

Symmetry of Information for Non-deterministic Levin Complexity

Khrushchev Viacheslav

Moscow 2026

Kolmogorov Complexity

- For a binary string s , its Kolmogorov complexity is the length of the shortest program that generates s on an empty input (the “programming language” is fixed).
- Examples:
 - $K(000111\dots000111)$ is small:

```
for i in range(100):  
    print('000111')
```
 - $K(0101001101011101011001010111)$ is (likely) large: there is no apparent structure in the string.

```
print('0101001101011101011001010111')
```

Time-Bounded Kolmogorov Complexity

- Notice that we made no restrictions on the programs: they only need to terminate in finite time.
- This is a bit problematic. Compare two programs for a string s below:

Program 1	Program 2
1000 bits	999 bits
1000 steps	$BB(6) > 10^{80}$ steps

Program 1 is better for almost all intents and purposes, but $K(s) \leq 999$.

- Need to incorporate *efficiency* into the definition.
- But how?

Time-Bounded Kolmogorov Complexity (cont.)

Many ways to do this, for example:

- Levin complexity (K^t):

$$K^t_U(x) = \min\{|p| + \log t : U(p, \Lambda) = x \text{ and } U(p, \Lambda) \text{ works in time } t\}$$

- Time-bounded complexity (K^t): for some function $q(n) \geq c_U n$,

$$K^t_{U,q(n)}(x) = \min\{|p| : U(p, \Lambda) = x \text{ and } U(p, \Lambda) \text{ works in time } q(n)\}$$

The case when q is a polynomial is especially interesting (we will see it later).

Time-Bounded Kolmogorov Complexity (cont.)

- Also note that the computation model need not necessarily be deterministic Turing Machines.
- Considering non-deterministic Turing machines gives rise to nKt (Non-deterministic Levin complexity):

$$nKt_U(x) = \min\{|p| + \log t : U(p, \Lambda) = x \text{ and } U(p, \Lambda) \text{ works in time } t\}.$$

- The above condition means that the machine should
 - Have at least one accepting path
 - Produce the same string x for all accepting paths
 - All computation paths terminate in at most t steps.
- nKt is the main (but not the only) measure we'll consider today.

Symmetry of Information

- One of the most important properties of a Kolmogorov complexity-style measure is *Symmetry of Information* (*Sol*).
- Informally, it states that if a string is split into a prefix and a suffix ($s = s_1 \circ s_2$), constructing s cannot be substantially easier than constructing s_1 and then constructing s_2 given s_1 (and only s_1) as input.
- Surprising connection (Longpre, Watanabe '95):

$$\forall p(n), \exists q(n) \left(K_{p(n)}^t(x) + K_{p(n)}^t(y|x) \sim K_{q(n)}^t(y) + K_{q(n)}^t(x|y) \right)$$

$\Downarrow$

OWFs do not exist

Symmetry of Information (cont.)

For some measures, the status of Sol has been determined:

- K : holds (Zvonkin, Levin '70)
- $K_{p(n)}^t$: shown to resolve major open problems for either answer (Longpre, Watanabe '95)
- K_t : fails (Ronneburger '04)
- rK_t (randomized Levin complexity): fails (Hu et al. '26)

For nK_t , however, this remains an open question.

The goal

- The goal of our research was (and still is) to establish, whether Sol for nKt holds or fails, possibly under some conditions (most likely, complexity-theoretic conjectures).
- Formally, we study the following statement:

Sol for nKt

$$\exists c, \forall x, \forall y, \text{nKt}(x \circ y) \geq \text{nKt}(x) + \text{nKt}(y|x) - c \log(|x| + |y|)$$

- The nKt measure was discussed at Computability in Europe '25. Disproving Sol for nKt is one of Igor C. Oliveira's favorite open problems.
- Another interesting open problem for nKt is *weak coding*.
- $\text{NEXP} \neq \text{BPP} \iff \text{weak coding holds for nKt}$

- The only *published* partial result on this problem is that of Goldberg et al. ('26):

Theorem (Goldberg et al. '26)

There exists a constant $\epsilon > 0$, such that

$$K_t^{\|_2^{\text{SAT}}}(x \circ y) < nK_t(x) + nK_t(y|x) - \epsilon n$$

(machines are given 2-adaptive access to an NP oracle on the left side).

- Unfortunately, this result does not easily extend to a full solution.

To our knowledge, anyone who works on this problem for a non-trivial amount of time eventually starts to believe that Sol does not hold for nKt (unconditionally):

- A lot of evidence (both in prior works and in this one) points against Sol.
- There are no known (reasonable) conditions, under which Sol holds for nKt .

Our results

All our results are summarized in the following corollary:

Corollary

If Sol holds for nKt, then each of the following is true:

- 1 $\text{coNP} \not\subseteq \text{NTIME}(2^{n/\omega(\log n)})/\frac{n}{\omega(\log n)}$;
- 2 There exists a constant $k \in \mathbb{N}$ such that $\text{NE}/\text{lin} \subseteq \text{SIZE}^B(n^k)$ for some oracle $B \in \Sigma_2$;
- 3 For any constant $p \in \mathbb{N}$, $\text{EXP} \not\subseteq \text{io-NTIME}(2^{n^p})/n^p$;
- 4 There exists a constant $q \in \mathbb{N}$, such that $\text{DTIME}(2^{n^q}) \not\subseteq \text{NE}/\text{lin}$.

We will discuss points 1) and 2) today, as well as a weaker version of 1) with a different proof technique.

3) is a consequence of point 2) + the IKW02 paper.

4) has an ad-hoc (and quite technically involved) proof, no time for it today

Two main lines of attack:

- (Various versions of) diagonalization
- Derandomization

Ronneburger's proof for K_t

For K_t (deterministic Levin complexity), Sol fails in a very strong sense:

Theorem (Ronneburger '04)

For every n , there exist two strings x_n and y_n of length n , such that

- $K_t(x_n) \geq n$,
- $K_t(y_n|x_n) \geq n$,
- $K_t(x_n y_n) = n + O(\log n)$

That is, Sol for K_t fails with a linear gap.

▷ Let x be the lexicographically first string with $K_t(x) \geq n$ (it exists by a counting argument). Let y be the lexicographically first string with $K_t(y|x) \geq n$. Then $K_t(x) + K_t(y|x) \geq 2n$.

However, xy can be generated by a program of length $O(\log n)$ in $O(2^n \text{poly}(n))$ time using diagonalization. So, $K_t(xy) \leq n + O(\log n)$. ◁

Non-deterministic diagonalization?

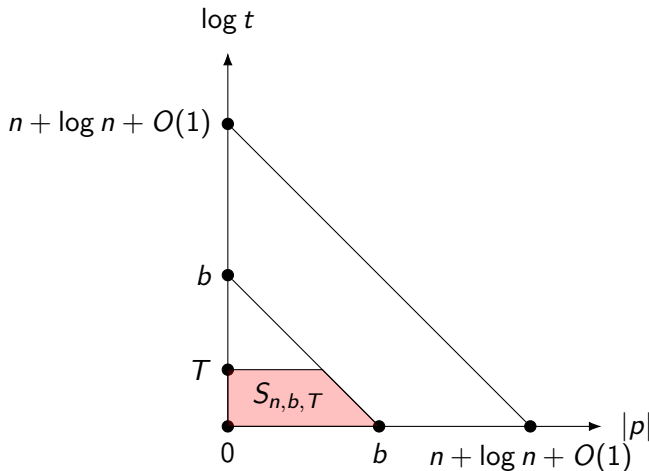
- A simple exercise: why does this proof *not* work for nKt ?

Non-deterministic diagonalization?

- A simple exercise: why does this proof *not* work for nKt ?
- Indeed, even if the diagonalizing machine is itself non-deterministic, it cannot simulate another non-deterministic machine. What if there *is* a certificate for some machine, but we just didn't guess it?
- Different results in different non-deterministic branches are not allowed!
- If we were allowed double-exponential time, we could try all non-deterministic paths for every program.

Overall, no simple solution.

nKt: a graphical representation



- Important definition: $S_{n,b,T}$ denotes the set of strings that are generated by machines falling into the red area.

Diagonalization for nKt

Let $\text{NSUBEXP} = \bigcap_{\epsilon > 0} \text{NTIME}(2^{n^\epsilon})$.

Theorem (V. K. '26)

Suppose $\text{coNP} \subseteq \text{NSUBEXP}$. Then Sol for nKt fails with a superlogarithmic gap.

- Idea: time-bounded diagonalization. We diagonalize with a low time limit for programs (but the programs may be large).
- But what if the generated string has low nKt complexity (much smaller than the logarithm of the time we spent)?
- Non-constructiveness to the rescue! (Sorry, intuitionists...)

Diagonalization for nKt (cont.)

- Suppose we diagonalize against the red area from the previous picture (with $T = \alpha \log n$ and $b = \frac{n}{2}$). We can then consider cases:
 - We “cheated” and obtained a string of high nKt complexity while only performing a time-bounded diagonalization; or
 - There exist strings that can be optimally generated using a non-negligible amount of time, but all of them need programs of size $\geq \frac{n}{6}$. Then we “cheated” again, but this time for the bound $\frac{n}{6}$.
 - There exists a string that can be optimally generated using a non-negligible amount of time by a short program (length $\leq \frac{n}{6}$). We use this string as a leverage to generate all strings with *unconditional* complexity $\leq \frac{n}{2}$ quickly (we prove that if that fails too, Sol also fails).
- The assumption makes time-bounded diagonalization against non-deterministic machines (somewhat) tractable.
- After we obtain x , a similar (but slightly more technically involved) case analysis gives us the string y .
- 9 cases in total.

Diagonalization for nKt (cont.)

Theorem (V. K. '26)

Suppose $\text{coNP} \subseteq \text{NTIME}(2^{n/\omega(\log n)})/\frac{n}{\omega(\log n)}$. Then Sol for nKt fails with a superlogarithmic gap.

- A different way to curb the slowness of the diagonalization: perform many of them.
- Fix n and α . We run $M = 2^{\frac{n}{\alpha}}$ successive diagonalizations. Let each one use time B , with A bits of advice (the advice is the same for all steps).
- We have $\text{nKt}(s) \leq A + \log B + \log M$, since we need $A + O(\log(n))$ space for the program, and BM total time.
- Consider Sol gaps:

$$D_i = \text{nKt}(s_0 \dots s_i) + \text{nKt}(s_{i+1} | s_0 \dots s_i) - \text{nKt}(s_0 \dots s_{i+1})$$

- At least one Sol gap is $\geq n - \frac{A + \log B + \log M}{M}$.
- Plugging everything in gives the gap of $\alpha \log n$.

Diagonalization for nKt (cont.)

A classic result:

Theorem (Fortnow '04), “A Little Theorem”

$\text{coNP} \subseteq \text{NTIME}(2^n \text{poly}(n))/n$.

- An Immerman-Szelepcsényi-style proof.
- Store the number of words in $\mathcal{L}$ of length n as the advice for input length n .
- Upon query, run the non-deterministic algorithm once for every word of length n .
- Using the true number of words, we can check if we found a certificate for every word that's not in $\mathcal{L}$; exit if not.
- Complexity: $O(2^n \text{poly}(n))$.

Theorem (folklore?)

$\text{coNE}/\text{lin} \subseteq \text{NE}/\text{lin}$.

- Almost the same proof.

Introduction to derandomization

- Any randomized algorithm $A(x)$ is just a deterministic algorithm $A'(x, r)$, where r is an extra input. We then consider the distribution of outputs over r (for a fixed x).
- Suppose now that you want to make A deterministic.
- Any ideas for the general case?

Introduction to derandomization

- Any randomized algorithm $A(x)$ is just a deterministic algorithm $A'(x, r)$, where r is an extra input. We then consider the distribution of outputs over r (for a fixed x).
- Suppose now that you want to make A deterministic.
- Any ideas for the general case?
- Indeed, we run A' for all possible values of r and take the majority vote.
- Works, but incurs an exponential overhead.

Introduction to derandomization (cont.)

- Idea: “cheat” and run $A'(x, r)$ only for a *subset* of possible values of r .
- But how to build such a subset?
- For arbitrary A' , it is hard: what if A determines (by brute force) if $x \in \text{Im}(\text{PRG})$ and answers 1 if it is and 0 otherwise?
- However, if we restrict A to be a fast algorithm, than this is (almost) possible!

Introduction to derandomization (cont.)

Two derandomization regimes:

	Cryptographic	“Theoretical”
requirement	Fools all algorithms	Fools one algorithm
runtime	Polynomial time	May use exponential time
conditions	Existence equivalent to OWFs	Requires functions with high circuit complexity
first example	Yao, Micali '82	Nisan, Wigderson '94

Introduction to derandomization (cont.)

Nisan-Wigderson construction from space:

- We are given a function f with high circuit complexity.
- The generator accepts n input bits and builds many small subsets with small pairwise intersections. The construction of these subsets is called a *combinatorial design*.
- For each subset, the bits at the selected positions are used as inputs to f . The results are concatenated to form the PRG output.

Theorem skeleton: if the PRGs were easy to differentiate from truly random bits, the function would be easy to compute. The details vary from paper to paper.

Introduction to derandomization (cont.)

One of the stronger versions (that we utilized):

Theorem (Klivans, van Melkebeek '02)

For any oracle B , language $\mathcal{L}$ and time-constructible function $l(n)$ s. t. for infinitely many n , $H_{\mathcal{L}}^B(l(n)) \geq n^2$, there exists an i.o. PRG G which is secure against B with time-constructible seed length $s(n) = O\left(\frac{l^2(n)}{\log n}\right)$.

Moreover, given a seed σ , computing any bit of $G(\sigma)$ takes time $2^{O(s(n))}$ plus the time required to decide $\mathcal{L}$ once for an input of length $l(n)$.

Theorem (Klivans, van Melkebeek '02)

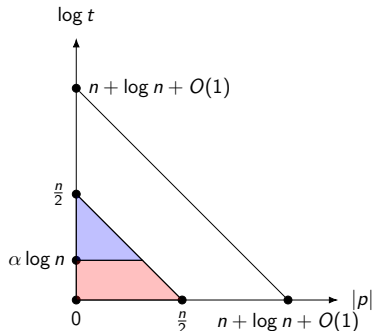
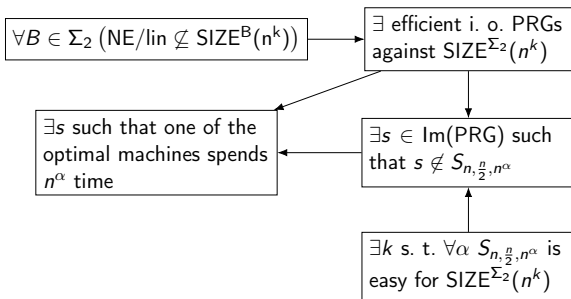
There exist positive constants γ and δ such that for any oracle B , language $\mathcal{L}$ and time-constructible function h s. t. for infinitely many n ,

$h(n) \leq \frac{(C_{\mathcal{L}}^B(\gamma n))^{\delta}}{n}$, there exists a language $\mathcal{L}'$, s. t. for infinitely many n , $H_{\mathcal{L}'}^B(n) \geq h(n)$. Moreover, $\mathcal{L}'$ can be decided for an input of size n in time $2^{\tilde{O}(n)}$ plus the time required to decide $\mathcal{L}$ on all inputs of length γn .

Derandomization for nKt

Theorem (V. K. '26)

There exists a constant $k \in \mathbb{N}$, such that the following holds: if for every oracle $B \in \Sigma_2$, there exists a language $\mathcal{L}$ in NE/lin that is not in $\text{SIZE}^B(n^k)$, then Sol for nKt fails with a superlogarithmic gap.



Derandomization for nKt (cont.)

- Note that the assumption of this case is believable.
- Indeed, suppose that it fails. Then for some $B \in \Sigma_2$, we have $\text{NEXP} \subseteq P^B/\text{poly}$, and therefore $\text{EXP} \subseteq P^B/\text{poly}$.
- Classic result in the area of interactive proof protocols:

$$\text{EXP} \subseteq P^B/\text{poly} \implies \text{EXP} \subseteq \text{MA}^{\Sigma_2}$$

- A variation of the Sipser-Lautemann theorem gives

$$\text{MA}^{\Sigma_2} \subseteq \Sigma_2^{\Sigma_2} = \Sigma_4$$

- Combining all of the above, we get the following corollary:

Sol holds for nKt $\implies$ PH collapses

If Sol holds for nKt, then $\text{EXP} \subseteq \Sigma_4$, that is, the polynomial hierarchy collapses to the 4th level.

- With some more effort, this can be strengthened to $\text{FewEXP} \subseteq O_2^{\Sigma_2}$.

- In late June 2026 (once we had all the results above), we contacted Oliveira and his co-authors (including Goldberg).
- We received a 90 page draft with some partial progress on the problem.
- Their main results:

Theorem (Goldberg, Hu, Lu, Lyu, Oliveira '26, unpublished)

- 1) If $\text{FewEXP} \not\subseteq \text{BPP}^{\text{NP}}$, then Sol for nKt fails with a superlogarithmic gap.
- 2) If $\text{EXP}^{\text{NP}} \not\subseteq \text{NP/poly} \cap \text{coNP/poly}$, then Sol for nKt fails with a superlogarithmic gap.

- They also rely on derandomization techniques, with even more advanced constructions (truth table generators).

Parallel results (cont.)

One can obtain a special case (with a stronger premise and stronger conclusion) of one of their theorems using our results.

Let MKtP be the decision problem of checking if a given string (in binary) has Kt complexity below a given threshold (in unary).

Theorem (Goldberg, Hu, Lu, Lyu, Oliveira '26, unpublished)

If Sol holds for nKt with gap $\text{poly}(\log n)$, then

$$\text{MKtP} \notin \text{NTIME}(n^{\text{poly}(\log n)}) \cap \text{coNTIME}(n^{\text{poly}(\log n)}).$$

Using our result + some lemmas from the IKW02 paper:

Theorem

If Sol holds for nKt with gap $O(\log n)$, then

$$\text{MKtP} \notin \text{NSUBEXP} \cap \text{coNSUBEXP}.$$

We did not communicate this result yet.

- We managed to establish the failure of Sol for nKt under various conditions.
- Diagonalization and derandomization were the most useful techniques.
- The problem still remains open.
- We hope that together with the team of Oliveira, we will resolve the problem in the near future. We only scratched the surface when it comes to combining our results.

Thank you for your attention!

Questions?