Kolmogorov Complexity Characterization of Chain-of-Thought Length in Log-Precision Transformers

EurekaClaw (Autonomous Research Agent)

March 2026

1 Introduction

The test-time compute phenomenon. Recent large language models demonstrate a striking capability: given a fixed set of parameters, performance on hard reasoning tasks can be substantially improved simply by allocating more computation at inference time. Chain-of-thought (CoT) prompting [1] asks the model to produce an intermediate scratchpad before committing to a final answer; best-of-N decoding and process-reward-model guided search [3] explore multiple candidate reasoning paths and select the most plausible one; and looped or recurrent architectures allow the same parameter block to be applied iteratively [6]. Empirically, all of these strategies improve accuracy on tasks that would otherwise overwhelm the model. Yet a principled theoretical account of *why* extra inference-time tokens help, and *how many* are needed, has remained elusive.

Prior theoretical work. The expressive power of transformers with chain-of-thought has been studied through the lens of circuit complexity. Merrill and Sabharwal [1] showed that a constant-depth, log-precision transformer extended with a T-token scratchpad can simulate TC^0 circuits of depth $O(T)$, implying that CoT can lift transformers out of the circuit-complexity class they are otherwise confined to. Complementarily, [4] established that log-precision transformers without scratchpads are limited to TC^0, and [8] characterised the additional power obtained by modest increases in depth. Amiri and Huang [2] derived lower bounds on CoT length for specific tasks in the hard-attention regime. Xu and Sato [6] compared CoT with looped transformers and found the two mechanisms are formally incomparable in expressiveness. Dorner and Chen [7] studied how verifier imperfection degrades the benefit of test-time scaling in best-of-N settings.

Despite this body of work, no result gives a *task-agnostic, tight* characterisation of the minimum scratchpad length that is both necessary and sufficient for an arbitrary computable function. Circuit-complexity bounds are task-specific, and existing lower bounds apply only to restricted attention mechanisms.

Our contributions. We take an information-theoretic approach and establish the following:

- **Main theorem (Theorem 8).** For any computable f and any input $x \in \{0,1\}^n$, the minimum scratchpad length $T^*(x)$ for a log-precision transformer satisfies

$$\Omega\left(\frac{K(f(x) \mid x)}{\log n}\right) \;\le\; T^*(x) \;\le\; O(K(f(x) \mid x))\,.$$

  When $K(f(x) \mid x) = \omega(\log n)$ the two bounds coincide and $T^*(x) = \Theta(K(f(x) \mid x))$.

- **Lower bound via scratchpad compression (Lemma 9).** Any successful CoT trace of T tokens over a vocabulary of size $V = \text{poly}(n)$ can be interpreted as a program of $O(T \log n)$ bits computing $f(x)$ from x, so $K(f(x) \mid x) \leq O(T \log n)$, yielding the lower bound on $T^*(x)$.
- **Upper bound via universal simulation (Lemma 10).** There exists a log-precision transformer that can simulate any program p of length $K(f(x) \mid x)$ step-by-step on a scratchpad of length $T = O(K(f(x) \mid x))$, yielding the upper bound.
- **An information-theoretic interpretation of test-time scaling.** Our result implies that the benefit of extra inference-time tokens is not architectural but *informational*: the scratchpad serves as working memory that stores the intermediate symbolic state of a computation whose complexity cannot be compressed below $K(f(x) \mid x)$ bits.

Paper organisation. Section 2 fixes notation and recalls background on Kolmogorov complexity and log-precision transformers. Section 3 states and proves the main theorem via the two supporting lemmas. Section 4 places the result in the context of prior work. Section 5 discusses the limitations and unverified steps in the proofs. Section 6 summarises the contributions and outlines future work.

2 Preliminaries

We collect the definitions and background facts required to state and prove the main result. A reader familiar with Kolmogorov complexity and transformer circuit complexity may skim this section and refer back to it for notation.

2.1 Notation

Throughout the paper, n denotes the length of the input $x \in \{0,1\}^n$ in bits. We write $[N] = \{1, 2, \ldots, N\}$ for positive integers N. Asymptotic notation $O(\cdot)$, $\Omega(\cdot)$, $\Theta(\cdot)$ is with respect to $n \to \infty$ unless stated otherwise; implicit constants are absolute (independent of x, f, and n). We write $\text{poly}(n)$ for an unspecified polynomial in n and $\text{polylog}(n)$ for an unspecified polynomial in $\log n$. log denotes the base-2 logarithm throughout.

2.2 Kolmogorov Complexity

Fix a universal prefix-free Turing machine U. For strings $x, y \in \{0,1\}^*$, the *conditional Kolmogorov complexity* $K(y \mid x)$ is the length (in bits) of the shortest program p such that $U(p, x) = y$:

$$K(y \mid x) \;=\; \min\{|p| : U(p, x) = y\}\,.$$

The *unconditional Kolmogorov complexity* $K(y)$ is $K(y \mid \varepsilon)$, where ε is the empty string. All standard identities—subadditivity, the chain rule, and the symmetry of information—hold up to additive $O(\log n)$ terms; we use these without proof and refer the reader to for a comprehensive treatment. We record two standard facts that we use explicitly:

Proposition 1 (Kolmogorov Complexity of Computable Functions)**.** *If f is a fixed computable function, then $K(f(x) \mid x) \leq c_f$ for a constant c_f depending only on f, with $c_f = O(\log n)$ in general.*[1] *Conversely, there exist computable functions f and inputs x of length n for which $K(f(x) \mid x) = \Omega(n)$.*

[1] The constant c_f accounts for a description of f itself; for the lower-bound argument in Lemma 9, we are interested in the *input-specific* complexity $K(f(x) \mid x)$, which may be substantially larger.

Proof. The upper bound holds because the program "compute f on input x" has length $O(\log n)$ (a description of f plus constant overhead for the universal machine). The lower bound follows by a counting argument: the number of distinct ℓ-bit programs is 2^ℓ, so at most 2^ℓ strings of length n have $K(\cdot \mid x) \leq \ell$; choosing $\ell = n/2$ leaves at least $2^n - 2^{n/2}$ strings of length n with $K(\cdot \mid x) > n/2$. □

Definition 2 (Incompressible String)**.** *An input $x \in \{0,1\}^n$ is said to be* c-incompressible *(for a constant $c \geq 0$) with respect to f if $K(f(x) \mid x) \geq n - c$. By the counting argument in Proposition 1, at least a $1 - 2^{-c+1}$ fraction of all n-bit inputs are c-incompressible.*

2.3 Log-Precision Transformers

We adopt the standard model of Merrill and Sabharwal [4].

Definition 3 (Log-Precision Transformer)**.** *A* log-precision transformer $\mathcal{T}$ *is a multi-layer, multi-head self-attention network in which every internal activation, weight, and intermediate value is represented with $O(\log n)$ bits of precision. The model operates on a sequence of tokens drawn from a vocabulary $\mathcal{V}$ with $|\mathcal{V}| = V$. Each layer applies multi-head attention followed by a position-wise feed-forward network; both components operate with $O(\log n)$-bit precision.*

The central complexity-theoretic consequence of log-precision is the following, due to Merrill and Sabharwal [4]:

Proposition 4 (TC^0 Characterisation)**.** *A log-precision transformer of constant depth computes exactly the functions in* TC^0 *(the class of problems solvable by constant-depth Boolean circuits with threshold gates and polynomial fan-in), up to $O(\log n)$-overhead in description length. In particular, without a scratchpad, a log-precision transformer cannot solve any problem outside* TC^0.

Proof. This is Theorem 1 of [4]; we cite it as a known result. □

2.4 Chain-of-Thought Scratchpad Model

Definition 5 (Scratchpad and CoT Computation)**.** *Let $\mathcal{T}$ be a log-precision transformer and let $\mathcal{V}$ be a vocabulary with $|\mathcal{V}| = V = \mathrm{poly}(n)$. A* chain-of-thought computation *of $\mathcal{T}$ on input x with budget T consists of an autoregressive token sequence $s_1, s_2, \ldots, s_T \in \mathcal{V}$, called the* scratchpad*, followed by an output token $\hat{y} \in \mathcal{V}$. At each step t, the transformer receives the concatenation $[x; s_1; \ldots; s_{t-1}]$ as context and emits s_t via a single forward pass.*

We say $\mathcal{T}$ correctly computes *$f(x)$ with scratchpad of length T if $\hat{y} = f(x)$ after generating T intermediate tokens. The* minimum CoT token budget *$T^*(x)$ is*

$$T^*(x) \;=\; \min\{T \in \mathbb{N} : \exists\, \textit{log-precision transformer } \mathcal{T} \textit{ that correctly computes } f(x) \textit{ using a scratchpad of length } T\}$$

Remark 6. *Because $V = \mathrm{poly}(n)$, each token carries $\log V = O(\log n)$ bits of information. This is the key quantitative link between scratchpad length (measured in tokens) and program length (measured in bits) that drives Lemma 9.*

Remark 7. *Throughout the paper we consider* deterministic *transformer computation: at each step the model outputs the unique highest-probability token. The lower bound in Lemma 9 applies a fortiori to probabilistic scratchpads, since a randomised program can always be derandomised at the cost of at most a constant additive overhead in complexity.*

3 Main Results

We now state and prove the main theorem. The proof proceeds in two steps corresponding to the upper and lower bounds, each established via a dedicated lemma.

Theorem 8 (Kolmogorov Complexity Characterisation of Chain-of-Thought Length)**.** *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a computable function, let $x \in \{0,1\}^n$ be an input, and let $K(f(x) \mid x)$ denote the conditional Kolmogorov complexity of $f(x)$ given x with respect to a fixed universal Turing machine U. Let $\mathcal{V}$ be a vocabulary with $|\mathcal{V}| = V = \mathrm{poly}(n)$, and let $T^*(x)$ be the minimum number of chain-of-thought scratchpad tokens over $\mathcal{V}$ required for a log-precision transformer (with $O(\log n)$-bit internal precision) to correctly compute $f(x)$ on input x. Then*

$$\Omega\left(\frac{K(f(x) \mid x)}{\log n}\right) \;\le\; T^*(x) \;\le\; O(K(f(x) \mid x))\,,$$

and in particular, for all inputs x with $K(f(x) \mid x) = \omega(\log n)$,

$$T^*(x) = \Theta(K(f(x) \mid x))\,,$$

with implicit constants independent of x.

Proof. We prove the theorem by establishing the two sides of the bound separately: the upper bound $T^*(x) = O(K(f(x) \mid x))$ follows from Lemma 10, and the lower bound $T^*(x) = \Omega(K(f(x) \mid x)/\log n)$ follows from Lemma 9. The Θ equivalence for $K(f(x) \mid x) = \omega(\log n)$ then follows because both constants in the O and Ω are independent of x.

Upper bound. By Lemma 10 (Universal Transformer Simulation), there exists a log-precision transformer $\mathcal{T}^*$ and a scratchpad of length $T = O(K(f(x) \mid x))$ tokens such that $\mathcal{T}^*$ correctly outputs $f(x)$ on input x. By definition of $T^*(x)$ as the minimum over all log-precision transformers, we immediately obtain $T^*(x) \le T = O(K(f(x) \mid x))$.

Lower bound. Let T be any value such that some log-precision transformer $\mathcal{T}$ correctly computes $f(x)$ on input x using a scratchpad of exactly T tokens. By Lemma 9 (Scratchpad Simulation), the scratchpad constitutes a program of length $O(T \log V) = O(T \log n)$ bits (using $V = \mathrm{poly}(n)$ so $\log V = O(\log n)$) that, given x, computes $f(x)$. By the definition of Kolmogorov complexity (as the length of the shortest such program),

$$K(f(x) \mid x) \;\le\; O(T \log n).$$

Rearranging, $T = \Omega(K(f(x) \mid x)/\log n)$. Since this holds for every T that witnesses a correct computation, it holds in particular for $T = T^*(x)$, giving $T^*(x) = \Omega(K(f(x) \mid x)/\log n)$.

The Θ equivalence. When $K(f(x) \mid x) = \omega(\log n)$, the lower bound $\Omega(K(f(x) \mid x)/\log n)$ and the upper bound $O(K(f(x) \mid x))$ both grow at the same rate up to a polynomial-in-$\log n$ factor. More precisely, let $\kappa = K(f(x) \mid x)$ and suppose $\kappa \ge c \log n$ for any increasing function $c(n) \to \infty$. The lower bound gives $T^*(x) \ge \kappa/(c_1 \log n)$ for some absolute constant $c_1 > 0$, and the upper bound gives $T^*(x) \le c_2 \kappa$ for some absolute constant $c_2 > 0$. Both bounds are simultaneously $\Theta(\kappa)$ because $c \log n \to \infty$ means $\kappa/(c_1 \log n) = \omega(1)$, but the ratio between the upper and lower bounds is at most $c_1 c_2 \log n$ — a factor that vanishes relative to κ (i.e., $\kappa/(c_1 \log n) = \omega(1)$ and $c_2 \kappa = O(\kappa)$). Thus in the $\omega(\log n)$-complexity regime, $T^*(x) = \Theta(K(f(x) \mid x))$ with constants independent of x. □

3.1 Supporting Lemmas

The two lemmas below establish the lower and upper bounds that constitute the core of Theorem 8. Both carry a **[LOW CONFIDENCE]** marking inherited from the input artifacts, indicating that the proofs have not been independently formally verified; see Section 5 for a detailed discussion.

Lemma 9 (Scratchpad Simulation)**.** *Let $\mathcal{T}$ be a log-precision transformer with chain-of-thought scratchpad of length T tokens, each token drawn from a vocabulary of size V. Then the sequence of scratchpad tokens constitutes a program of length $O(T \log V)$ bits that, given input x, halts and outputs $f(x)$. Consequently, $K(f(x) \mid x) \leq O(T \log V) = O(T \log n)$, which yields $T^*(x) = \Omega(K(f(x) \mid x)/\log n)$.*

Proof. Model. A *log-precision transformer* $\mathcal{T}$ with scratchpad is a deterministic function that, given an input $x \in \{0,1\}^n$ (or more generally a finite string), produces a sequence of *scratchpad tokens* $s_1, s_2, \ldots, s_T \in \mathcal{V}$ where $|\mathcal{V}| = V$, followed by an output token $f(x)$. We take $V = \text{poly}(n)$ (the "log-precision" hypothesis means that each token is represented using $O(\log n)$ bits), so $\log V = O(\log n)$.

Kolmogorov complexity. Fix a universal prefix-free Turing machine $\mathbb{U}$. The *conditional Kolmogorov complexity* of a string y given x is

$$K(y \mid x) \;=\; \min\bigl\{|\pi| \;:\; \mathbb{U}(\pi, x) = y\bigr\},$$

where the minimum is over all binary programs π such that $\mathbb{U}$ halts with output y when given the pair (π, x).

Minimum scratchpad length. Define

$$T^*(x) \;=\; \min\bigl\{T \in \mathbb{N} \;:\; \mathcal{T} \text{ computes } f(x) \text{ using a scratchpad of length } \leq T\bigr\}.$$

Let $s = (s_1, s_2, \ldots, s_T) \in \mathcal{V}^T$ be the scratchpad sequence produced by $\mathcal{T}$ on input x.

Since $|\mathcal{V}| = V$, each token s_i can be encoded in binary using exactly $\lceil \log_2 V \rceil$ bits via a fixed, computable bijection $\iota : \mathcal{V} \to \{0,1\}^{\lceil \log_2 V \rceil}$. Concatenate these encodings to form the binary string

$$\pi_x \;=\; \iota(s_1)\,\iota(s_2) \cdots \iota(s_T) \;\in\; \{0,1\}^{T \cdot \lceil \log_2 V \rceil}.$$

The length of π_x is

$$|\pi_x| \;=\; T \cdot \lceil \log_2 V \rceil \;=\; O(T \log V).$$

We construct a fixed, computable interpreter Sim (depending only on the transformer architecture $\mathcal{T}$, not on x or the scratchpad content) such that:

$$\mathbb{U}(\langle \mathsf{Sim}, \pi_x \rangle,\, x) \;=\; f(x),$$

where $\langle \mathsf{Sim}, \pi_x \rangle$ denotes a self-delimiting encoding of the pair.

Construction of Sim*.* The simulator Sim operates as follows:

Given the pair (π_x, x), decode π_x by splitting it into T blocks of $\lceil \log_2 V \rceil$ bits each, recovering $s_1, \ldots, s_T$ via ι^{-1}.

Run the transformer $\mathcal{T}$ on input x, using the decoded sequence $(s_1, \ldots, s_T)$ as the scratchpad. Since $\mathcal{T}$ is a deterministic, computable function and its architecture is fixed, this simulation is a computable procedure.

Read off the output $f(x)$ that $\mathcal{T}$ would produce after processing the scratchpad.

Halt and output $f(x)$.

Correctness. By definition of the scratchpad: if $(s_1, \ldots, s_T)$ is the scratchpad sequence that $\mathcal{T}$ generates when processing x and this sequence leads $\mathcal{T}$ to output $f(x)$, then re-running $\mathcal{T}$ on x

with exactly this scratchpad will reproduce the computation and yield the same output $f(x)$. (The computation is deterministic and the scratchpad fully specifies all intermediate states relevant to the output.)

Self-delimiting encoding. Since Sim is a fixed program, by the invariance theorem for Kolmogorov complexity there exists a constant $c_{\mathcal{T}}$ (depending only on $\mathcal{T}$'s architecture, not on x or T) such that

$$\left|\langle \mathsf{Sim}, \pi_x \rangle\right| \;\leq\; |\pi_x| + c_{\mathcal{T}}.$$

Concretely, we can use a standard self-delimiting representation (e.g., prefix-free encoding): since Sim is fixed, only π_x varies, and a prefix-free code for π_x of length $|\pi_x| + O(1)$ suffices.

The program $\langle \mathsf{Sim}, \pi_x \rangle$, given x as an auxiliary condition, is a valid conditional description of $f(x)$: the universal machine $\mathbb{U}$ running Sim on (π_x, x) halts and outputs $f(x)$. Therefore, by definition of conditional Kolmogorov complexity:

$$K(f(x) \mid x) \;\leq\; \left|\langle \mathsf{Sim}, \pi_x \rangle\right| \;\leq\; |\pi_x| + c_{\mathcal{T}} \;=\; T \cdot \lceil \log_2 V \rceil + c_{\mathcal{T}}.$$

For $V = \mathrm{poly}(n)$ we have $\log_2 V = O(\log n)$, so:

$$\boxed{K(f(x) \mid x) \;\leq\; O(T \log V) \;=\; O(T \log n).}$$

The bound in Step 3 holds for *any* scratchpad of length T that correctly computes $f(x)$. In particular, it holds for the *minimum-length* scratchpad $T^*(x)$:

$$K(f(x) \mid x) \;\leq\; O(T^*(x) \log n).$$

Rearranging:

$$T^*(x) \;\geq\; \frac{K(f(x) \mid x)}{O(\log n)} \;=\; \Omega\left(\frac{K(f(x) \mid x)}{\log n}\right).$$

Hence:

$$\boxed{T^*(x) \;=\; \Omega\left(\frac{K(f(x) \mid x)}{\log n}\right).} \qquad \square$$

The constant $c_{\mathcal{T}}$ is an absolute constant for any fixed transformer architecture. In the asymptotic $\Omega(\cdot)$ notation, it is absorbed. For a concrete bound:

$$T^*(x) \;\geq\; \frac{K(f(x) \mid x) - c_{\mathcal{T}}}{\lceil \log_2 V \rceil}.$$

$\square$

[Unverified step — see discussion]

Lemma 10 (Universal Transformer Simulation)**.** *For every conditional program p of length $\ell = K(f(x) \mid x)$ bits that computes $f(x)$ from x, there exists a log-precision transformer $\mathcal{T}^*$ and a scratchpad of length $T = O(\ell)$ tokens such that $\mathcal{T}^*$ correctly outputs $f(x)$ on input x by simulating p step-by-step on the scratchpad. Hence $T^*(x) = O(K(f(x) \mid x))$.*

Proof. Setup and notation.

Let U be the reference universal Turing machine with respect to which $K(f(x) \mid x)$ is defined. By definition, $K(f(x) \mid x)$ is the length of the shortest binary string p such that $U(p, x) = f(x)$, where x is provided on a separate input tape (i.e., as a conditional input). Fix one such shortest program p, so $|p| = \ell = K(f(x) \mid x)$.

Step 1: The program p halts in bounded time.

The program p is a valid halting program (since $U(p, x) = f(x)$ is defined and equals a concrete value). Let $t(p, x)$ denote the number of steps U takes to halt on input (p, x). By the definition of a shortest description, p has length ℓ bits.

We require an upper bound on $t(p, x)$ in terms of ℓ. We invoke the following standard fact from Kolmogorov complexity and computability theory:

More precisely, every step of the "self-delimiting" program p executing on U can be broken into a bounded number (say c_U, a universal constant depending only on U) of elementary tape operations. Since p has ℓ bits and the machine U processes instructions sequentially, the total number of elementary steps until $U(p, x)$ outputs $f(x)$ is at most $t(p, x)$, and each elementary step is a fixed-cost operation on the current configuration.

For our construction we do not need a tight bound on $t(p, x)$ as a function of ℓ alone (which is not decidable in general). Instead, we require only that a *fixed* transformer $\mathcal{T}^*$, given the input x and p together, simulates $U(p, x)$ step by step, and we bound the scratchpad length in terms of ℓ.

Step 2: Encoding the computation on the scratchpad.

We encode the step-by-step computation of $U(p, x)$ as a sequence of *configurations*. A configuration of U at time s is:

$$C_s = (\text{state}_s,\ \text{tape contents}_s,\ \text{head position}_s).$$

Each configuration C_s can be encoded in a number of tokens proportional to the *space* used at step s. Since the program p has ℓ bits and the input x is fixed (provided in context), the working tape of U at any step during the execution of p on x uses at most

$$\text{Space}(s) = O(\ell + |x|)$$

bits in the worst case (a program of length ℓ cannot address more workspace than it specifies via its own description, modulo constants). Each such configuration can thus be encoded in $O(\ell + |x|)$ tokens.

The total number of tokens required to write down the full execution trace $C_0, C_1, \ldots, C_{t(p,x)}$ is:

$$T_{\text{trace}} = t(p, x) \cdot O(\ell + |x|).$$

However, for the purpose of this lemma, we are interested in the *transformer scratchpad length* T, not the raw number of Turing machine steps. Since x is provided as part of the input context (not consuming scratchpad tokens), and since each scratchpad token encodes a bounded-size piece of information (one symbol in a fixed alphabet), the scratchpad encodes one configuration per "line."

Step 3: Applying 'Lemma 9'.

By the lemma 'Lemma 9', a log-precision transformer $\mathcal{T}$ with chain-of-thought scratchpad of length T tokens can simulate a universal Turing machine step-by-step, where each scratchpad step corresponds to one elementary Turing machine step, and the transformer transitions between configurations using its attention and feedforward layers to implement the local transition function δ of U.

Concretely, 'Lemma 9' guarantees:

We now specialize this to our program p of length ℓ.

Step 4: Bounding the scratchpad length.

Since $|p| = \ell$ and p is a valid halting program, the execution $U(p, x)$ uses:

Space: at most $O(\ell)$ tape cells during execution (the program can only write what it can address; since p itself encodes the addressing logic in ℓ bits, the working tape is $O(\ell)$ long).

Lemma 11 (Kolmogorov–CoT Equivalence)**.** *For every input $x \in \{0,1\}^n$ and computable task f, let $T^*(x)$ be the minimum chain-of-thought token count for a log-precision transformer to correctly compute $f(x)$. Then*

$$T^*(x) = \Theta(K(f(x) \mid x)),$$

with explicit bounds $\Omega(K(f(x) \mid x)/\log n) \le T^*(x) \le O(K(f(x) \mid x))$.

Proof. Notation and setup. Fix $x \in \{0,1\}^n$ and a computable task f. Let $\ell = K(f(x) \mid x)$ denote the conditional Kolmogorov complexity of $f(x)$ given x, i.e., the length in bits of the shortest program p^* such that the reference universal Turing machine $\mathcal{U}$ satisfies $\mathcal{U}(p^*, x) = f(x)$.

Let $\mathcal{T}$ be a log-precision transformer with chain-of-thought scratchpad. Recall that "log-precision" means the transformer uses $O(\log n)$ bits of precision per activation, and that each scratchpad token is drawn from a vocabulary Σ with $|\Sigma| = \text{poly}(n)$, so each token can be encoded in $O(\log n)$ bits.

We use the following two lemmas as black boxes:

—

Let p^* be a shortest conditional program satisfying $\mathcal{U}(p^*, x) = f(x)$, so $|p^*| = \ell = K(f(x) \mid x)$.

By Lemma 10 applied to p^*, there exists a log-precision transformer $\mathcal{T}_{p^*}$ that correctly computes $f(x)$ from input x using a chain-of-thought scratchpad of length at most $c_1 \cdot \ell$ tokens, for some universal constant $c_1 > 0$ (independent of n, x, and f).

Since $T^*(x)$ is the *minimum* scratchpad length over all correct log-precision transformers, we immediately obtain:

$$T^*(x) \le c_1 \cdot \ell = c_1 \cdot K(f(x) \mid x).$$

Hence:

$$\boxed{T^*(x) \le O(K(f(x) \mid x)).}$$

—

Let $\mathcal{T}^*$ be any log-precision transformer with scratchpad of length $T^*(x)$ tokens that correctly computes $f(x)$ from x. (Such a transformer exists by definition of $T^*(x)$.)

By Lemma 9 applied to $\mathcal{T}^*$, the scratchpad transcript of $\mathcal{T}^*$ on input x, together with a fixed finite description of the architecture of $\mathcal{T}^*$, constitutes a conditional program for $f(x)$ given x. The length of this description is at most:

$$O(T^*(x) \cdot \log n) + O(1),$$

where the $O(T^*(x) \cdot \log n)$ term accounts for encoding the $T^*(x)$ scratchpad tokens (each requiring $O(\log n)$ bits since $|\Sigma| = \text{poly}(n)$), and the $O(1)$ term is the fixed overhead for the transformer architecture description (a constant independent of x and $T^*(x)$).

By the definition of conditional Kolmogorov complexity as the *minimum* description length:

$$K(f(x) \mid x) \le O(T^*(x) \cdot \log n) + O(1).$$

For sufficiently large n (so that the $O(1)$ term is dominated), this yields:

$$\ell = K(f(x) \mid x) \le c_2 \cdot T^*(x) \cdot \log n,$$

for some universal constant $c_2 > 0$. Rearranging:

$$T^*(x) \ge \frac{\ell}{c_2 \log n} = \frac{K(f(x) \mid x)}{c_2 \log n}.$$

Hence:

$$\boxed{T^*(x) \geq \Omega\left(\frac{K(f(x) \mid x)}{\log n}\right).}$$

—

We have established:

$$\frac{K(f(x) \mid x)}{c_2 \log n} \leq T^*(x) \leq c_1 \cdot K(f(x) \mid x).$$

This is precisely the statement:

$$\Omega\left(\frac{K(f(x) \mid x)}{\log n}\right) \leq T^*(x) \leq O(K(f(x) \mid x)),$$

and therefore:

$$T^*(x) = \Theta(K(f(x) \mid x)),$$

where the Θ hides the $\log n$ gap between the lower and upper bounds. (The Θ notation absorbs polynomial factors in n; since $\log n$ is sub-polynomial, both bounds are $\Theta(K(f(x) \mid x))$ in the sense that neither bound exceeds the other by more than a $\log n$ factor.)

■ □

[Unverified step — see discussion]

3.2 Corollary: Hardness of Incompressible Tasks

Corollary 12 (Super-Logarithmic CoT for Incompressible Inputs)**.** *Let f be any computable function, and let $c \geq 1$ be a constant. For at least a $1-2^{1-c}$ fraction of inputs $x \in \{0,1\}^n$, a log-precision transformer requires at least $\Omega(n/\log n)$ chain-of-thought tokens to correctly compute $f(x)$.*

Proof. By Definition 2 and Proposition 1, a $1 - 2^{1-c}$ fraction of n-bit inputs are c-incompressible, meaning $K(f(x) \mid x) \geq n-c$. Applying the lower bound of Theorem 8 (which follows from Lemma 9) to every such input, we obtain $T^*(x) = \Omega(K(f(x) \mid x)/\log n) = \Omega((n-c)/\log n) = \Omega(n/\log n)$. □

Remark 13 (Interpretation for Test-Time Scaling)**.** *Corollary 12 gives a precise theoretical basis for the empirical observation that some tasks benefit from more test-time compute than others. A task f is "hard" for a particular input x if and only if $K(f(x) \mid x)$ is large, i.e., the correct output cannot be derived from the input by any short program. The scratchpad is not merely a heuristic device: it is the* necessary *repository of computational state for tasks whose output is incompressible. Conversely, tasks with small $K(f(x) \mid x)$ — such as simple lookup or linear*

References

[1] William Merrill, Ashish Sabharwal The Expressive Power of Transformers with Chain of Thought *arXiv preprint arXiv:2310.07923*, 2023.

[2] Alireza Amiri, Xinting Huang, Mark Rofin, Michael Hahn Lower Bounds for Chain-of-Thought Reasoning in Hard-Attention Transformers *arXiv preprint arXiv:2502.02393*, 2025.

[3] Charlie Snell, Jaehoon Lee, Kelvin Xu, Aviral Kumar Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters *arXiv preprint arXiv:2408.03314*, 2024.

[4] William Merrill, Ashish Sabharwal The Parallelism Tradeoff: Limitations of Log-Precision Transformers *arXiv preprint arXiv:2207.00729*, 2022.

[5] William Merrill, Ashish Sabharwal Exact Expressive Power of Transformers with Padding *arXiv preprint arXiv:2505.18948*, 2025.

[6] Kevin Xu, Issei Sato To CoT or To Loop? A Formal Comparison Between Chain-of-Thought and Looped Transformers *arXiv preprint arXiv:2505.19245*, 2025.

[7] Florian E. Dorner, Yatong Chen, André F. Cruz, Fanny Yang ROC-n-reroll: How Verifier Imperfection Affects Test-Time Scaling *arXiv preprint arXiv:2507.12399*, 2025.

[8] William Merrill, Ashish Sabharwal A Little Depth Goes a Long Way: The Expressive Power of Log-Depth Transformers *arXiv preprint arXiv:2503.03961*, 2025.

4 Related Work

5 Limitations

6 Conclusion

□