsf{poly}(|x|, t, s, \ell, a, c)$. We claim that B computes $\mathsf{MIN}\ell\text{-}\mathsf{pnKT}$ with error at most $1/3$. Let $Y_i = \mathsf{MINpnKT}(x \circ \hat{R}_i, 1^s, 1^t, 1^a, 1^b)$. First, since the error of A^m is bounded by $2^{-m/18}$, by a union bound, we have

$$\mathbf{Pr}[\exists i \in [k], X_i \neq Y_i] \leq k \cdot 2^{-36 \log k/18} = \frac{1}{k}. \tag{42}$$

That is, with probability $(1-1/k)$, all $\mathsf{MINpnKT}(x \circ \hat{R}_i, 1^s, 1^t, 1^a, 1^b)$ are computed correctly by $A^{\lceil 36 \log k \rceil}$. Next, if the input is in $\mathcal{YES}$, then by the definition of $\mathsf{MIN}\ell\text{-}\mathsf{pnKT}$, we have $\mathbb{E}[\sum_{i=1}^k Y_i] \geq k \cdot d/c$. Also, Y_i are independent identically distributed random variables in $\{0, 1\}$. Therefore, using Chernoff bound, we get

$$\mathbf{Pr}\left[\sum_{i=1}^k Y_i < k \cdot \frac{d - 1/3}{c}\right] \leq \exp\left(-2 \cdot \frac{1}{9c^2} \cdot k\right) = e^{-4}. \tag{43}$$

Combining Equations (42) and (43) with union bound, we have

$$\mathbf{Pr}\left[\sum_{i=1}^k X_i < k \cdot \frac{d - 1/3}{c}\right] \leq e^{-4} + \frac{1}{k} \leq \frac{1}{3}$$

That is, the probability of B rejecting an input from $\mathcal{YES}$ is at most $1/3$. Similarly, one can show that the probability of B accepting an input from $\mathcal{NO}$ is at most $1/3$. Hence we conclude that $\mathsf{MIN}\ell\text{-}\mathsf{pnKT} = (\mathcal{YES}, \mathcal{NO}) \in \mathsf{prBPP}$. □

C.2.2 Useful Bounds for $\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}$

In this section, we state and prove the following lemmas for $\ell\text{-}\mathsf{pnK}$. They are important ingredients in the proof of the hard direction of Theorem 9.

Lemma 70. *There exists a polynomial p such that the following holds. For any $n, m, k, t, \ell \in \mathbb{N}^+$, any $\lambda, \gamma \in (0, 1)$ and any string $x \in \{0,1\}^n, y \in \{0,1\}^m, z \in \{0,1\}^{mk}$, we have*

$$\ell\text{-}\mathsf{pnK}^{t+p(nmk\ell)}_{\lambda,\gamma}(x \circ \mathsf{DP}_k(y; z)) \leq \ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x \circ y) + mk + \log p(nmk\ell).$$

Proof. Let $s = \ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x \circ y)$. We say that $\hat{r} \in \{0,1\}^\ell$ is good, if $\mathsf{pnK}^t_\lambda(x \circ y \circ \hat{r}) \leq s + \ell$. Then by the definition of ℓ-pnK, with probability at least γ over $\hat{r} \sim \mathcal{U}_\ell$, $\hat{r}$ is good. For a good $\hat{r}$, by the definition of pnK, we have

$$\Pr_{r \sim \mathcal{U}_t}\left[\mathsf{nK}^t(x \circ y \circ \hat{r} \mid \epsilon; r) \leq s + \ell\right] \geq \lambda.$$

By randomness extension (Definition 12), for any $t' \geq t$ we have

$$\Pr_{r' \sim \mathcal{U}_{t'}}\left[\mathsf{nK}^t(x \circ y \circ \hat{r} \mid \epsilon; r') \leq s + \ell\right] \geq \lambda.$$

For any integer i, let $\widetilde{i}$ denote the binary encoding of i with each bit duplicated, concatenated with 01. Let M be the Turing machine satisfying the following.

> For any $n, m, k, \ell \in \mathbb{N}^+$, for any strings $x \in \{0,1\}^n, y \in \{0,1\}^m, z \in \{0,1\}^{mk}, \hat{r} \in \{0,1\}^\ell, w \in \{0,1\}^*$,
>
> $$M(\widetilde{n} \circ \widetilde{m} \circ \widetilde{k} \circ \widetilde{\ell} \circ z; x \circ y \circ \hat{r}; w; \epsilon) = (x \circ \mathsf{DP}_k(y; z) \circ \hat{r})$$
>
> in $\mathsf{poly}(nmk\ell)$ steps.

By efficient universality (Definition 12), $(x \circ y \circ \hat{r}) \to (x \circ \mathsf{DP}_k(y; z) \circ \hat{r})$ is in non-deterministic time $\mathsf{poly}(nmk\ell)$ and size $(mk + O(\log(nmk\ell)))$. Then by efficient composition (Definition 12), for a large enough polynomial p, for any good $\hat{r}$, we get

$$\Pr_{r' \sim \mathcal{U}_{t+p(nmk\ell)}}\left[\mathsf{nK}^{t+p(nmk\ell)}(x \circ \mathsf{DP}_k(y; z) \circ \hat{r} \mid \epsilon; r') \leq s + \ell + mk + \log p(nmk\ell)\right] \geq \lambda.$$

In other words, $\mathsf{pnK}^{t+p(nmk\ell)}_\lambda(x \circ \mathsf{DP}_k(y; z) \circ \hat{r}) \leq s + \ell + mk + \log p(nmk\ell)$. Because the fraction of good $\hat{r} \sim \mathcal{U}_\ell$ is at least γ, we conclude that

$$\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x \circ \mathsf{DP}_k(y; z)) \leq s + mk + \log p(nmk\ell),$$

Substituting s by $\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x \circ y)$ finishes the proof. □

Lemma 71. *For any $n, m, k, t, \ell \in \mathbb{N}^+$, any $\lambda, \gamma, \alpha \in (0,1)$ satisfying $\alpha < \gamma$, and any string $x \in \{0,1\}^n$, we have*

$$\Pr_{z \sim \mathcal{U}_m}\left[\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x \circ z) \geq (\ell + m)\text{-}\mathsf{pnK}^t_{\lambda,\gamma-\alpha}(x) + m\right] \geq \alpha.$$

Proof. Let s be the smallest integer satisfying

$$\Pr_{z \sim \mathcal{U}_m}\left[\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x \circ z) \leq s\right] \geq 1 - \alpha.$$

Then by the definition of ℓ-pnK, we have

$$\Pr_{z \sim \mathcal{U}_m}\left[\Pr_{\hat{r} \sim \mathcal{U}_\ell}\left[\mathsf{pnK}^t_\lambda(x \circ z \circ \hat{r}) \leq s + \ell\right] \geq \gamma\right] \geq 1 - \alpha.$$

Therefore, we have

$$\Pr_{\substack{z \sim \mathcal{U}_m \\ \hat{r} \sim \mathcal{U}_\ell}}\left[\mathsf{pnK}^t_\lambda(x \circ z \circ \hat{r}) \leq s + \ell\right] \geq \gamma(1 - \alpha) \geq \gamma - \alpha.$$

By the definition of ℓ-pnK, we get

$$(\ell + m)\text{-}\mathsf{pnK}^t_{\lambda,\gamma-\alpha}(x) \leq s - m.$$

In other words, $s \geq (\ell+m)\text{-}\mathsf{pnK}^t_{\lambda,\gamma-\alpha}(x)+m$. Now by minimality of s, we get

$$\Pr_{z\sim\mathcal{U}_m}\left[\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x\circ z)\leq s-1\right]<1-\alpha.$$

In other words, $\mathbf{Pr}_{w\sim\mathcal{U}_m}[\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x\circ w)\geq s]>\alpha$. Substituting s, we get

$$\Pr_{z\sim\mathcal{U}_m}\left[\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x\circ z)\geq(\ell+m)\text{-}\mathsf{pnK}^t_{\lambda,\gamma-\alpha}(x)+m\right]\geq\alpha.$$

This completes the proof. □

Lemma 72. *There exists a polynomial p, such that the following holds. For any string $x,y\in\{0,1\}^*$ and any parameters $\ell,t,t_1,t_2\in\mathbb{N}^+$ and $\gamma,\gamma',\lambda,\lambda'\in(0,1)$ satisfying the following constraints*:

- $t_2\leq\ell$,
- $t\geq t_1+p(|x|+|y|+\ell)$,
- $(1-\gamma)\geq(1-\gamma')+(1-\lambda')$,

we have

$$\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x\circ y)\leq\ell\text{-}\mathsf{pnK}^{t_1}_{\lambda,\gamma'}(x)+\mathsf{pnK}^{t_2}_{\lambda'}(y\mid x)+\log p(|x|+|y|+\ell).$$

Proof. The idea is, if $t_2\leq\ell$, then we can use the random string generated along with x for $\mathsf{pnK}^{t_2}_{\lambda'}(y\mid x)$. Let $\ell\text{-}\mathsf{pnK}^{t_1}_{\lambda,\gamma'}(x)=s_1$, and $\mathsf{pnK}^{\ell}_{\lambda'}(y\mid x)=s_2$. Without loss of generality, we assume that $s_2\leq|x|+|y|$, because otherwise, by Lemma 65, when p is a large enough polynomial, we have $\ell\text{-}\mathsf{pnK}^{p(|x|+|y|+\ell)}_{\lambda,\gamma}(x\circ y)\leq|x|+|y|+\log p(|x|+|y|+\ell)$, which finishes the proof. For any string $\hat{r}\in\{0,1\}^\ell$, we define the following conditions:

(C1) $\hat{r}$ satisfies $\mathsf{pnK}^{t_1}_{\lambda}(x\circ\hat{r})\leq s_1+\ell$.

(C2) $\hat{r}$ satisfies $\mathsf{nK}^{\ell}(y\mid x;\hat{r})\leq s_2$.

Then we have the following claim:

Claim 73. *If $\hat{r}$ satisfies both (C1) and (C2), then for large enough polynomial p, we have* $\mathsf{pnK}^t_\lambda(x\circ y\circ\hat{r})\leq s_1+s_2+\ell+\log p(|x|+|y|+\ell)$.

Proof of Claim 73. By (C1), we have

$$\Pr_{r\sim\mathcal{U}_{t_1}}\left[\mathsf{nK}^{t_1}(x\circ\hat{r}\mid\epsilon;r)\leq s_1+\ell\right]\geq\lambda.$$

By randomness extension (Definition 12), for any $t'\geq t_1$, we get

$$\Pr_{r'\sim\mathcal{U}_{t'}}\left[\mathsf{nK}^{t_1}(x\circ\hat{r}\mid\epsilon;r')\leq s_1+\ell\right]\geq\lambda. \tag{44}$$

By (C2), we have

$$(x;\hat{r})\to y\text{ in time }\ell\text{ and size }s_2. \tag{45}$$

Let Π be the program of Equation (45). Let M be the Turing machine which gives efficient simulation for the computational model (Definition 12), then for a large enough polynomial p_0, since $s_2\leq|x|+|y|$, we have

- $\forall w \in \{0,1\}^\ell$, $M(\Pi; x; u; \hat{r})$ outputs either y or $\perp$ in $p_0(|x| + |y| + \ell)$ steps,
- $\exists w \in \{0,1\}^\ell$, $M(\Pi; x; u; \hat{r})$ outputs y in $p_0(|x| + |y| + \ell)$ steps.

For any integer i, let $\widetilde{i}$ denote the binary encoding of i with each bit duplicated, concatenated with 01. Let M' be the Turing machine satisfying the following.

> For any $\Pi, x, w, \hat{r} \in \{0,1\}^*$ and any $\ell \in \mathbb{N}$ satisfying $\ell \le |u|$, on input $(\widetilde{|x|} \circ \widetilde{\ell} \circ \Pi; x \circ \hat{r}; w; \epsilon)$, M' runs $M(\Pi; x; w_\ell; \hat{r})$ to obtain y, where w_ℓ denotes the length-ℓ prefix of w. If $y \neq \perp$, it outputs $x \circ y \circ \hat{r}$; otherwise it outputs $\perp$.

Then for a large enough polynomial p_1, for any $t' \ge t$ we have

- $\forall w \in \{0,1\}^{t'}$, $M'(\widetilde{|x|} \circ \widetilde{\ell} \circ \Pi; x \circ \hat{r}; w; \epsilon)$ outputs either $(x \circ y \circ \hat{r})$ or $\perp$ in $p_1(|x| + |y| + \ell)$ steps,
- $\exists w \in \{0,1\}^{t'}$, $M'(\widetilde{|x|} \circ \widetilde{\ell} \circ \Pi; x \circ \hat{r}; w; \epsilon)$ outputs $(x \circ y \circ \hat{r})$ in $p_1(|x| + |y| + \ell)$ steps.

By efficient universality (Definition 12), when p_2 is a large enough polynomial, we get

$$(x \circ \hat{r}) \to (x \circ y \circ \hat{r}) \text{ in non-deterministic time } p_2(|x| + |y| + \ell) \text{ and size } (s_2 + \log p_2(|x| + |y| + \ell)). \quad (46)$$

By apply efficient composition (Definition 12) to Equations (44) and (46), when p is a large enough polynomial, we get

$$\mathop{\mathbf{Pr}}_{r' \sim \mathcal{U}_{t_1 + p(|x|+|y|+\ell)}} \left[\mathsf{nK}^{t_1 + p(|x|+|y|+\ell)}(x \circ y \circ \hat{r} \mid \epsilon; r') \le s_1 + s_2 + \ell + \log p(|x| + |y| + \ell) \right] \ge \lambda.$$

Since $t \ge t_1 + p(|x| + |y| + \ell)$, we get

$$\mathsf{pnK}^t_\lambda(x \circ y \circ \hat{r}) \le s_1 + s_2 + \ell + \log p(|x| + |y| + \ell).$$

This completes the proof of the claim. $\diamond$

By the definitions of ℓ-pnK and pnK, for $\hat{r} \sim \mathcal{U}_\ell$, $\hat{r}$ satisfies (C1) with probability at least γ', and $\hat{r}$ satisfies (C2) with probability at least λ'. By union bound, with probability at least $1-(1-\gamma')-(1-\lambda') \ge \gamma$, $\hat{r}$ satisfies both (C1) and (C2). Hence by the definition of ℓ-pnK and Claim 73, we have

$$\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x \circ y) \le \ell\text{-}\mathsf{pnK}^{t_1}_{\lambda,\gamma'}(x) + \mathsf{pnK}^\ell_{\lambda'}(y \mid x) + \log p(|x| + |y| + \ell).$$

Since $t_2 \le \ell$, by randomness & non-determinism extension (Definition 12), we have $\mathsf{pnK}^\ell_{\lambda'}(x) \le \mathsf{pnK}^{t_2}_{\lambda'}(x)$. Hence we get

$$\ell\text{-}\mathsf{pnK}^t_{\lambda,\gamma}(x \circ y) \le \ell\text{-}\mathsf{pnK}^{t_1}_{\lambda,\gamma'}(x) + \mathsf{pnK}^{t_2}_{\lambda'}(y \mid x) + \log p(|x| + |y| + \ell).$$

as desired. $\square$

C.2.3 Main Lemma: Bounding Conditional pK

Lemma 74. *Suppose* $\mathsf{MINpnKT} \in \mathsf{prBPP}$. *Then there exists a polynomial* p *and an integer* $N_0 \in \mathbb{N}^+$ *such that the following holds. For any* $n, m, \tau, \ell, t, b, d \in \mathbb{N}^+$ *and any string* $x \in \{0,1\}^n, y \in \{0,1\}^m$ *satisfying the following constraints*:

- $2/3 < b/n^3, d/n^3 < 1$,
- $N_0 \le n \le m \le \tau$.
- $\tau \le \ell \le \tau^4$,
- $t \ge 2p(\tau)$,

we have

$$\mathsf{pK}^{p(t)}(y \mid x) \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{pnK}^{t-p(\tau)}_{b/n^3,(d+1)/n^3}(x) - (\ell+\tau^3)\text{-}\mathsf{pnK}^{t+p(\tau)}_{(b-1)/n^3,(d-2)/n^3}(x) + \log p(t).$$

Proof. Let p_1, p_2, p_3, p_4 be the polynomials defined in Lemmas 65, 67, 70 and 72 respectively. Without loss of generality, we assume they are all monotone. We define k as

$$k := \ell\text{-}\mathsf{pnK}^{t-p_3(\tau^8)}_{b/n^3,d/n^3}(x \circ y) - (\ell+\tau^3)\text{-}\mathsf{pnK}^{t+p_2(\tau^6)}_{(b-1)/n^3,(d-2)/n^3}(x) + \lceil \log p_3(\tau^8) \rceil + \lceil \log p_2(\tau^6) \rceil + 1. \quad (47)$$

By our assumptions on the parameters, when N_0 is large enough, we have $n + m + \ell + \tau^3 \le \tau^6$. Hence we have

$$\begin{aligned}
k &\ge \ell\text{-}\mathsf{pnK}^{t-p_3(\tau^8)}_{b/n^3,d/n^3}(x \circ y) - (\ell+\tau^3)\text{-}\mathsf{pnK}^{t+p_2(\tau^6)}_{(b-1)/n^3,(d-2)/n^3}(x) + \log p_2(\tau^6) + 1 \\
&\ge \ell\text{-}\mathsf{pnK}^{t-p_3(\tau^8)}_{b/n^3,d/n^3}(x \circ y) - \ell\text{-}\mathsf{pnK}^{t}_{(b-1)/n^3,(d-2)/n^3}(x \circ y) + 1 && \text{(By Lemma 67)} \\
&\ge \ell\text{-}\mathsf{pnK}^{t-p_3(\tau^8)}_{b/n^3,d/n^3}(x \circ y) - \ell\text{-}\mathsf{pnK}^{t-p_3(\tau^8)}_{b/n^3,d/n^3}(x \circ y) + 1 && \text{(By Lemma 66)} \\
&= 1. && (48)
\end{aligned}$$

Next, we upper bound and lower bound the value of k, and use DPG reconstruction lemma to finish the proof.

Upper Bound for k. Since $n \le m \le \tau$, by Lemma 72, we get

$$\ell\text{-}\mathsf{pnK}^{t-p_3(\tau^8)}_{b/n^3,d/n^3}(x \circ y) \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{pnK}^{t-p_3(\tau^8)-p_4(\tau^6)}_{b/n^3,(d+1)/n^3}(x) + \log p_4(\tau^6). \quad (49)$$

Therefore, by combining Equations (47) and (49), we have

$$\begin{aligned}
k \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{pnK}^{t-p_3(\tau^8)-p_4(\tau^6)}_{b/n^3,(d+1)/n^3}(x) - (\ell+\tau^3)\text{-}\mathsf{pnK}^{t+p_2(\tau^6)}_{(b-1)/n^3,(d-2)/n^3}(x) \\
+ \log p_3(\tau^8) + \log p_2(\tau^6) + \log p_4(\tau^6) + 3.
\end{aligned}$$

Therefore, when q_1 is a large enough polynomial, we get

$$k \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{pnK}^{t-q_1(\tau)}_{b/n^3,(d+1)/n^3}(x) - (\ell+\tau^3)\text{-}\mathsf{pnK}^{t+q_1(\tau)}_{(b-1)/n^3,(d-2)/n^3}(x) + \log q_1(\tau). \quad (50)$$

Lower Bound for k. We first show a loose upper bound on k when p and N_0 are large enough. Based on this, we give a lower bound for k. When p is a large enough polynomial such that $p(z) \geq p_3(z^8) + p_1(z^6)$, $t \geq p(\tau) \geq p_3(\tau^8) + p_1(\tau^6)$. By Lemma 65, $\ell\text{-}\mathsf{pnK}^{t-p_3(\tau^8)}_{b/n^3,d/n^3}(x \circ y) \leq n + m + \log p_1(n+m+\ell) \leq 2\tau + \log p_1(\tau^6)$. By Lemma 64, we also have $(\ell+\tau^3)\text{-}\mathsf{pnK}^{t+p_2(\tau^6)}_{(b-1)/n^3,(d-2)/n^3}(x) \geq -3$. Therefore, we have a loose upper bound for k:

$$k \leq 2\tau + \log p_1(\tau^6) + \log p_3(\tau^8) + \log p_2(\tau^6) + 6.$$

Since $\tau \geq N_0$, when N_0 is a large enough constant, we have $k \leq 3\tau$. Therefore, we get $mk + k \leq \tau^3$, and $\tau^6 \geq \ell + \tau^3 + n$. By Lemma 67, we get

$$\begin{aligned}(\ell+\tau^3)\text{-}\mathsf{pnK}^{t+p_2(\tau^6)}_{(b-1)/n^3,(d-2)/n^3}(x) - \log p_2(\tau^6) &\leq (\ell+\tau^3)\text{-}\mathsf{pnK}^{t+p_2(\ell+\tau^3+n)}_{(b-1)/n^3,(d-2)/n^3}(x) - \log p_2(\ell+\tau^3+n)\\ &\leq (\ell+mk+k)\text{-}\mathsf{pnK}^{t}_{(b-1)/n^3,(d-2)/n^3}(x).\end{aligned}$$

We also have $\tau^8 \geq nmk\ell$, which gives us

$$\ell\text{-}\mathsf{pnK}^{t-p_3(\tau^8)}_{b/n^3,d/n^3}(x \circ y) + \log p_3(\tau^8) \geq \ell\text{-}\mathsf{pnK}^{t-p_3(nmk\ell)}_{b/n^3,d/n^3}(x \circ y) + \log p_3(nmk\ell).$$

Therefore, we can now lower bound k:

$$\begin{aligned}k &= \ell\text{-}\mathsf{pnK}^{t-p_3(\tau^8)}_{b/n^3,d/n^3}(x \circ y) - (\ell+\tau^3)\text{-}\mathsf{pnK}^{t+p_2(\tau^6)}_{(b-1)/n^3,(d-2)/n^3}(x) + \lceil \log p_3(\tau^8) \rceil + \lceil \log p_2(\tau^6) \rceil + 1\\ &\geq \ell\text{-}\mathsf{pnK}^{t-p_3(nmk\ell)}_{b/n^3,d/n^3}(x \circ y) + \log p_3(nmk\ell) - (\ell+mk+k)\text{-}\mathsf{pnK}^{t}_{(b-1)/n^3,(d-2)/n^3}(x) + 1. \qquad (51)\end{aligned}$$

Distinguisher for the Direct Product Generator. By Equation (18), when N_0 is large enough, $k \geq 1$ always holds. Here we define an algorithm D to distinguish $\mathsf{DP}_k(y;\mathcal{U}_{mk})$ from $\mathcal{U}_{mk+k}$. We set the threshold

$$s = \ell\text{-}\mathsf{pnK}^{t-p_3(nmk\ell)}_{b/n^3,d/n^3}(x \circ y) + mk + \lfloor \log p_3(nmk\ell) \rfloor. \qquad (52)$$

By Lemma 69, $\mathsf{MINpnKT} \in \mathsf{BPP}$ implies $\mathsf{MIN}\ell\text{-}\mathsf{pnKT} \in \mathsf{prBPP}$. Let B be the polynomial-time algorithm that decides $\mathsf{MIN}\ell\text{-}\mathsf{pnKT}$ with error $1/n^4$. We define the distinguisher D as follows:

1. D gets input $z \in \{0,1\}^{mk+k}$.
2. D outputs $B(x \circ z, 1^s, 1^t, 1^\ell, 1^{n^3}, 1^b, 1^{n^3}, 1^d)$.

In fact, if $z \sim \mathsf{DP}_k(y;\mathcal{U}_{mk})$, then by Lemma 70 and eq. (52), we get

$$\Pr_{w\sim\mathcal{U}_{mk}}\left[\ell\text{-}\mathsf{pnK}^{t}_{b/n^3,d/n^3}(x \circ \mathsf{DP}_k(y;w)) \leq s\right] = 1.$$

Therefore, we have

$$\Pr_{\substack{w\sim\mathcal{U}_{mk}\\ \text{Randomness of } D}}[D(\mathsf{DP}_k(y;w)) = 1] \geq 1 - \frac{1}{n^4}. \qquad (53)$$

On the other hand, if $z \sim \mathcal{U}_{mk+k}$, then by Lemma 71, we get

$$\Pr_{w\sim\mathcal{U}_{mk+k}}\left[\ell\text{-}\mathsf{pnK}^{t}_{(b-1)/n^3,(d-1)/n^3}(x \circ w) \geq (\ell+mk+k)\text{-}\mathsf{pnK}^{t}_{(b-1)/n^3,(d-2)/n^3}(x) + mk + k\right] \geq \frac{1}{n^3}.$$

But by Equations (51) and (52), we have

$$(\ell+mk+k)\text{-}\mathsf{pnK}^{t}_{(b-1)/n^3,(d-2)/n^3}(x) + mk + k \geq s + 1.$$

Combining these two inequalities, we get

$$\mathop{\mathbf{Pr}}_{w\sim\mathcal{U}_{mk+k}}\left[\ell\text{-}\mathsf{pnK}^t_{(b-1)/n^3,(d-1)/n^3}(x\circ w)\geq s+1\right]\geq\frac{1}{n^3}.$$

Therefore, we have

$$\mathop{\mathbf{Pr}}_{\substack{w\sim\mathcal{U}_{mk+k}\\ \text{Randomness of } D}}[D(w)=1]\leq 1-\frac{1}{n^3}+\frac{1}{n^4}. \tag{54}$$

By comparing Equation (53) and Equation (54), we see that D distinguishes $\mathsf{DP}_k(y;\mathcal{U}_{mk})$ from $\mathcal{U}_{mk+k}$ with probability $1/n^3-2/n^4\geq 1/n^4$. Then by Corollary 24, there exists some polynomial p_{DP} such that

$$\mathsf{pK}^{p_{\mathsf{DP}}(t),\|D}(y)\leq k+\log p_{\mathsf{DP}}(t).$$

We can store the program of D in the description, as well as t, s, ℓ, which takes no more than $O(\log t)$ bits. If we have x, then we can simulate D in time $\mathsf{poly}(t)$, and answer the oracle queries. Therefore, when q_2 is a large enough polynomial, we have

$$\mathsf{pK}^{q_2(t)}(y\mid x)\leq k+\log q_2(t). \tag{55}$$

Putting It All Together. To summarize the previous paragraphs, when p is a large enough polynomial and N_0 is a large enough constant, by Equation (50) there exists some polynomial q_1 such that

$$k\leq\mathsf{pnK}^\tau_{1-1/n^3}(y\mid x)+\ell\text{-}\mathsf{pnK}^{t-q_1(\tau)}_{b/n^3,(d+1)/n^3}(x)-(\ell+\tau^3)\text{-}\mathsf{pnK}^{t+q_1(\tau)}_{(b-1)/n^3,(d-2)/n^3}(x)+\log q_1(\tau).$$

Also, by Equation (55), there exists some polynomial q_2 such that

$$k\geq\mathsf{pK}^{q_2(t)}(y\mid x)-\log q_2(t).$$

Combining these two inequalities, we get

$$\begin{aligned}\mathsf{pK}^{q_2(t)}(y\mid x)\leq\mathsf{pnK}^\tau_{1-1/n^3}(y\mid x)+\ell\text{-}\mathsf{pnK}^{t-q_1(\tau)}_{b/n^3,(d+1)/n^3}(x)\\ -(\ell+\tau^3)\text{-}\mathsf{pnK}^{t+q_1(\tau)}_{(b-1)/n^3,(d-2)/n^3}(x)+\log q_1(\tau)+\log q_2(t).\end{aligned}$$

Therefore, when p is a large enough polynomial, we get

$$\mathsf{pK}^{p(t)}(y\mid x)\leq\mathsf{pnK}^\tau_{1-1/n^3}(y\mid x)+\ell\text{-}\mathsf{pnK}^{t-p(\tau)}_{b/n^3,(d+1)/n^3}(x)-(\ell+\tau^3)\text{-}\mathsf{pnK}^{t+p(\tau)}_{(b-1)/n^3,(d-2)/n^3}(x)+\log p(t).$$

This concludes the proof. □

C.2.4 Proof of Theorem 62

We first state and prove the following lemma, then we prove Theorem 62 using this lemma.

Lemma 75. *Let $L=\mathsf{SAT}$, and define $L_n=L\cap\{0,1\}^n$. Suppose $L\in\mathsf{NTIME}[n^c]$. For a given $x\in\{0,1\}^n$, define $A_x\subseteq\{0,1\}^{n^c}$ to be the set of witnesses for x. If $\mathsf{MINpnKT}\in\mathsf{prBPP}$, then there exists a polynomial p such that for any n, any $x\in L_n$ and any $y\in A_x$, we have*

$$\mathsf{pK}^{p(n)}(y\mid x)\leq\log|A_x|+\log p(n).$$

Proof. If $x \in L_n$, then by Theorem 4, there exists some polynomial p_{LC} such that for any $y \in A_x$, $\mathsf{pnK}^{p_{\mathsf{LC}}(n^c),A_x}(y) \le \log |A_x| + \log p_{\mathsf{LC}}(n^c)$. Since A_x is decidable in time n^c given x, and by Lemma 29 we can amplify the success probability, we get that there exists some polynomial p_1 such that $p_1(z) \ge z^d$, and

$$\mathsf{pnK}^{p_1(n)}_{1-1/n^3}(y \mid x) \le \log |A_x| + \log p_1(n). \tag{56}$$

Let p_2 be the polynomial stated in Lemma 74. Let $m = n^c, \tau = p_1(n)$. Then by Lemma 74, when n is large enough, for any $\ell, t, b, d \in \mathbb{N}^+$ satisfying $2/3 < b/n^3, d/n^3 < 1$, $\tau \le \ell \le \tau^4$, $t \ge p_2(\tau)$, and any string $x \in \{0,1\}^n$, $y \in \{0,1\}^{n^c}$, we have

$$\mathsf{pK}^{p_2(t)}(y \mid x) \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \ell\text{-}\mathsf{pnK}^{t-p_2(\tau)}_{b/n^3,(d+1)/n^3}(x) - (\ell+\tau^3)\text{-}\mathsf{pnK}^{t+p_2(\tau)}_{(b-1)/n^3,(d-2)/n^3}(x) + \log p_2(t).$$

Now, setting t to be some polynomial of n, if we can bound

$$\ell\text{-}\mathsf{pnK}^{t-p(\tau)}_{b/n^3,(d+1)/n^3}(x) - (\ell+\tau^3)\text{-}\mathsf{pnK}^{t+p(\tau)}_{(b-1)/n^3,(d-2)/n^3}(x)$$

by a constant, then using Equation (56), we are done. Even though we don't know how to do this directly, we can use the telescoping trick. More specifically, let p_3 be the polynomial stated in Lemma 65. We define four sequences $\{\ell_i\}_{i=0}^n$, $\{t_i\}_{i=0}^n$, $\{b_i\}_{i=0}^n$ and $\{d_i\}_{i=0}^n$ inductively:

- $\ell_0 = \tau^3$; $\ell_i = \ell_{i-1} + \tau^3, \forall i \in [n]$.
- $t_0 = 2p_2(\tau) + p_3(\tau^6)$; $t_i = t_{i-1} + 2p_2(\tau), \forall i \in [n]$.
- $b_0 = \lfloor 3n^3/4 \rfloor$; $b_i = b_{i-1} - 1, \forall i \in [n]$.
- $d_0 = \lfloor 3n^3/4 \rfloor$; $d_i = d_{i-1} - 3, \forall i \in [n]$.

It is not hard to verify that for any $i \in \{0, \dots, n\}$, ℓ_i, t_i, b_i, d_i satisfies the conditions of Lemma 74. Hence we get

$$\mathsf{pK}^{p_2(t_i)}(y \mid x) \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \ell_i\text{-}\mathsf{pnK}^{t_i-p_2(\tau)}_{b_i/n^3,(d_i+1)/n^3}(x) - (\ell_i+\tau^3)\text{-}\mathsf{pnK}^{t_i+p_2(\tau)}_{(b_i-1)/n^3,(d_i-2)/n^3}(x) + \log p_2(t_i). \tag{57}$$

Now using a telescoping argument, we get

$$\begin{aligned}
&\ell_0\text{-}\mathsf{pnK}^{t_0-p_2(\tau)}_{b_0/n^3,(d_0+1)/n^3}(x) - \ell_n\text{-}\mathsf{pnK}^{t_n-p_2(\tau)}_{b_n/n^3,(d_n+1)/n^3}(x) \\
&= \sum_{i=0}^{n-1} \left(\ell_i\text{-}\mathsf{pnK}^{t_i-p_2(\tau)}_{b_i/n^3,(d_i+1)/n^3}(x) - \ell_{i+1}\text{-}\mathsf{pnK}^{t_{i+1}-p_2(\tau)}_{b_{i+1}/n^3,(d_{i+1}+1)/n^3}(x) \right) \\
&= \sum_{i=0}^{n-1} \left(\ell_i\text{-}\mathsf{pnK}^{t_i-p_2(\tau)}_{b_i/n^3,(d_i+1)/n^3}(x) - (\ell_i+\tau^3)\text{-}\mathsf{pnK}^{t_i+p_2(\tau)}_{(b_i-1)/n^3,(d_i-2)/n^3}(x) \right)
\end{aligned}$$

By Lemma 65, we have

$$\ell_0\text{-}\mathsf{pnK}^{t_0-p_2(\tau)}_{b_0/n^3,(d_0+1)/n^3}(x) \le \tau^3\text{-}\mathsf{pnK}^{p_3(\tau^6)}_{b_0/n^3,(d_0+1)/n^3}(x) \le n + \log p_3(\tau^6),$$

which is at most $2n$ for all large enough n. By averaging, there must be some $i \in \{0, \dots, n-1\}$ such that $\left(\ell_i\text{-}\mathsf{pnK}^{t_i-p_2(\tau)}_{b_i/n^3,(d_i+1)/n^3}(x) - (\ell_i+\tau^3)\text{-}\mathsf{pnK}^{t_i+p_2(\tau)}_{(b_i-1)/n^3,(d_i-2)/n^3}(x) \right) \le 2$. For such an i, by Equation (57) we have

$$\mathsf{pK}^{p_2(t_i)}(y \mid x) \le \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \log p_2(t_i) + 2.$$

Note that $t_i \leq t_n$. Then the above yields

$$\mathsf{pK}^{p_2(t_n)}(y \mid x) \leq \mathsf{pnK}^{\tau}_{1-1/n^3}(y \mid x) + \log p_2(t_n) + 2.$$

Combining this with Equation (56), we get that for all large enough n, for any $x \in L_n$

$$\mathsf{pK}^{p_2(t_n)}(y \mid x) \leq \log |A_x| + \log p_1(n) + \log p_2(t_n) + 2.$$

Notice that $t_n \leq (2n+2)p_2(\tau) + p_3(\tau^6) = (2n+2)p_2(p_1(n)) + p_3(p_1^6(n))$, which is polynomially bounded by n. Hence for a large enough polynomial p, we have

$$\mathsf{pK}^{p(n)}(y \mid x) \leq \log |A_x| + \log p(n),$$

as desired. □

Proof of Theorem 62 from Lemma 75. Our randomized polynomial-time algorithm B for deciding $L = \mathsf{SAT}$ works as follows:

1. B gets $x \in \{0,1\}^n$ as input.
2. B takes a sample y from $\mathsf{USamp}(1^{n^c}, 1^{p(n)}, x)$.
3. B checks if y is a witness for x.
4. B repeats step 2 and 3 for $O(n^c \cdot p(n))$ rounds. If at any round, y is indeed a witness for x, B accepts; otherwise B rejects.

If $x \notin L_n$, then $A_x = \varnothing$, so B never accepts x. If $x \in L_n$, then by Lemma 75 and Proposition 26, for any $y \in A_x$, the probability that $\mathsf{USamp}(1^{n^c}, 1^{p(n)}, x) = y$ is at least $\Omega(1/(n^c \cdot p(n) \cdot |A_x|))$. Summing the probability, we get

$$\mathop{\mathbf{Pr}}_{y \sim \mathsf{USamp}(1^{n^c}, 1^{p(n)}, x)}[y \in A_x] \geq \Omega\left(\frac{1}{n^c \cdot p(n)}\right).$$

Therefore, if we sample for $O(n^c \cdot p(n))$ times, we can obtain some $y \in A_x$ with probability $\Omega(1)$. Hence B accepts x with probability $\Omega(1)$. Hence we get $L \in \mathsf{BPP}$. Since $L = \mathsf{SAT}$ is NP-complete, we conclude that $\mathsf{NP} \subseteq \mathsf{BPP}$. □

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il