ON THE TURING COMPLETENESS ANALYSIS OF TRANSFORMERS AND AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers have emerged as the dominant architecture in sequence modeling, achieving remarkable success in natural language processing and reasoning tasks. While existing literature has established the Turing completeness of transformers under bounded input length, the reasoning power of a single transformer operating on inputs of unbounded length is not fully explored. In this paper, we theoretically investigate the reasoning limitations of a single transformer and the enhanced capabilities of agent systems. We show that a single fixed finite precision transformer cannot memorize certain Turing machines with inputs of arbitrary length, such as the arithmetic; and a single fixed infinite precision transformer trained with a random algorithm is not Turing complete with probability one under certain conditions. To overcome the limitation of a single transformer, we define a formal agent architecture consisting of decision, execution, and memory modules and show that for any Turing machine $\mathbb{T}$, there exists an agent that can memorize $\mathbb{T}$ and is computationally the same as $\mathbb{T}$. Thus, agents are Turing complete.

1 INTRODUCTION

Transformers are the central architecture in modern sequence modeling and have achieved remarkable practical success in natural language processing, code generation, and mathematical reasoning. Recent theoretical studies have made substantial progress on the reasoning power of transformers. Log-precision transformers have been shown to be Turing complete (Pérez et al., 2021; Brosamle & Eckstein, 2026; Jiang et al., 2026) in a certain sense; constant size transformers have also been shown to be Turing complete (Li & Wang, 2025). Moreover, log-precision transformers with Chain-of-thought (CoT) are capable of solving any polynomial-time problem (Merrill & Sabharwal, 2024), and they are also able to handle the arithmetic task $\mathbf{Arith}_{p,n}$ (Feng et al., 2023). CoT has likewise been shown to improve the reasoning power of fixed-precision transformers (Li et al., 2023; Feng et al., 2025). On the negative side, it is proven that there exists a class of reasoning problems for which transformers, with and without CoT, have the same solving power (Yu et al., 2025).

As stated by Cui et al. (2026), there is a limitation shared by much of the existing literature on transformer reasoning: the conclusions of most works are limited to finite length questions or depended on the input length, such as (Pérez et al., 2021; Merrill & Sabharwal, 2024; Feng et al., 2023; Brosamle & Eckstein, 2026; Jiang et al., 2026). This means that for a given Turing machine $\mathbb{T}$ and length limitation n, they can construct a transformer $\mathcal{F}_{\mathbb{T},n}$ depending on n such that $\mathcal{F}_{\mathbb{T},n}$ can memorize $\mathbb{T}$ on any input with a length not exceeding n. In other words, $\{\mathcal{F}_{\mathbb{T},n}\}_{n=1}^{\infty}$ is Turing complete for simulating $\mathbb{T}$, but not for a single transformer. However, people do not usually use a family of transformers to solve problems in practice. Mudarisov et al. (2025) showed why transformers struggle to process long sequences, indicating that as the input length grows, the attention weights tend to become nearly uniform.

In order to understand the reasoning power of the transformer on inputs of arbitrary length, we have refined the relevant theories by answering the following question.

Question 1. Is it possible for a single fixed transformer with CoT to memorize a Turing machine for inputs of arbitrary length?

For a Turing machine $\mathbb{T}$, we say that a transformer $\mathcal{F}$ memorizes $\mathbb{T}$ if, for any stoppable input x, it holds $\widehat{\mathcal{F}}_{\text{out}}(x) = \mathbb{T}(x)$. In this paper, we consider the case of unbounded length inputs for Turing

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

machines, and we show that under reasonable conditions, a single transformer cannot achieve Turing completeness even with CoT. For finite precision situations, we employ an explicit constructive approach to demonstrate Turing incompleteness. For infinite precision situations, we provide a tight sensitivity bound on how much a transformer's output can change for two inputs that differ at only a single position (Theorems 4.2 and 4.4), which is then used to obtain their Turing incompleteness. Overall, we obtain the following conclusions (Refer to Theorems 4.1, 4.5, and 4.6):

Theorem 1.1 (Informal)**.** *(1) A single finite precision transformer is not Turing complete. (2) A single infinite precision transformer is not Turing complete under positive confidence. (3) A single infinite precision transformer trained with a random algorithm is not Turing complete with probability one under certain conditions including CoTs.*

Theorem 1.1 shows that single transformer with CoT cannot memorize certain Turing machines on unbounded length inputs. On the other hand, agents based on transformer are widely used to solve complex tasks, such as mathematical reasoning. Through the combination of LLMs, skills and tools, the capabilities of agents can be significantly improved. In light of the negative results in Theorem 1.1, it is interesting to examine the reasoning capabilities of LLM-based agents.

Question 2. Are LLM-based agents Turing complete?

It was shown that memory augmented LLMs are computationally universal (Schuurmans, 2023). Recently, memory augmented LLMs have been shown to be Turing complete with explicit constructions (Chen & Liu, 2026). In this paper, we focus on agents that are more realistic and show their Turing completeness. Specifically, we arrive at the following conclusions (Refer to Theorem 5.1):

Theorem 1.2 (Informal)**.** *We define an agent architecture consisting of finite precision transformers as decision and execution modules, a memory module, and linear-time computable tools, such that for any Turing machine $\mathbb{T}$, there exists an agent that can memorize $\mathbb{T}$ and is computationally the same as $\mathbb{T}$.*

Theorem 1.2 demonstrates that combining simple tool calls with a loop execution pattern is significantly more effective than general CoT for solving long problems.

Finally, we experimentally verify our theoretical conclusions using several widely used LLMs. We should emphasize that this is primarily a theory paper, and the experiments serve as proof-of-concept demonstrations rather than comprehensive large-scale benchmarks.

In summary, the main contributions of this paper are threefold.

1. We show that a fixed finite precision transformer cannot memorize certain Turing machines, such as arithmetic, and thus is not Turing complete. Infinite precision transformers are also not Turing complete with positive confidence or with probability one under certain conditions.
2. We show that an agent built on finite-precision transformers can memorize Turing machines with inputs of unbounded length and thus is Turing complete.
3. We validate our theoretical results experimentally on some common LLMs and agents.

2 Related Work

Expressive power and Turing completeness. A substantial line of work studies transformer expressiveness through Turing-machine simulation and circuit complexity. Early results established Turing completeness for encoder–decoder transformers and analyzed positional encodings, masking, and architectural components (Pérez et al., 2021; Bhattamishra et al., 2020; Brosamle & Eckstein, 2026). Later work extended these results to decoder-only and CoT transformers (Roberts, 2023; Merrill & Sabharwal, 2024), while recent constructions establish Turing completeness for constant-bit-size and softmax CoT transformers under suitable assumptions (Li & Wang, 2025; 2026; Jiang et al., 2026). Circuit-complexity results further characterize the roles of numerical precision, depth, and intermediate reasoning steps (Merrill et al., 2022; Merrill & Sabharwal, 2023; Chiang, 2025; Merrill & Sabharwal, 2025; Li et al., 2023). Other work studies limitations of attention normalization, including selective capacity and token separation (Mudarisov et al., 2025). Length generalization has also been widely studied, focusing on whether capabilities learned on bounded-length

inputs extend to longer sequences (Kazemnejad et al., 2023; Zhou et al., 2024; Wang et al., 2024; Huang et al., 2025; Chen et al., 2026; Yang et al., 2026).

Agents and external computation. LLM-based agents combine repeated model invocation with external state, context management, and tools (Yao et al., 2023; Schick et al., 2023). External memory and context-management mechanisms provide additional computational resources (Chen & Liu, 2026), and theoretical work studies how such mechanisms affect the computational power of systems built around a fixed language model (Schuurmans, 2023; Cui et al., 2026). A detailed comparison with (Chen & Liu, 2026) is given in Remark 5.1.

3 Notation, Model, and Basic Assumptions

In this section, we provide the basic notations. More details can be found in Appendix A.

3.1 Autoregressive Transformer

Let $\Sigma = \{\sigma_i\}_{i=1}^T$ be the vocabulary or alphabet set, and let Σ^* denote the set of all finite sequences over Σ. The symbol σ_0 is used only as an end-of-output token and does not appear in intermediate positions of an input sequence. For any sequence $x \in \Sigma^*$, let $\mathrm{len}(x)$ denote its length, let $x[i]$ denote its i-th symbol, and let $\mathrm{last}(x) = x[\mathrm{len}(x)]$ denote its last symbol. For any two sequences $x_1, x_2 \in \Sigma^*$, write $x_1 \oplus x_2$ for their concatenation. If $S \subset \Sigma^*$ is finite, write $\mathrm{len}(S) = \max_{x\in S} \mathrm{len}(x)$.

This article considers a transformer $\mathcal{F} = E_{\mathrm{out}} \circ \mathcal{F}_L \circ \mathcal{F}_{L-1} \circ \cdots \circ \mathcal{F}_1 \circ E_{\mathrm{emb}} : \Sigma^* \to \mathbb{R}^{T+1}$, where E_{emb} is the embedding layer, E_{out} is the output layer, $\mathcal{F}_l$ is the hidden layer with the RoPE position embedding R, position mask M, and multi-head attention, as follows:

$$\mathcal{F}_{\ell,1}(x) = \sum_{h=1}^{H} \mathrm{softmax}(R(xQ_{\ell,h}, K_{\ell,h}x^\top) + M)\, xV_{\ell,h},$$
$$\mathcal{F}_\ell(x) = x + \mathcal{F}_{\ell,1}(x) + \mathrm{FNN}(x + \mathcal{F}_{\ell,1}(x)).$$

This is a commonly used transformer architecture. We focus on token-by-token autoregressive generation as the output mode, and we denote the transformer's output for an input x by $\widehat{\mathcal{F}}(x)$.

3.2 Memorization of Turing machine by Transformer

In this section, we define how a transformer could memorize a Turing machine.

We represent a Turing machine as $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$, where Q is the state set, Σ_1 is the set of symbols that can be input, Σ_2 is the set of symbols that can appear in the calculation process and output, and δ is the transfer function. The empty-symbol B is in $\Sigma_2 \setminus \Sigma_1$. For a Turing machine $\mathbb{T}$, we call an input $x \in \Sigma_1^*$ *stoppable* for $\mathbb{T}$ if $\mathbb{T}(x)$ halts after a finite number of steps. For any stoppable input x, we denote $t_{\mathbb{T}}(x)$ to be the number of steps that $\mathbb{T}$ performs on input x before halting, $t_{\mathbb{T}}(x)$ is also referred to as the *running time* of the Turing machine $\mathbb{T}$ for solving x.

To guarantee that a transformer can reproduce the entire output of a Turing machine, we require its alphabet Σ to contain a special symbol, denoted by σ_T. The symbol σ_T serves as a delimiter that separates the CoT from the final output. For any generated sequence $\widehat{\mathcal{F}}(x)$ containing σ_T, we use the last occurrence of σ_T as the delimiter and write $\widehat{\mathcal{F}}(x) = \mathrm{CoT}(\mathcal{F}(x)) \oplus \sigma_T \oplus \widehat{\mathcal{F}}_{\mathrm{out}}(x)$, where $\mathrm{CoT}(\mathcal{F}(x))$ is the prefix preceding the last occurrence of σ_T, and $\widehat{\mathcal{F}}_{\mathrm{out}}(x)$ is the suffix following it. Denote $\Sigma_0 = \Sigma \setminus \{\sigma_T\}$.

Definition 3.1. *We say that a transformer $\mathcal{F}$ memorizes a Turing machine $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$ with a CoT if $\Sigma_2 \subset \Sigma_0$ and for every stoppable input $x \in \Sigma_1^*$ for $\mathbb{T}$, $\widehat{\mathcal{F}}(x)$ contains σ_T and $\widehat{\mathcal{F}}_{\mathrm{out}}(x) = \mathbb{T}(x)$. The CoT length is defined as $C(x) = |\,\mathrm{CoT}(\mathcal{F}(x))| = |\widehat{\mathcal{F}}(x)| - |\widehat{\mathcal{F}}_{\mathrm{out}}(x)| - 1$.*

For simplicity, when we later say the transformer memorizes a Turing machine, we mean it does so using CoT and $\Sigma_2 \subset \Sigma_0$. Let us use an example to illustrate these notions.

Example 3.1. *Let $\Sigma_1 = \{0,1,2,3,4,5,6,7,8,9,+,-,*,/,(,),|\}$, $\Sigma_2 = \Sigma_1 \cup \{B\}$, and $\mathbb{T}_{\mathbf{Arith}} = (Q, \Sigma_1, \Sigma_2, \delta)$ be the Turing machine for solving the arithmetic of rational numbers, with $n|m$ to*

represent $\frac{n}{m}$. *For instance, if* $x = 2 + 2 * (5 - 1) + 3$, *then* $\mathbb{T}_{\mathbf{Arith}}(x) = 13$. *Let* $\Sigma = \Sigma_2 \cup \{=\}$ *and* σ_T *be '='. For a transformer* $\mathcal{F}$, *if* $\widehat{\mathcal{F}}(x)$ *is '*$= 2 + 2 * 4 + 3 = 2 + 8 + 3 = 10 + 3 = 13$*' with final output* σ_0, *then* $\widehat{\mathcal{F}}_{\text{out}}(x)$ *is* 13, *which is the sequence between the last* $=$ *and* σ_0.

4 Limitations of Transformer in Memorizing Turing machine

Prior studies have demonstrated that a CoT transformer is capable of memorizing a Turing machine for inputs with *bounded length* (Pérez et al., 2021; Merrill & Sabharwal, 2024; Yu et al., 2025; Brosamle & Eckstein, 2026), as detailed in the Appendix B. In this section, we show that a single transformer cannot memorize most Turing machines for input with *unbounded length*.

We consider finite-precision and infinite-precision transformers separately. For finite-precision transformers, we prove the result via an explicit construction. For infinite-precision transformers, we prove the result by deriving a tight bound on value propagation.

4.1 A Finite-precision Transformer is not Turing Complete

In this section, we demonstrate that a single fixed transformer with finite precision is unable to solve simple algorithmic tasks such as $\mathbf{Arith}$, which admit inputs of unbounded length. Therefore, such a model is not Turing complete. Firstly, we define the transformer with finite precision.

Definition 4.1. *By saying that a transformer* $\mathcal{F}$ *has* q*-precision, we mean that all its parameters, as well as the matrix calculation, Softmax calculation, and the position embedding calculation in the hidden-layer output, are represented with* q *decimal real numbers. We denote a transformer with* q*-precision by* $\mathcal{F}_q$. *Unless otherwise specified, it is assumed that* $\mathcal{F}$ *has unlimited precision in this paper. More details about the finite precision transformer can be found in Appendix C.1.*

Given any sequence x and $n \in \mathbb{N}_+$, we define x^n to be the concatenation of n copies of x. Below, we define a class of Turing machines that cannot be memorized by fixed-precision transformers.

Definition 4.2. *A Turing machine* $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$ *is called* non-converging *if there exist* $x, y, z \in \Sigma_1^*$ *such that every* $x \oplus y^n \oplus z$, $n \in \mathbb{N}_+$, *is a stoppable input for* $\mathbb{T}$, *and* $\mathbb{T}(x \oplus y^n \oplus z) \neq \mathbb{T}(x \oplus y^m \oplus z)$ *for any distinct* $m, n \in \mathbb{Z}_+$.

Based on this definition, we establish the following result about the Turing incompleteness for finite precision transformers:

Theorem 4.1. *Let* $\mathbb{T}$ *be a non-converging Turing machine. Then, for any* $q \in \mathbb{N}_+$ *and any fixed precision transformer* $\mathcal{F}_q$, $\mathcal{F}_q$ *cannot memorize* $\mathbb{T}$. *Consequently, a fixed-precision transformer cannot be Turing complete.*

Proof. In processing long inputs, a Turing machine is limited only by memory capacity. In contrast, Transformers must also contend with precision limits. This is particularly evident in position embeddings, where precision loss in trigonometric calculations with sequence length leads to cyclic phenomena in RoPE encoding. Additionally, the attention mechanism is susceptible to numerical instability; the exponential operations can produce values so minute that they underflow to zero, effectively truncating the computation. The complete proof is in Appendix C. □

In practice, a large portion of practical Turing machines with infinite output image is non-converging. For instance, $\mathbb{T}_{\mathbf{Arith}}$ is obviously non-converging, hence any finite precision transformer $\mathcal{F}_q$ cannot memorize $\mathbb{T}_{\mathbf{Arith}}$.

4.2 Infinite-precision Transformer is not Turing Complete under some conditions

In this section, we discuss infinite precision transformers, which are more powerful than finite precision transformers. We first present a key fact in Section 4.2.1 and use that fact to obtain the main result about Turing incompleteness in Sections 4.2.2 and 4.2.3. The application scopes of the theorems in this paper are clarified in Appendix H.

4.2.1 A TIGHT BOUND OF VALUE PROPAGATION FOR TWO DISTINCT TOKENS

In this section, we derive a tight bound on the transformer outputs for the embeddings of two tokens, which will be used to prove the Turing incompleteness of infinite-precision transformers in the subsequent section. We first define a local sensitivity measure for transformers, which quantifies the effect of modifying a single input token at a non-output position on the final representation.

Definition 4.3. *Let $\mathbb{B}(x,k)$ be the set of x' such that $\text{len}(x) = \text{len}(x') = n$ and $x[j] \neq x'[j]$ for no more than k values of $j \in [n-1]$, and $x[n] = x'[n]$. That is, x' differs from x by no more than k non-output positions. For a sequence x and a transformer $\mathcal{F}$, the* sensitivity *of $\mathcal{F}$ with respect to x is defined as*

$$\text{sen}(\mathcal{F}, x) = \max_{x' \in \mathbb{B}(x,1)} ||\mathcal{F}(x) - \mathcal{F}(x')||_2.$$

Now, we present the following key technical result.

Theorem 4.2. *For any input x and any transformer $\mathcal{F}$ previously defined with depth L, we have*

$$\text{sen}(\mathcal{F}, x) \leq C_{\mathcal{F}} \frac{\ln^{L-1}(\text{len}(x))}{\text{len}(x)},$$

where $C_{\mathcal{F}}$ is a constant that depends only on $\mathcal{F}$.

Proof. Consider two inputs differing only at position i. Since the hidden states and attention logits of a fixed transformer are uniformly bounded, each causal attention weight at position j is at most $O(1/j)$. Hence, a local perturbation is diluted to $O(1/j)$ after the first layer. Inductively, if the perturbation at layer $\ell-1$ is $O((\ln j)^{\ell-2}/j)$, then the attention aggregation contributes one additional logarithmic factor. Using the Lipschitz continuity of the remaining components and taking $j = \text{len}(x)$ yields $\text{sen}(F, x) \leq C_F \frac{(\ln \text{len}(x))^{L-1}}{\text{len}(x)}$. The complete proof is in Appendix D. □

By the triangle inequality, Theorem 4.2 can be extended to $\mathbb{B}(x, K)$.

Corollary 4.3. *There exists a constant $C_{\mathcal{F}} > 0$, depending only on $\mathcal{F}$, such that for every input x, every $K \in \mathbb{N}_+$, and every $x' \in \mathbb{B}(x, K)$,*

$$\|\mathcal{F}(x) - \mathcal{F}(x')\| \leq K C_{\mathcal{F}} \frac{\ln^{L-1}(\text{len}(x))}{\text{len}(x)}.$$

The bound in Theorem 4.2 is tight up to constant factors, as shown below.

Theorem 4.4. *For any L, there exists a transformer $\mathcal{F}$ of depth L and a constant $c_{\mathcal{F}} > 0$ such that, for any $M \in \mathbb{N}_+$, there is an input x with $\text{len}(x) \geq M$ satisfying*

$$\text{sen}(\mathcal{F}, x) \geq c_{\mathcal{F}} \frac{\ln^{L-1}(\text{len}(x))}{\text{len}(x)}.$$

Proof. We construct a fixed transformer with uniform causal attention so that each layer averages the perturbation over all preceding positions. Starting from a perturbation at the first position, this repeated averaging accumulates one logarithmic factor per layer, and after L layers, the perturbation at the final position matches the upper bound in Theorem 4.2 up to a constant factor. Refer to Appendix D.3 for the whole proof. □

4.2.2 INFINITE-PRECISION TRANSFORMER IS NOT TURING COMPLETE UNDER CONFIDENCE

In this section, we show that if the output of the transformer is required to be under a certain threshold, then even infinite-precision transformers are not Turing complete. Firstly, we define that:

Definition 4.4. *$\mathcal{F}(x)$ is said to output y with* confidence *$\epsilon > 0$, if $y = \widehat{\mathcal{F}}(x)$, and*

$$\mathcal{F}_{m_y^j}(x \oplus y[1:j-1]) - \max_{k \neq m_y^j} F_k(x \oplus y[1:j-1]) > \epsilon$$

for any $j \in [\text{len}(y)]$, where $m_y^j = \arg\max_{k \in [T+1]} \mathcal{F}_k(x \oplus y[1:j-1])$.

We say that a transformer $\mathcal{F}$ can memorize a Turing machine $\mathbb{T}$ with confidence c if $\widehat{\mathcal{F}}_{\text{out}}(x) = \mathbb{T}(x)$ and every step in the complete generation with confidence c for any stoppable input x of $\mathbb{T}$.

We define a class of Turing machines that cannot be memorized by unlimited precision transformers with positive confidence.

Definition 4.5. *An input $x \in \Sigma_1^*$ is called (C, α)-sensitive if $x \in \Sigma_1^*$ is stoppable and there is an $x_1 \in \Sigma_1^* \cap \mathbb{B}(x, \lceil C \operatorname{len}(x)^\alpha \rceil)$ that is a stoppable input such that $\mathbb{T}(x) \neq \mathbb{T}(x_1)$.*

A Turing machine $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$ is said to be non-tame *if there exist constants $C > 0$ and $0 \leq \alpha < 1$ such that $\mathbb{T}$ has infinitely many (C, α)-sensitive inputs.*

Based on this kind of Turing machine, we establish the following result about Turing incompleteness for infinite precision transformers:

Theorem 4.5. *A non-tame Turing machine $\mathbb{T}$ cannot be memorized by any fixed transformer $\mathcal{F}$ with any given confidence $c > 0$.*

Proof. Corollary 4.3 implies that, for any $\epsilon > 0$, there exists N such that for all x for which $\operatorname{len}(x) > N$, $\operatorname{sen}(\mathcal{F}, x) < \epsilon$. In other words, once the sequence is sufficiently long, changing tokens at some non-output positions has only an infinitesimal influence on the final output position. The complete proof is in the Appendix E □

Most practically relevant Turing machines are non-tame and, therefore, by Theorem 4.5, cannot be memorized by a single transformer with any fixed confidence. In particular, $\mathbb{T}_{\mathbf{Arith}}$ is non-tame since it contains infinitely many $(1, 0)$-sensitive inputs.

Remark 4.1. *The incompleteness with confidence result carries practical implications. During the actual inference, the outputs of LLMs are probabilistic. To control generation diversity, sampling strategies such as Top-K or Top-p are typically employed. When the input is very long, the transformer produces low-confidence results, which can prevent it from generating a correct final answer with a high probability. This observation will be theoretically validated in the subsequent section.*

4.2.3 Memorization of Turing machine Fails Almost Everywhere

The confidence-based result is sensible for a transformer using probabilistic sampling, but it does not hold for a transformer using greedy sampling. Therefore, in this section we drop the confidence-margin assumption. The trade-off is that the conclusion becomes measure-theoretic: the probability that a trained transformer is Turing complete is zero.

First, we define a distribution satisfied by the transformer.

Definition 4.6. *For the fixed vocabulary set Σ, dataset $S \subset \Sigma^* \times \Sigma^*$, depth L, width W, head H, and a random algorithm $\mathbf{A}$, let $\mathbf{A}(S, L, W, H, \Sigma)$ be the distribution of parameters of transformers with depth L, width W, head H, and vocabulary Σ obtained after training on S with $\mathbf{A}$.*

Then, for a Turing machine, we define the following space:

Definition 4.7. *For a $G : \Sigma^* \to 2^{\Sigma^*}$ and an $f : \mathbb{N} \to \mathbb{R}$, we say that G is an f-number of rule if, for every x, $|G(x)| \leq f(\operatorname{len}(x))$.*

Let G be a f-number of rule and $\mathbb{T}$ be a Turing machine. We say a transformer $\mathcal{F} \in M_G(\mathbb{T})$ if $\mathcal{F}$ can memorize $\mathbb{T}$ and $\operatorname{CoT}(\mathcal{F}(x)) \in G(x)$ for any stoppable input x. In other words, there exist at most $f(\operatorname{len}(x))$ ways to generate the CoT for $\mathcal{F}(x)$ when it memorizes $\mathbb{T}$.

Then, we have the following result:

Theorem 4.6. *For arbitrary S, L, W, H, Σ, assume that the following conditions are satisfied.*

(1) $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$ is a (C, α) non-tame Turing machine and $\Sigma_2 \subset \Sigma_0$.

(2) $\mathbf{A}(S, L, W, H, \Sigma)$ is an absolutely continuous distribution.

(3) G is a f-number of rules, and f satisfies $\lim_{n\to\infty} \frac{f(n)(\ln n)^{(L-1)/2}}{\sqrt{n^{1-\alpha}}} = 0$.

Then, $\Pr_{\mathcal{F}\sim\mathbf{A}(S,L,W,H,\Sigma)}(\mathcal{F} \in M_G(\mathbb{T})) = 0$. That is, if fewer than $\widetilde{o}(\sqrt{n^{1-\alpha}})$ CoTs are allowed for inputs of length n, then the probability for $\mathcal{F}$ trained by algorithm $\mathbf{A}$ to memorize $\mathbb{T}$ is zero.

Proof. Theorem 4.2 bounds the change of logits between nearby inputs but does not determine whether their predicted tokens differ. Fixing all parameters except the output bias, we consider the first disagreement of each pair of admissible generation sequences. Theorem 4.2 confines the corresponding bias difference to a thin interval. Since there are at most $f(n)^2$ such pairs, the resulting slabs have total measure tending to zero, so the memorization parameter set has probability zero. See Appendix F.2. □

The three conditions of Theorem 4.6 are reasonable. The first two conditions are clearly reasonable. For condition (3), the number of CoT could be considered the number of ways to solve the problem, to ensure the logical correctness of CoT, f isn't always very large.

In fact, even if only a portion of the inputs meet condition (3), we can still draw the same conclusion. For $\mathbb{T}_{\mathbf{Arith}}$, we have the following stronger result (the proof is given in Appendix F.3).

Corollary 4.7. *For $\mathbb{T}_{\mathbf{Arith}}$, we restrict the generation rules G to a fixed deterministic one-step calculation rule. Then, if $\mathbf{A}(S, L, W, H, \Sigma)$ is an absolutely continuous distribution, then $\Pr_{\mathcal{F}\sim\mathbf{A}(S,L,W,H,\Sigma)}(\mathcal{F} \in M_G(\mathbb{T}_{\mathbf{Arith}})) = 0$.*

5 Memorizing Turing machine by Agent

In this section, we formally define a class of agents that incorporates multiple transformers along with simple tools and show that such agents are Turing complete.

5.1 Definition of Agent

First, we introduce the agent A and how it operates.

Definition 5.1. *An agent $A = (\mathcal{F}_d, \mathcal{F}_c, \mathbf{M}, v_{\text{end}}, v_d, v_r, v_{\text{out}})$ defined over the vocabulary set Σ is specified as follows:*

(a1) $\widehat{\mathcal{F}}_d : \Sigma^* \to \Sigma^*$ is a transformer that serves as the decision module of the agent A;

(a2) $\widehat{\mathcal{F}}_c : \Sigma^* \to \Sigma^*$ is a transformer that serves as the execution module of the agent A;

(a3) $\mathbf{M}$ is the memory module of the agent A, which is a space to store sequences in Σ^*;

(a4) $v_{\text{end}} : \Sigma^* \to \{0, 1\}$, $v_{\text{out}} : \Sigma^* \to \Sigma^*$, $v_d : \Sigma^* \times \Sigma^* \to \Sigma^*$, and $v_r : \Sigma^* \times \Sigma^* \to \Sigma^*$ represent the tools used by the agent for stop checking, output, module connection, and memory writing.

Given an input $x \in \Sigma^*$ to A, such an agent operates in the following manner:

(b1) Let M be the sequence in the $\mathbf{M}$, and let $M = x$ at first;

(b2) Calculate $v_1 = \widehat{\mathcal{F}}_d(M)$. If $v_{\text{end}}(v_1) = 0$, go to (b3); otherwise, go to (b5);

(b3) Calculate $v_2 = \widehat{\mathcal{F}}_c(v_d(M, v_1))$;

(b4) $M = v_r(M, v_2)$ and go to (b2);

(b5) Output $A(x) = v_{\text{out}}(M)$.

This agent is defined such that its input space, memory space, and output space all lie within Σ^*. Both the decision and execution models are defined by the corresponding transformer, and $v_d, v_{\text{end}}, v_{\text{out}}$ and v_r represent external tools that the agent can call. The five steps (b1)-(b5) represent the five common parts in the operation of an agent: initialization, decision-making module operation, stop determination, execution module operation, writing in the memory module, and final output. For $q \in \mathbb{Z}_+ \cup \{\infty\}$, A_q means that each transformer in the agent has a precision of q. We further define that:

Definition 5.2. *We say that an agent A_q can memorize a Turing machine $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$ if it holds that $A_q(x) = \mathbb{T}(x)$ for every stoppable input x of $\mathbb{T}$.*

5.2 Agent is Turing Complete.

We show that the agent defined in Section 5.1 is Turing complete.

Theorem 5.1. *There are functions v_d, v_{out}, v_{end}, and v_r computable in linear time, such that for any Turing machine $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$, one can construct an agent A_q that is able to memorize $\mathbb{T}$. Moreover, A_q satisfies the following:*

(1) The transformers $\mathcal{F}_d, \mathcal{F}_c$ have constant depth $O(1)$, width $O(|Q| \cdot |\Sigma_2|)$, number of heads $O(\ln |Q|)$, and numerical precision $q = O(\ln |Q|)$.

(2) The agent executes step (b2) at most $t_{\mathbb{T}}(x) + 1$ times; the memory module requires at most $O(t_{\mathbb{T}}(x) + \text{len}(x) + \log_2(|Q|))$ space; the length of output for $\mathcal{F}_d, \mathcal{F}_c$ in each reasoning is $O(\ln |Q|)$.

As a consequence, the agent A_q is computationally the same as $\mathbb{T}$.

Proof. Each cycle of the agent memorizes one step of a Turing machine. The decision module decides whether to halt by checking for the existence of a state q_s. The execution module repeatedly memorizes the Turing machine's transfer function and saves the outcomes in the memory module. Finally, the output is produced according to the content in the memory module. We show the complete proof in Appendix G.2. □

In terms of time and space complexity, for any input x, such an agent needs $\text{Poly}(|Q| \cdot |\Sigma_2|, (t_{\mathbb{T}}(x) + \text{len}(x)))$-time and $O(t_{\mathbb{T}}(x) + \text{len}(x) + \ln |Q|)$ space to memorize $\mathbb{T}(x)$. This implies that the agent can memorize $\mathbb{T}$ with polynomial overhead. Therefore, such an agent is computationally the same as the Turing machine if $|Q|$ and $|\Sigma_2|$ are considered constants. From this, it can be seen that by using simple but appropriate tools that do not depend on the target Turing machine, agents can achieve Turing completeness.

Moreover, the two transformers $\mathcal{F}_c, \mathcal{F}_d$ in the agent can be the same one. We can consolidate $\mathcal{F}_c$ and $\mathcal{F}_d$ into a single transformer, $\mathcal{F}_s$, and include a suitable prompt so that $\mathcal{F}_s$ can distinguish whether it is performing decision-making or execution. The detail is given in the Appendix G.3.

In summary, this section states that for agents, by choosing appropriate tools (that do not depend on the specific target Turing machine), we can construct the decision model and the execution model to memorize any Turing machine.

Remark 5.1. *In comparison with (Chen & Liu, 2026), our results exhibit the following differences. Our agent maps directly to real-world agent architectures: the decision transformer $\mathcal{F}_d$ corresponds to the "Planner" that orchestrates workflows; the execution transformer $\mathcal{F}_c$ mirrors the "Worker" models; and the memory M represents the context management mechanism. We use finite-precision transformers, and the agent is computationally the same as $\mathbb{T}$.*

6 Experiment

Our theoretical analysis characterizes a computational separation between a single transformer and an agent that can repeatedly invoke transformers while maintaining a memory module. To complement our theoretical analysis, we empirically examine whether iterative LLM execution with external memory and tools improves reliability in increasingly long computations. We compare one-shot inference with an agent that performs the computation step by step, focusing on how their accuracy changes as the number of required operations grows.

6.1 Experiment Settings

We use arithmetic **Arith** because it provides unambiguous answers and direct control over computation length. The benchmark contains 1,300 expressions spanning 1 to 30 arithmetic operators and covers addition, subtraction, multiplication, division, and mixed operations. For each evaluated language model, we compare two inference architectures. Refer to Appendix I.1 for further details.

Single model. The model receives the complete expression in one request and must return its final answer before that request terminates. It may produce textual reasoning within the response, but it cannot call a tool, update external state, or invoke the model again.

Agent. Each component of the implementation directly instantiates one element of our theoretical agent framework in Definition 5.1. The memory module $\mathbf{M}_t$ stores the expression in iteration t,

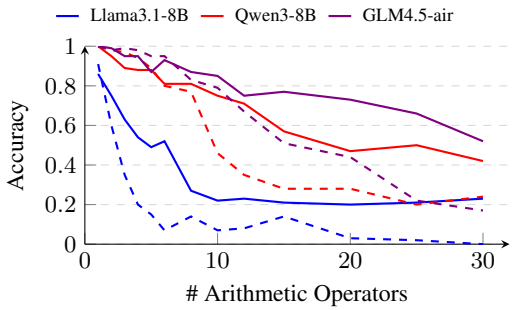


Figure 1: Accuracy rate of single model and agent as the number of required operations grows. Solid: agent; dashed: single model.

Failure Reason	Single Model	Agent
Protocol	11	90
Mathematics	341	126
Ordering	–	60
Arithmetic	–	66
Total errors	352	216
Accuracy	72.92%	83.38%

Table 1: Failure breakdown for GLM4.5-air in two settings. Protocol failures result from output truncation or invalid format, whereas mathematics failures reflect errors in operator ordering or arithmetic execution.

where $\mathbf{M}_0$ keeps the original input expression. The decision component $\mathcal{F}_d$ is an LLM that selects the identifier of the next operator. The linear-time read function v_d retrieves that operator and its two adjacent operands from $\mathbf{M}_t$. The execution module $\mathcal{F}_c$ is an LLM that computes this local binary operation. Finally, the linear-time replacement function v_r writes the value provided by $\mathcal{F}_c$ back to memory to construct $\mathbf{M}_{t+1}$. The process repeats until one value remains. Neither v_d nor v_r performs arithmetic, chooses an operator, or provides correctness feedback. We show a real example of the agent workflow in Appendix I.2.

6.2 Results

Figure 1 compares performance across expression lengths. For the three models, the agent increasingly and substantially outperforms its single-model counterpart as the number of operators grows, demonstrating that, compared to standard CoT alone, leveraging simple tool calls within a loop execution can effectively enhance the ability to solve long-horizon problems. At 30 operators, for example, agent accuracy exceeds single-model accuracy by 23, 18, and 35 percentage points for Llama3.1-8B, Qwen3-8B, and GLM4.5-air, respectively. For short expressions, however, the agent may match or underperform the single model. The reason is that the agent introduces multiple decision and execution steps, and an error at any one of these steps is sufficient to invalidate the final answer.

Table 1 reports the failure analysis specifically for GLM4.5-air. Protocol failures denote outputs that reach the generation limit or are in an invalid format, whereas mathematical failures capture either an incorrect operator ordering or erroneous arithmetic execution. Most importantly, the agent reduces mathematical failures markedly, from 341 under the single-model setting to 126, of which 60 are ordering errors and 66 are arithmetic errors. Although the agent incurs more protocol failures at the task level, all 90 arise from limitations in experimental conditions or the insufficient ability of the model to strictly follow instructions.

7 Conclusions

This paper investigates the Turing-completeness of transformers and agents under unbounded-length inputs. Prior results show that transformers can memorize Turing machines for bounded-length inputs. We show that finite-precision transformers suffer from precision errors and cannot memorize non-converging Turing machines such as Arithmetic. Even infinite-precision transformers exhibit sensitivity decay: input perturbations are diluted for long sequences, making them unable to memorize non-tame Turing machines under positive output confidence. Random training further yields a zero probability of obtaining such a transformer with limited chain-of-thought. By contrast, we show that agents with decision, execution, and memory modules are Turing complete even with finite precision transformers. At last, we empirically validate these theoretical results.

Limitations. It remains open whether infinite-precision transformers are Turing complete without additional assumptions and how to design computationally optimal agent architectures for fixed transformer scales.

STATEMENTS

AI USE STATEMENT

LLMs were used to revise the manuscript's text and to check the proofs. In the experiment, LLMs were also used to assist in generating the test data and debugging the code. They were not employed to produce the paper's primary results. All LLM-assisted edits and proof-related suggestions were reviewed and verified by the authors.

ETHICS STATEMENT

Our research is grounded in theoretical foundations, focusing primarily on understanding the expressive capabilities of transformers and agents, and it does not raise any ethical or moral concerns.

REFERENCES

Satwik Bhattamishra, Arkil Patel, and Navin Goyal. On the computational power of transformers and its implications in sequence modeling. In *Proceedings of the 24th Conference on Computational Natural Language Learning (CoNLL)*, pp. 455–475, 2020.

Moritz Brosamle and Stephan Eckstein. The expressive power of low precision softmax transformers with (summarized) chain-of-thought. *ICML*, 2026.

Liang Chen and Qi Liu. Neural paging: Learning context management policies for turing-complete agents. *arXiv preprint arXiv:2603.02228*, 2026.

Yang Chen, Yitao Liang, and Zhouchen Lin. On the limitations and capabilities of position embeddings for length generalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026. doi: 10.1109/TPAMI.2026.3688702.

David Chiang. Transformers in uniform TC^0. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=ZA7D4nQuQF.

Guanyu Cui, Zhewei Wei, and Kun He. Position: The turing-completeness of autoregressive transformers relies heavily on context management. *arXiv e-prints*, pp. arXiv–2605, 2026.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 70757–70798. Curran Associates, Inc., 2023. doi: 10.52202/075280-3100. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/dfc310e81992d2e4cedc09ac47eff13e-Paper-Conference.pdf.

Guhao Feng, Kai Yang, Yuntian Gu, Xinyue Ai, Shengjie Luo, Jiacheng Sun, Di He, Zhenguo Li, and Liwei Wang. How numerical precision affects arithmetical reasoning capabilities of LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 46–85, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.3. URL https://aclanthology.org/2025.findings-acl.3/.

Xinting Huang, Andy Yang, Satwik Bhattamishra, Yash Sarrof, Andreas Krebs, Hattie Zhou, Preetum Nakkiran, and Michael Hahn. A formal framework for understanding length generalization in transformers. In *ICLR*, 2025. URL https://arxiv.org/abs/2410.02140.

Hongjian Jiang, Michael Hahn, Georg Zetzsche, and Anthony W. Lin. Softmax transformers are turing-complete. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 43932–43952, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/48c218b01802a712ee8d7243d27fd9cf-Paper-Conference.pdf.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers, 2023. URL https://arxiv.org/abs/2305.19466.

Qian Li and Yuyi Wang. Constant bit-size transformers are turing complete. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 62273–62292. Curran Associates, Inc., 2025. doi: 10.52202/085713-2083. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/59d501de2e283999feb3bbca259ca4d1-Paper-Conference.pdf.

Qian Li and Yuyi Wang. Efficient turing machine simulation with transformers. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=bxVuILo1xx.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *ICLR2024*, 2023.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 7690–7706, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/1f59721c106ea80f613299039112f651-Paper-Conference.pdf.

William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. *arXiv preprint arXiv:2503.03961*, 2025.

William Merrill, Ashish Sabharwal, and Noah A. Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

Timur Mudarisov, Mikhail Burtsev, Tatiana Petrova, and Radu State. Limitations of normalization in attention mechanism. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is turing complete. *Journal of Machine Learning Research*, 22(1):3463–3497, 2021.

Jesse Roberts. How powerful are decoder-only transformer neural models? *arXiv preprint arXiv:2305.17026*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

Dale Schuurmans. Memory augmented large language models are computationally universal. *arXiv e-prints*, pp. arXiv:2301.04589, 2023.

Jie Wang, Tao Ji, Yuanbin Wu, Hang Yan, Tao Gui, Qi Zhang, Xuanjing Huang, and Xiaoling Wang. Length generalization of causal transformers without position encoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14024–14040, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.834. URL https://aclanthology.org/2024.findings-acl.834/.

Andy Yang, Pascal Bergsträßer, Georg Zetzsche, David Chiang, and Anthony W. Lin. Length generalization bounds for transformers. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2603.02238. To appear.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023. URL https://mlanthology.org/iclr/2023/yao2023iclr-react/.

Lijia Yu, Xiao-Shan Gao, and Lijun Zhang. Analyzing the power of chain of thought through memorization capabilities. In *Advances in Neural Information Processing Systems (NeurIPS 2025)*, 2025.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Joshua Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 15898–15926, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/45ed1a72597594c097152ef9cc187762-Paper-Conference.pdf.

A Detailed Notations

Appendix A clarifies the main definitions, assumptions, and concepts used in the paper.

A.1 Complete Turing machine

Firstly, we define a Turing machine as follows:

Definition A.1. *A Turing machine $\mathbb{T}$ is a tuple*

$$\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta) \quad \textit{or more detailed} \quad \mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta, q_0, q_s, B)$$

defined subject to the following conditions:

(a1): Q is a finite set of states containing two distinguished states $q_0, q_s \in Q$. The symbol q_0 is the initial state, and q_s is the unique halt state.

(a2): Σ_1 is a finite input alphabet, and $B \notin \Sigma_1$, where B denotes the blank tape symbol.

(a3): Σ_2 is the tape alphabet satisfying $\Sigma_1 \cup \{B\} \subseteq \Sigma_2$.

(a4): The transition function is a transfer function

$$\delta \colon (Q \setminus \{q_s\}) \times \Sigma_2 \to Q \times \Sigma_2 \times \{-1, 0, 1\}.$$

A configuration of $\mathbb{T}$ is a triple (q_p, σ, w_p) where $q_p \in Q$ is the current state, $\sigma \colon \mathbb{Z} \to \Sigma_2$ is the tape function, assigning to each integer tape position a symbol from Σ_2, and $w_p \in \mathbb{Z}$ is the current position of the read/write head.

The computation rules for $\mathbb{T}$:

(b1): Initial configuration. Given an input string $x = x_0 x_1 \cdots x_{n-1} \in \Sigma_1^*$, the initial tape satisfies

$$\sigma(k) = \begin{cases} x_k & 0 \le k \le n-1, \\ B & \textit{otherwise}. \end{cases}$$

The initial configuration sets $q_p = q_0$, $w_p = 0$.

(b2): Single transition step. Suppose the current configuration satisfies $q_p \neq q_s$. Let $\sigma(w_p)$ be the symbol at head position w_p. Compute

$$(q_\delta, \sigma_\delta, s_\delta) = \delta\big(q_p, \sigma(w_p)\big).$$

The machine transitions to a new configuration by updating:

$$q_p := q_\delta, \quad \sigma(w_p) := \sigma_\delta, \quad w_p := w_p + s_\delta.$$

(b3): Halt condition and output. If $q_p = q_s$, the machine halts. Let $L = \min\{k \in \mathbb{Z} \mid \sigma(k) \neq B\}$ and $R = \max\{k \in \mathbb{Z} \mid \sigma(k) \neq B\}$. The output string is $\sigma(L)\sigma(L+1)\cdots\sigma(R)$. If no such L, R exists (all tape symbols are blank), the output is the empty string.

Remark A.1. *Without loss of generality, in this paper, we assume that the final output of the Turing machine does not contain the symbol B.*

A.2 Complete transformer structure

We consider a fixed-architecture soft-attention transformer. The model consists of an embedding layer, several transformer blocks, and an output layer.

Embedding layer E_{emb}. The transformer first maps each symbol $\sigma_i \in \Sigma$ to a vector $v_i \in \mathbb{R}^d$, where d is the embedding dimension. For an input sequence $x = (\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_n}) \in \Sigma^*$ with length n, its embedded representation is written as $x^{(0)} = (v_{i_1}, v_{i_2}, \ldots, v_{i_n})^\top \in \mathbb{R}^{n \times d}$. When no confusion arises, we do not distinguish between the symbolic sequence x and its embedded matrix representation and use the same notation for both.

Multi-head attention. For an input $x \in \mathbb{R}^{n \times d}$, the output of the h-th attention head is defined by

$$\mathrm{ATT}_h(x) = \mathrm{softmax}(R(xQ_h, K_h x^\top) + M)\, xV_h,$$

where $Q_h \in \mathbb{R}^{d\times W}$, $K_h \in \mathbb{R}^{W\times d}$, $V_h \in \mathbb{R}^{d\times d}$. Without loss of generality, we take $W = d$ in this paper. $M \in \{-\infty, 0\}^{n\times n}$ is the causal mask satisfying $M_{i,j} = -\infty \iff j > i$.

Relative position embedding $R(xQ_h, K_h x^\tau) \in \mathbb{R}^{n\times n}$ is used, where the (u, v)-th weight of $R(xQ_h, K_h x^\tau)$ is $x[u]Q_h A_{u-v} K_h x^\tau[v]$, $x[i]$ is the i-th row of x, and $A_{u-v} \in [-1, 1]^{W\times W}$ is the matrix for the relative position embedding. Note that A_{u-v} does not depend on which head or layer is used. Summing over all H heads gives the total multi-head attention output:

$$\mathrm{ATT}(x) = \sum_{h=1}^{H} \mathrm{ATT}_h(x).$$

Position Embedding.

For position encoding, according to the commonly used form of RoPE, let $K = \lfloor \frac{W}{2} \rfloor$.The position encoding is parameterized by $\{C_i\}_{i=1}^{K}$. For $r = n - m$, define $A_r \in \mathbb{R}^{W\times W}$ by

$$(A_r)_{2i-1,2i-1} = (A_r)_{2i,2i} = \cos(rC_i), \qquad i \in [K],$$

$$(A_r)_{2i-1,2i} = \sin(rC_i), \qquad i \in [K],$$

$$(A_r)_{2i,2i-1} = -\sin(rC_i), \qquad i \in [K].$$

If W is odd, we additionally set $(A_r)_{W,W} = 1$.All remaining entries of A_r are set to zero.

Equivalently,

$$A_r = \mathrm{diag}\big(R(rC_1), \ldots, R(rC_K), I_{W-2K}\big),$$

where

$$R(\theta) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}.$$

This ensures that the position encoding remains uniformly bounded as the relative position increases. We treat $\{C_i\}_{i=1}^{K}$ as adjustable parameters throughout this paper.

Feedforward network. For $x \in \mathbb{R}^{n\times d}$, the feedforward layer is defined as

$$\mathrm{FNN}(x) = \mathrm{ReLU}(xE_1 \oplus b)E_2,$$

where $E_1 \in \mathbb{R}^{d\times W}$, $b \in \mathbb{R}^{1\times W}$, $E_2 \in \mathbb{R}^{W\times d}$ are parameters, and $xE_1 \oplus b$ means that the bias vector b is added to every row of xE_1.

Transformer block. Let $x^{\ell-1}$ be the input to layer ℓ, and let $x^0 = x$. Then we have that:

$$x^\ell = x^{\ell-1} + \mathrm{ATT}_\ell(x^{\ell-1}) + \mathrm{FNN}_\ell(x^{\ell-1} + \mathrm{ATT}_\ell(x^{\ell-1})).$$

Output layer E_{out}. Assume the transformer has L hidden layers. The output layer applies a linear transformation only to the final position of the last hidden layer:

$$\mathcal{F}(x) = x^L[k]W_0 + c \in \mathbb{R}^{T+1},$$

where $k = \mathrm{len}(x^L)$, $W_0 \in \mathbb{R}^{d\times(T+1)}$, $c \in \mathbb{R}^{1\times(T+1)}$.

The Output of the autoregressive transformer. The final output $y = \widehat{\mathcal{F}}(x)$ is obtained as follows, with the initial value $y = ()$:

(1) The next token is $\sigma_s \in \Sigma$, where $s = \arg\max_{i\in[T+1]} \mathcal{F}_i(x \oplus y)$ and $\mathcal{F}_i(x)$ is the i-th weight of $\mathcal{F}(x)$. Let $\sigma_{T+1} = \sigma_0$.

(2) If $\sigma_s \neq \sigma_0$, then let $y = y \oplus \{\sigma_s\}$ and return to step (1).

(3) If $\sigma_s = \sigma_0$, stop and return the output y. For convenience, we say $y = \widehat{\mathcal{F}}(x) \in \Sigma^*$.

A.3 MEMORIZATION, SIMULATION, AND TURING COMPLETENESS

Two forms of correspondence between transformers and Turing machines. Some previous work has attempted to study the power of Transformers in simulating Turing machines, whereas we consider the power of Transformers in memorizing Turing machines. Simulating Turing machines aims to reproduce the calculation process of Turing machines step-by-step, hence establishing computational equivalence. In contrast, memory Turing machines only require the final output to match that of Turing machines, without requiring the process to completely reproduce the internal computation. In computability theory, Turing completeness can be achieved by either simulating or memorizing a general Turing machine. Both approaches establish that the model is equivalent to a Turing machine in computing power.

Throughout this paper, our primary focus is on how Transformers simulate Turing machines on inputs that guarantee halting. Since we are concerned with computational Turing machines such as arithmetic rather than focus on studying the halting problem, we impose no specific constraints on the Transformer's behavior in non-halting scenarios.

Statements about the "Turing completeness" of transformers in the existing literature can correspond to two different levels of construction.

(1) Family-level correspondence. Given a Turing machine T, for each input length n, one constructs a transformer

$$\mathcal{F}_n$$

that depends on n and can memorize the computation of T on all inputs of length n. This gives a model family

$$\{\mathcal{F}_n\}_{n\geq 1}.$$

In this sense, the correspondence is

$$\text{input length } n \quad \longmapsto \quad \text{model } \mathcal{F}_n,$$

meaning that different input lengths are allowed to use different transformers.

(2) Single-model correspondence. A stronger requirement asks whether there exists one fixed transformer $\mathcal{F}$ such that, for inputs of all lengths to the Turing machine T, the same $\mathcal{F}$ can uniformly memorize the computation. In this sense, the correspondence is no longer between a length n and a model $\mathcal{F}_n$, but rather

$$\text{Turing machine } T \quad \longmapsto \quad \text{one fixed model } \mathcal{F}.$$

The same model must cover the entire input-size range of T.

The key difference is, therefore, that the first formulation allows the model to vary with input length and studies a family of transformer models. The second keeps the model fixed and studies the uniform processing ability of one transformer over arbitrary-length inputs. The first captures length-by-length simulation, whereas the second captures uniform simulation by a single model.

A.4 THE ARITHMETIC TASK

Details for Example 1. Let $\Sigma_1 = \{0,1,2,3,4,5,6,7,8,9,+,-,*,/,(,),|\}$, $\Sigma_2 = \Sigma_1 \cup \{B\}$, and $\mathbb{T}_{\mathbf{Arith}} = (Q, \Sigma_1, \Sigma_2, \delta)$ be the Turing machines for evaluating arithmetic expressions over rational number, with $n|m$ to represent a rational number $\frac{n}{m}$. The arithmetic problem $\mathbf{Arith}$ is the set $S_{\mathbf{Arith}} = \{(x, y)\} \subset \Sigma_1^* \times \Sigma_1^*$, where x is an arithmetic formula such as $2 + 2 * (5 - 1) + 3$, and y is the answer to x, such as 13.

We say a Turing machine $\mathbb{T}_{\mathbf{Arith}} = (Q, \Sigma_1, \Sigma_2, \delta)$ is the Turing machine to solve four arithmetic problems if $\mathbb{T}_{\mathbf{Arith}}(x) = y$ for any $(x, y) \in S_{\mathbf{Arith}}$, and $\mathbb{T}_{\mathbf{Arith}}(x)$ does not halt if $x \in \Sigma_1^*$ is not an arithmetic formula.

Let $\Sigma = \Sigma_2 \cup \{=\}$ and σ_T be '='. For a transformer $\mathcal{F}$, if $\widehat{\mathcal{F}}(x)$ is '$= 2 + 2 * 4 + 3 = 2 + 8 + 3 = 10 + 3 = 13$' and then output σ_0 to halt. Then $\widehat{\mathcal{F}}_{\text{out}}(x)$ is 13, which is the sequence between the last $=$ and σ_0, other parts are CoT. If $\hat{\mathcal{F}}_{\text{out}}(x) = y$ for any $(x, y) \in S_{\mathbf{Arith}}$, then such a transformer $\mathcal{F}$ can memorize $\mathbb{T}_{\mathbf{Arith}}$.

Remark A.2. *It is reasonable to require the Turing machine $\mathbb{T}_{\mathbf{Arith}}$ not to stop on equations that do not conform to the calculation rules (such as $1 + 2 * *6+$); we can also require the Turing machine to output a symbol representing rejection for such equations. The definition of such inputs is not crucial and does not affect the following conclusions.*

B Recent results on memorizing Turing machines when inputs are restricted to a fixed length

A Turing machine $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$ is said to have input length n if every input it receives has a length of at most n. For example, arithmetic over $\mathbb{Z}_p$ on inputs of length n (denoted $\mathbf{Arith}_{p,n}$) can be implemented by a Turing machine with input length n.

Since there are finitely many inputs for a Turing machine $\mathbb{T}$ with input length n, we may define its maximal running time:

$$t_{\mathbb{T}} = \max_{x \text{ is stoppable}} \{|t_{\mathbb{T}}(x)|\}.$$

A number of prior studies (Pérez et al., 2021; Merrill & Sabharwal, 2024; Yu et al., 2025; Brosamle & Eckstein, 2026) have demonstrated that a CoT transformer is capable of simulating such a Turing machine. The size of the memorization transformer and the length of the CoT can be divided into two separate groups:

Variant-precision transformer (Brosamle & Eckstein, 2026): Let $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$ be a Turing machine with input length n. Then there exists a transformer $\mathcal{F}$ with a token set Σ_0 such that $\Sigma_0 \supset \Sigma_2$ and $|\Sigma_0| > |\Sigma_2| + O(|Q| \cdot |\Sigma_2|)$, with depth $O(\ln t_{\mathbb{T}})$, width $O(\max\{|Q| \cdot |\Sigma_2|, \ln t_{\mathbb{T}}\})$, $O(1)$ heads, and precision $O(\ln \ln t_{\mathbb{T}})$, which is capable of memorizing $\mathbb{T}$ using a CoT whose length is $O(t_{\mathbb{T}}(x))$.

Fix-precision transformer (Yu et al., 2025): Let $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$ be a Turing machine with input length n. Then, for any $q \geq 2$, there exists a transformer $\mathcal{F}_q$ with a token set Σ_0 satisfying $\Sigma_0 \supset \Sigma_2$ and $|\Sigma_0| \geq |\Sigma_2| + 1$, with depth $O(T^{O(t_{\mathbb{T}})})$ and $O(T)$ heads, which is capable of memorizing $\mathbb{T}$ using a CoT of length $O(1)$.

Taken together, these two results indicate that a Turing machine with inputs of bounded length can be memorized by a transformer, even when only a short CoT is used. In practice, most Turing machines are not subject to such bounds. Consequently, in the main paper, we focus mainly on the more common setting of general Turing machines.

C Incompleteness of Finite-Precision Transformers

In this section, we will prove Theorem 4.1.

C.1 Detailed Definition for Finite-Precision Transformers

We first provide a precise definition of finite-precision computation in a Transformer.

Definition C.1. *For $x \in \mathbb{R}$ with $|x| \leq 10^q$, let $[x]_q$ denote the value obtained by rounding x to q decimal places. If $|x| > 10^q$, define*

$$[x]_q = sign(x)\mathrm{Inf}.$$

For a vector or matrix M, define $[M]_q$ entry-wise by

$$([M]_q)_{i,j} = [M_{i,j}]_q.$$

We now define the q-precision version of a Transformer.

Definition C.2. *Let $\mathcal{F}$ be a Transformer. Its q-precision version, denoted by $\mathcal{F}_q$, is defined as follows.*

First, $\mathcal{F}$ and $\mathcal{F}_q$ have the same depth L, width W, number of heads H, and vocabulary Σ. Every parameter p of $\mathcal{F}$ is replaced by $[p]_q$ in $\mathcal{F}_q$.

Second, all arithmetic operations in the positional embedding, hidden layers, and output layer are carried out under q-precision.

For the trigonometric functions in the positional encoding, we maintain the reduced phase recursively so that an unbounded positional product need not be formed. Let $M_q := 2[\pi]_q$.

For each q-precision frequency ω, let

$$\bar{\omega} = \omega - M_q \left\lfloor \frac{\omega}{M_q} \right\rfloor \in [0, M_q).$$

Define $r_0 = 0$ and, for $n \geq 0$,

$$r_{n+1} = \begin{cases} r_n + \bar{\omega}, & r_n < M_q - \bar{\omega}, \\ r_n - (M_q - \bar{\omega}), & r_n \geq M_q - \bar{\omega}. \end{cases}$$

Thus $0 \leq r_n < M_q$ for every n, and the possibly unbounded product $n\omega$ is never formed during the positional-encoding calculation.

We then define

$$\sin_q(n\omega) = [\sin(r_n)]_q, \qquad \cos_q(n\omega) = [\cos(r_n)]_q.$$

For vector addition, define

$$[x + z]_q = \big([x_i + z_i]_q\big)_i. \tag{1}$$

For the inner product, every intermediate multiplication is also rounded:

$$\langle x, z \rangle_q = \left[\sum_{i=1}^{M} [x_i z_i]_q \right]_q. \tag{2}$$

For softmax, let $x_{\max} = max_{1 \leq j \leq M} x_j$.

Define

$$E_i = \left[e^{[x_i - x_{\max}]_q} \right]_q \quad \textit{and} \quad Z = \left[\sum_{j=1}^{M} E_j \right]_q. \tag{3}$$

Then $\mathrm{softmax}_q(x)_i = \left[\frac{E_i}{Z}\right]_q$.

Finally, values outside the precision range are handled according to

$$\frac{c}{\mathrm{Inf}} = 0, \qquad c + \mathrm{Inf} = \mathrm{Inf} \qquad when\ c \neq \mathrm{Inf}, \tag{4}$$

$$c_{\neq 0} \cdot \mathrm{Inf} = \mathrm{Inf}, \qquad \frac{\mathrm{Inf}}{0} = \mathrm{Inf}, \tag{5}$$

and

$$\frac{c}{0} = \frac{\mathrm{Inf}}{\mathrm{Inf}} = (+\mathrm{Inf}) - (+\mathrm{Inf}) = (-\mathrm{Inf}) - (-\mathrm{Inf}) = 0 \cdot \mathrm{Inf} = \mathrm{NaN}. \tag{6}$$

If NaN *appears at any intermediate step, the final output is also defined to be* NaN*. Without loss of generality, we require that neither* Inf *nor* NaN *appears in the final output of the transformer in any results in this paper.*

C.2 Proof of Theorem 4.1

We begin with a periodicity property of finite-precision positional encoding.

Lemma C.1. *For every q-precision Transformer $\mathcal{F}_q$, there exists a positive integer C such that*

$$A_i = A_{i+C}$$

for every position i. Thus, under finite precision, the positional encoding is periodic.

Proof. After q-precision rounding, every frequency parameter appearing in the positional encoding is rational. Let the corresponding reduced frequencies be $\bar{\omega}_1, \ldots, \bar{\omega}_R$, and let $M_q = 2[\pi]_q$. Since all these numbers have at most q decimal places, for each r there exist integers a_r and $b > 0$ such that $\bar{\omega}_r = a_r/10^q$ and $M_q = b/10^q$.

Under the recursive phase update in Definition C.2, the phase corresponding to $\bar{\omega}_r$ is $r_n^{(r)} = (na_r \bmod b)/10^q$. Hence, this phase is periodic with period $C_r = b/\gcd(a_r, b)$.

Let $C = \operatorname{lcm}(C_1, \ldots, C_R)$. Then $r_{n+C}^{(r)} = r_n^{(r)}$ for every n and every r. Therefore, all sine and cosine components of the RoPE positional encoding agree at positions n and $n + C$, and hence $A_n = A_{n+C}$. □

We repeat Theorem 4.1 below.

Theorem C.2. *Let $\mathbb{T}$ be a non-converging Turing machine. Then, for any $q \in \mathbb{N}_+$ and any fixed precision transformer $\mathcal{F}_q$, $\mathcal{F}_q$ cannot memorize $\mathbb{T}$. Consequently, a fixed-precision transformer cannot be Turing complete.*

We can now prove Theorem 4.1.

Proof. For a q-precision Transformer $\mathcal{F}_q$, let $\mathcal{F}_q(s)_{m,n}$ denote the hidden state at position n in the m-th hidden layer. Let $\mathcal{T}$ be a non-converging Turing machine. For $x, u, z \in \Sigma^*$ and $k \in \mathbb{N}_+$, define $s_{x,u,k,z} := x \oplus u^k \oplus z$.

We shall repeatedly use causality: if two input sequences have the same prefix up to position r, then their hidden states at every position $i \le r$ are identical at every hidden layer. Assume, toward a contradiction, that there exist $q \in \mathbb{Z}_+$ and a q-precision Transformer $\mathcal{F}_q$ that memorizes $\mathcal{T}$.

Part One.

Fix $x, u \in \Sigma^*$ with $u \neq \varnothing$, an integer $k \ge C + 1$, a continuation $w \in \Sigma^*$, and a hidden-layer index m. Let $P := C|u|$, $s^- := s_{x,u,k-C,w}$, $s^+ := s_{x,u,k,w}$, and $N_\pm := |s^\pm|$, so that $N_+ = N_- + P$.

Suppose that the following three conditions hold.

First, the final hidden states agree:

$$\mathcal{F}_q(s^-)_{m,N_-} = \mathcal{F}_q(s^+)_{m,N_+}. \tag{c1}$$

Second, for every $1 \le t \le |w|$,

$$\mathcal{F}_q(s^-)_{m,|x|+(k-C)|u|+t} = \mathcal{F}_q(s^+)_{m,|x|+k|u|+t}. \tag{c2}$$

Third, for every $i \in \{|x| + (k - C)|u| + 1, \ldots, |x| + k|u|\}$, there exist more than 10^{2q} distinct indices $j \le |x| + (k - C)|u|$ such that

$$\mathcal{F}_q(s^+)_{m,i} = \mathcal{F}_q(s^-)_{m,j}, \qquad A_{N_+-i} = A_{N_--j}. \tag{c3}$$

We claim that

$$\mathcal{F}_q(s^-)_{m+1,N_-} = \mathcal{F}_q(s^+)_{m+1,N_+}. \tag{7}$$

At the final position, a Transformer layer consists of the residual connection, the attention output, and the feedforward update. By condition equation c1, the residual inputs at the two final positions are identical, so it remains only to compare their attention outputs.

Fix an attention head h. Write $h_i^\pm := \mathcal{F}_q(s^\pm)_{m,i}$ and let $\lambda_i^{(\pm,h)} := [h_{N_\pm}^\pm Q_h A_{N_\pm - i} K_h(h_i^\pm)^\top]_q$.

Every position of s^- has a corresponding position in s^+. On the common prefix $x \oplus u^{k-C}$, the hidden states agree by causality, while the relative position indices differ by $P = C|u|$. Since $A_r = A_{r+C}$, we also have $A_{r+P} = A_r$. On the common suffix w, the hidden states agree by condition equation c2, and the relative position indices are identical. Together with condition equation c1, the corresponding logits and value vectors are therefore equal.

Moreover, by condition equation c3, every additional position in s^+ has more than 10^{2q} matching positions in s^- with the same hidden state and the same relative positional matrix. Hence the corresponding attention logits also agree.

Thus the maximum logit is the same in the two computations; denote it by W_h. Let $E_i^{(\pm,h)} := [e^{[\lambda_i^{(\pm,h)} - W_h]_q}]_q$ and $Z_\pm^{(h)} := [\sum_{i=1}^{N_\pm} E_i^{(\pm,h)}]_q$.

If every additional position in s^+ satisfies $E_i^{(+,h)} = 0$, then these positions contribute nothing, while all remaining logits and value vectors match. Hence the two attention-head outputs are equal.

Otherwise, some additional position satisfies $E_i^{(+,h)} > 0$. By condition equation c3, more than 10^{2q} positions in s^- have the same positive exponential value. Since every positive q-precision number is at least 10^{-q},

$$\sum_{j=1}^{N_-} E_j^{(-,h)} > 10^{2q} 10^{-q} = 10^q,$$

so $Z_-^{(h)} = \mathrm{Inf}$. The same matching positions occur in s^+, hence $Z_+^{(h)} = \mathrm{Inf}$. By the convention $c/\mathrm{Inf} = 0$, all attention weights vanish in both computations. Thus the two attention-head outputs are again equal.

The same argument applies to every head. Together with condition equation c1, this proves equation 7.

Part Two.

We now prove a layerwise relation uniform over all continuations. Let $R_q := 10^{2q} + 1$. We prove by induction on m that there exists an integer $K_m \geq C + 1$ such that, for every $k \geq K_m$ and every $w \in \Sigma^*$,

$$\mathcal{F}_q(s_{x,u,k-C,w})_{m,|s_{x,u,k-C,w}|} = \mathcal{F}_q(s_{x,u,k,w})_{m,|s_{x,u,k,w}|}. \tag{$\mathcal{I}_m$}$$

For $m = 0$, the final token is identical in the two sequences: it is the last token of w if $w \neq \epsilon$, and the last token of u otherwise. Thus the claim holds, for example, with $K_0 = C + 1$.

Suppose equation $\mathcal{I}_m$ holds at layer m, and set $K_{m+1} := K_m + C(R_q + 1)$. Fix $k \geq K_{m+1}$ and $w \in \Sigma^*$. We verify the three conditions of Part One.

Condition equation c1 is exactly the induction hypothesis applied to w. For each $1 \leq t \leq |w|$, applying the induction hypothesis to the continuation $w[1 : t]$ and using causality gives condition equation c2.

For condition equation c3, fix $i \in \{|x| + (k - C)|u| + 1, \ldots, |x| + k|u|\}$. There are unique $b \in \{k - C + 1, \ldots, k\}$ and $t \in \{1, \ldots, |u|\}$ such that $i = |x| + (b - 1)|u| + t$. For $1 \leq r \leq R_q$, define $j_r := i - rP$, where $P = C|u|$.

Because $k \geq K_m + C(R_q + 1)$, all j_r lie in the common repeated u-block, and all applications of the induction hypothesis below involve an exponent at least K_m. Applying equation $\mathcal{I}_m$ successively to the continuation $u[1 : t]$, together with causality, gives

$$\mathcal{F}_q(s_{x,u,k,w})_{m,i} = \mathcal{F}_q(s_{x,u,k-C,w})_{m,j_r}, \qquad 1 \leq r \leq R_q.$$

Moreover, $|s_{x,u,k-C,w}| - j_r = |s_{x,u,k,w}| - i + (r - 1)P$. Since $P = C|u|$ is a multiple of the RoPE period C, Lemma C.1 gives

$$A_{|s_{x,u,k-C,w}|-j_r} = A_{|s_{x,u,k,w}|-i}.$$

The indices j_r are distinct, and $R_q = 10^{2q} + 1 > 10^{2q}$. Hence condition equation c3 holds.

Part One therefore implies equation $\mathcal{I}_m$ at layer $m + 1$. This completes the induction.

Part Three.

We now derive the contradiction. Since $\mathcal{T}$ is non-converging, choose witnesses $x, u, z \in \Sigma_1^*$ from Definition 4.2; necessarily $u \neq \varnothing$. Let L be the depth of $\mathcal{F}_q$ and choose any $k \geq K_L$.

By Part Two, for every continuation $v \in \Sigma^*$,

$$\mathcal{F}_q(s_{x,u,k-C,z\oplus v})_{L,|s_{x,u,k-C,z\oplus v}|} = \mathcal{F}_q(s_{x,u,k,z\oplus v})_{L,|s_{x,u,k,z\oplus v}|}. \tag{8}$$

Applying the common output layer gives $\mathcal{F}_q(s_{x,u,k-C,z\oplus v}) = \mathcal{F}_q(s_{x,u,k,z\oplus v})$ for every $v \in \Sigma^*$.

Taking $v = \varnothing$, the first generated tokens are identical. Inductively, if both generations have produced the same prefix v, then the same equality applied to that v shows that their next tokens are also identical. The same argument applies to the EOS decision. Hence

$$\widehat{\mathcal{F}}_q(s_{x,u,k-C,z}) = \widehat{\mathcal{F}}_q(s_{x,u,k,z}).$$

Therefore their extracted final outputs are identical. However, by Definition 4.2, both inputs are stoppable and

$$\mathcal{T}(s_{x,u,k-C,z}) \neq \mathcal{T}(s_{x,u,k,z}),$$

contradicting the assumption that $\mathcal{F}_q$ memorizes $\mathcal{T}$. Therefore no q-precision Transformer can memorize a non-converging Turing machine. □

D Sensitivity Bounds for Infinite-Precision Transformers

In this section, we will prove Theorems 4.2 and 4.4.

D.1 Lipschitz Continuity of Basic Maps

This section proves the Lipschitz continuity, on relevant bounded regions, of several basic component maps in a standard transformer defined in Section A.2, which will be used in Section D.2.

Notation. Unless otherwise stated, vector norms are Euclidean norms $\|\cdot\|_2$, and matrix norms are induced operator norms, still denoted by $\|\cdot\|$. For a vector-valued map f, if there exists a constant $L > 0$ such that

$$\|f(x) - f(y)\|_2 \leq L\|x - y\|_2$$

for all x, y in its domain, then f is said to be L-Lipschitz on that region.

Lemma D.1 (Composition and addition preserve Lipschitz continuity)**.**

- *If $f : X \to Y$ is L_f-Lipschitz and $g : Y \to Z$ is L_g-Lipschitz, then $g \circ f$ is $L_g L_f$-Lipschitz.*
- *If $f, g : X \to \mathbb{R}^d$ are L_f- and L_g-Lipschitz respectively, then $f + g$ is $(L_f + L_g)$-Lipschitz.*

Proof. The first statement follows from

$$\|g(f(x)) - g(f(y))\| \leq L_g\|f(x) - f(y)\| \leq L_g L_f\|x - y\|.$$

The second follows from

$$\|(f + g)(x) - (f + g)(y)\| \leq \|f(x) - f(y)\| + \|g(x) - g(y)\| \leq (L_f + L_g)\|x - y\|.$$

□

Lemma D.2 (Lipschitz continuity of affine maps)**.** *Let $A \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$, and $T(u) = Au + b$. Then T is globally Lipschitz, with a Lipschitz constant of at most $\|A\|$.*

Proof. For any $u, v \in \mathbb{R}^d$, $T(u) - T(v) = A(u - v)$. Therefore

$$\|T(u) - T(v)\|_2 = \|A(u - v)\|_2 \leq \|A\|\,\|u - v\|_2.$$

Thus T is $\|A\|$-Lipschitz. □

Lemma D.3 (Lipschitz continuity of ReLU)**.** *The coordinatewise ReLU map*

$$\mathrm{ReLU}(z) = (\max\{z_1, 0\}, \ldots, \max\{z_d, 0\})$$

is globally 1*-Lipschitz.*

Proof. For the one-dimensional function $\rho(t) = \max\{t, 0\}$, we have $|\rho(a) - \rho(b)| \le |a - b|$ for any $a, b \in \mathbb{R}$. Summing over coordinates gives

$$\|\operatorname{ReLU}(u) - \operatorname{ReLU}(v)\|_2^2 = \sum_{k=1}^{d} |\rho(u_k) - \rho(v_k)|^2 \le \sum_{k=1}^{d} |u_k - v_k|^2 = \|u - v\|_2^2.$$

Hence ReLU is globally 1-Lipschitz. □

Lemma D.4 (Lipschitz continuity of addition)**.** *Define*

$$S(u_1, \dots, u_m) = u_1 + \cdots + u_m, \qquad u_i \in \mathbb{R}^d.$$

Then S is Lipschitz with respect to the product-space variables. More precisely, if the product space is equipped with

$$\|(u_1, \dots, u_m)\|_{\oplus} := \sum_{i=1}^{m} \|u_i\|_2,$$

then S is 1-Lipschitz. If it is equipped with the Euclidean product norm

$$\|(u_1, \dots, u_m)\|_{2,\oplus} := \Big(\sum_{i=1}^{m} \|u_i\|_2^2\Big)^{1/2},$$

then S is $\sqrt{m}$-Lipschitz.

Proof. For any $(u_1, \dots, u_m)$ and $(v_1, \dots, v_m)$,

$$S(u_1, \dots, u_m) - S(v_1, \dots, v_m) = \sum_{i=1}^{m} (u_i - v_i).$$

By the triangle inequality,

$$\|S(u_1, \dots, u_m) - S(v_1, \dots, v_m)\|_2 \le \sum_{i=1}^{m} \|u_i - v_i\|_2 = \|(u_1 - v_1, \dots, u_m - v_m)\|_{\oplus}.$$

Thus S is 1-Lipschitz under $\|\cdot\|_{\oplus}$.

By Cauchy–Schwarz inequality,

$$\sum_{i=1}^{m} \|u_i - v_i\|_2 \le \sqrt{m}\Big(\sum_{i=1}^{m} \|u_i - v_i\|_2^2\Big)^{1/2}.$$

Thus S is $\sqrt{m}$-Lipschitz under $\|\cdot\|_{2,\oplus}$. □

Obviously, the following conclusion can be drawn from the above lemma.

Lemma D.5 (Lipschitz continuity of FNN)**.** *Let* $\mathrm{FNN}(x) = \operatorname{ReLU}(xE_1 + b)E_2$*, where* $E_1 \in R^{d\times w}, E_2 \in \mathbb{R}^{w\times d}, b \in \mathbb{R}^w$.

Then FNN *is globally Lipschitz, with Lipschitz constant* $L_{\mathrm{FNN}} \le \|E_1\|\,\|E_2\|$.

Combining the preceding Lipschitz properties of addition and the feed-forward network, we obtain the following result for the position-level update map.

Proposition D.6 (Lipschitz continuity of the position-level update on relevant bounded regions)**.** *Consider the position-level update map of a standard transformer:*

$$f^{(\ell)}(u, b_1, \dots, b_H) = u + \sum_{h=1}^{H} b_h + \mathrm{FNN}_\ell\left(u + \sum_{h=1}^{H} b_h\right).$$

If all parameters are uniformly bounded and the input $(u, b_1, \dots, b_H)$ lies in a bounded region, then for every layer ℓ, $f^{(\ell)} : \mathbb{R}^d \times (\mathbb{R}^d)^H \to \mathbb{R}^d$ is Lipschitz with respect to all input variables.

D.2 Proof of Theorem 4.2

We repeat Theorem 4.2 below.

Theorem D.7. *For any input x and any transformer $\mathcal{F}$ previously defined with depth L, we have*

$$\operatorname{sen}(\mathcal{F}, x) \le C_{\mathcal{F}} \frac{\ln^{L-1}(\operatorname{len}(x))}{\operatorname{len}(x)},$$

where $C_{\mathcal{F}}$ is a constant that depends only on $\mathcal{F}$.

Now we prove Theorem 4.2.

Proof. Let $n = \operatorname{len}(x)$, and take any $x' \in B(x, 1)$. If $x' = x$, then $\|\mathcal{F}(x) - \mathcal{F}(x')\|_2 = 0$. Otherwise, x and x' differ at exactly one position $i < n$. Let $y_j^{(\ell)}$ and $\tilde{y}_j^{(\ell)}$ denote the hidden states at position j after layer ℓ corresponding to x and x', respectively, and define $\Delta y_j^{(\ell)} := y_j^{(\ell)} - \tilde{y}_j^{(\ell)}$.

Since Σ is finite, the embedding vectors are uniformly bounded. Let $M_0 := \max_{\sigma\in\Sigma} \|v_\sigma\|_2$ and $D_0 := \max_{\sigma,\tau\in\Sigma} \|v_\sigma - v_\tau\|_2$. Then $\|\Delta y_i^{(0)}\|_2 \le D_0$ and $\Delta y_j^{(0)} = 0$ for $j \ne i$. Moreover, by causality, $\Delta y_j^{(\ell)} = 0$ for all $j < i$ and $0 \le \ell \le L$.

We prove that, for every $1 \le \ell \le L$, there exists a constant $C_\ell > 0$, depending only on the fixed transformer $\mathcal{F}$, such that

$$\|\Delta y_j^{(\ell)}\|_2 \le C_\ell \frac{(\ln j)^{\ell-1}}{j}, \qquad j > i. \tag{9}$$

Step 1: Uniform boundedness and basic perturbation estimates.

For the h-th attention head in layer ℓ, write

$$b_{j,\ell,h} = \sum_{w\le j} \alpha_{j,w}^{(\ell,h)} y_w^{(\ell-1)} V_{\ell,h}, \qquad \alpha_{j,w}^{(\ell,h)} = \frac{\exp(a_{j,w}^{(\ell,h)})}{\sum_{t\le j} \exp(a_{j,t}^{(\ell,h)})},$$

where

$$a_{j,w}^{(\ell,h)} = y_j^{(\ell-1)} Q_{\ell,h} A_{j-w} K_{\ell,h} (y_w^{(\ell-1)})^\top.$$

Since the attention weights are nonnegative and sum to one, $\|b_{j,\ell,h}\|_2 \le \|V_{\ell,h}\| \max_{w\le j} \|y_w^{(\ell-1)}\|_2$. Using the Lipschitz continuity of FNN_ℓ layer by layer, there exists $M > 0$, depending only on $\mathcal{F}$, such that $\|y_j^{(\ell)}\|_2, \|\tilde{y}_j^{(\ell)}\|_2 \le M$ for every j and $0 \le \ell \le L$. In particular, $\|\Delta y_i^{(\ell)}\|_2 \le 2M$.

Since $\sup_{r\in\mathbb{Z}} \|A_r\| < \infty$, there exists $S > 0$, depending only on $\mathcal{F}$, such that $|a_{j,w}^{(\ell,h)}|, |\tilde{a}_{j,w}^{(\ell,h)}| \le S$. Therefore, for some constant $C_0 > 0$,

$$\alpha_{j,w}^{(\ell,h)} \le \frac{C_0}{j}, \qquad \tilde{\alpha}_{j,w}^{(\ell,h)} \le \frac{C_0}{j}. \tag{10}$$

By adding and subtracting the corresponding mixed term, there exists $L_a > 0$, depending only on $\mathcal{F}$, such that

$$\left| a_{j,w}^{(\ell,h)} - \tilde{a}_{j,w}^{(\ell,h)} \right| \le L_a \left(\|\Delta y_j^{(\ell-1)}\|_2 + \|\Delta y_w^{(\ell-1)}\|_2 \right). \tag{11}$$

For the softmax map, write $\alpha = \operatorname{softmax}(a)$ and $\tilde{\alpha} = \operatorname{softmax}(\tilde{a})$. Since $\partial\alpha_u/\partial a_t = \alpha_u(\delta_{ut} - \alpha_t)$, we have $\sum_{u\le j} |\partial\alpha_u/\partial a_t| = 2\alpha_t(1 - \alpha_t) \le 2\alpha_t$. Every point on the line segment joining a and $\tilde{a}$ also lies in $[-S, S]^j$, so the bound in equation 10 holds along this segment. Hence, by the integral form of the mean-value theorem, there exists $C_s > 0$ such that

$$\sum_{u\le j} |\alpha_u - \tilde{\alpha}_u| \le \frac{C_s}{j} \sum_{t\le j} |a_t - \tilde{a}_t|. \tag{12}$$

Step 2: Base case $\ell = 1$.

Fix $j > i$ and an attention head h. Write

$$b_{j,1,h} - \tilde{b}_{j,1,h} = I_{j,1,h} + II_{j,1,h},$$

where

$$I_{j,1,h} := \sum_{w \le j} \alpha_{j,w}^{(1,h)} \Delta y_w^{(0)} V_{1,h}, \qquad II_{j,1,h} := \sum_{w \le j} \big(\alpha_{j,w}^{(1,h)} - \tilde{\alpha}_{j,w}^{(1,h)}\big) \tilde{y}_w^{(0)} V_{1,h}.$$

Since the layer-0 perturbation occurs only at position i, equation 10 gives $\|I_{j,1,h}\|_2 \le C/j$ for some constant $C > 0$. Moreover, $\Delta y_j^{(0)} = 0$ for $j > i$, so equation 11 gives $\sum_{w \le j} |a_{j,w}^{(1,h)} - \tilde{a}_{j,w}^{(1,h)}| \le L_a D_0$. Applying equation 12 and the uniform boundedness of the hidden states yields $\|II_{j,1,h}\|_2 \le C/j$. Hence, for some $C' > 0$,

$$\|b_{j,1,h} - \tilde{b}_{j,1,h}\|_2 \le \frac{C'}{j}.$$

For each layer ℓ, define

$$f^{(\ell)}(u, b_1, \ldots, b_H) := u + \sum_{h=1}^{H} b_h + \mathrm{FNN}_\ell \left(u + \sum_{h=1}^{H} b_h \right).$$

By the Lipschitz continuity of FNN_ℓ, the map $f^{(\ell)}$ is Lipschitz. Let $L_{\mathrm{act},\ell}$ be a Lipschitz constant. Since $\Delta y_j^{(0)} = 0$ for $j > i$, the preceding bound implies $\|\Delta y_j^{(1)}\|_2 \le C_1/j$ for some $C_1 > 0$. Thus equation 9 holds for $\ell = 1$.

Step 3: Induction step.

Suppose $\ell \ge 2$ and assume that equation 9 holds at layer $\ell - 1$, namely,

$$\|\Delta y_w^{(\ell-1)}\|_2 \le C_{\ell-1} \frac{(\ln w)^{\ell-2}}{w}, \qquad w > i.$$

Fix $j > i$ and an attention head h, and again write $b_{j,\ell,h} - \tilde{b}_{j,\ell,h} = I_{j,\ell,h} + II_{j,\ell,h}$, where

$$I_{j,\ell,h} := \sum_{w \le j} \alpha_{j,w}^{(\ell,h)} \Delta y_w^{(\ell-1)} V_{\ell,h}, \qquad II_{j,\ell,h} := \sum_{w \le j} \big(\alpha_{j,w}^{(\ell,h)} - \tilde{\alpha}_{j,w}^{(\ell,h)}\big) \tilde{y}_w^{(\ell-1)} V_{\ell,h}.$$

By causality, $\Delta y_w^{(\ell-1)} = 0$ for $w < i$. Using equation 10, the induction hypothesis, and $\|\Delta y_i^{(\ell-1)}\|_2 \le 2M$,

$$\|I_{j,\ell,h}\|_2 \le \frac{C}{j} \left(1 + \sum_{w=i+1}^{j} \frac{(\ln w)^{\ell-2}}{w} \right) \le C \frac{(\ln j)^{\ell-1}}{j},$$

where the last inequality follows from integral comparison.

For the second term, equation 11 gives

$$\sum_{w \le j} \left| a_{j,w}^{(\ell,h)} - \tilde{a}_{j,w}^{(\ell,h)} \right| \le L_a \left(j \|\Delta y_j^{(\ell-1)}\|_2 + \sum_{w \le j} \|\Delta y_w^{(\ell-1)}\|_2 \right).$$

By the induction hypothesis, $j\|\Delta y_j^{(\ell-1)}\|_2 \le C_{\ell-1} (\ln j)^{\ell-2}$, while

$$\sum_{w \le j} \|\Delta y_w^{(\ell-1)}\|_2 \le 2M + C_{\ell-1} \sum_{w=i+1}^{j} \frac{(\ln w)^{\ell-2}}{w} = O\big((\ln j)^{\ell-1}\big).$$

Therefore,

$$\sum_{w \le j} \left| a_{j,w}^{(\ell,h)} - \tilde{a}_{j,w}^{(\ell,h)} \right| = O\big((\ln j)^{\ell-1}\big).$$

Applying equation 12 and the uniform boundedness of the hidden states gives

$$\|II_{j,\ell,h}\|_2 \le C\frac{(\ln j)^{\ell-1}}{j}.$$

Combining the two estimates,

$$\|b_{j,\ell,h} - \tilde{b}_{j,\ell,h}\|_2 \le C\frac{(\ln j)^{\ell-1}}{j}.$$

Finally, the Lipschitz continuity of $f^{(\ell)}$ gives

$$\|\Delta y_j^{(\ell)}\|_2 \le L_{\mathrm{act},\ell}\left(\|\Delta y_j^{(\ell-1)}\|_2 + \sum_{h=1}^{H}\|b_{j,\ell,h} - \tilde{b}_{j,\ell,h}\|_2\right).$$

Using the induction hypothesis and the preceding attention bound, and absorbing both terms into a new constant $C_\ell > 0$, we obtain

$$\|\Delta y_j^{(\ell)}\|_2 \le C_\ell\frac{(\ln j)^{\ell-1}}{j}.$$

This completes the induction and proves equation 9.

Step 4: Output-layer bound.

Taking $\ell = L$ and $j = n$ in equation 9,

$$\|\Delta y_n^{(L)}\|_2 \le C_L\frac{(\ln n)^{L-1}}{n}.$$

Since $\mathcal{F}(x) = y_n^{(L)}W_0 + c$ and $\mathcal{F}(x') = \tilde{y}_n^{(L)}W_0 + c$, we obtain

$$\|\mathcal{F}(x) - \mathcal{F}(x')\|_2 \le \|W_0\|C_L\frac{(\ln n)^{L-1}}{n}.$$

All constants above depend only on the fixed transformer $\mathcal{F}$. Setting $C_{\mathcal{F}} := \|W_0\|C_L$ and taking the maximum over $x' \in B(x,1)$ gives

$$\mathrm{sen}(\mathcal{F},x) \le C_{\mathcal{F}}\frac{\ln^{L-1}(\mathrm{len}(x))}{\mathrm{len}(x)}.$$

This completes the proof. □

D.3 Proof of Theorem 4.4

We repeat Theorem 4.4 below.

Theorem D.8. *For any L, there exists a transformer $\mathcal{F}$ of depth L and a constant $c_{\mathcal{F}} > 0$ such that, for any $M \in \mathbb{N}_+$, there is an input x with $\mathrm{len}(x) \ge M$ satisfying*

$$\mathrm{sen}(\mathcal{F},x) \ge c_{\mathcal{F}}\frac{\ln^{L-1}(\mathrm{len}(x))}{\mathrm{len}(x)}.$$

Proof of Theorem 4.4. We construct a Transformer whose sensitivity matches the upper bound in Theorem 4.2 up to a constant factor.

Step 1: Construct the Transformer and derive the recurrence.

Let L denote the prescribed depth. Construct a depth-L Transformer F with one attention head per layer, hidden dimension 2, and zero feedforward networks. Let $e = (1,-1)$, choose distinct $a, b \in \Sigma$, and set $v_a = \eta e$ and $v_b = 0$ for some fixed $\eta > 0$. In every layer, take $Q = K = 0$ and $V = I_2$. Hence all causal attention weights are uniform: $\alpha_{j,w} = 1/j$ for $1 \le w \le j$.

For $n \geq 2$, let $x_n = (a, b, \ldots, b)$ and $x'_n = (b, b, \ldots, b)$. Then $x'_n \in B(x_n, 1)$ and $F(x'_n) = 0$.

For x_n, every hidden state remains in the span of e, so write $y_j^{(\ell)} = m_j^{(\ell)} e$. Initially, $m_1^{(0)} = \eta$ and $m_j^{(0)} = 0$ for $j > 1$. The uniform attention and residual connection give

$$m_j^{(\ell)} = m_j^{(\ell-1)} + \frac{1}{j} \sum_{w=1}^{j} m_w^{(\ell-1)}, \qquad 1 \leq \ell \leq L. \tag{13}$$

In particular, all $m_j^{(\ell)}$ are nonnegative.

Step 2: Establish the layerwise lower bound.

We prove by induction that, for every $1 \leq \ell \leq L$, there exist $c_\ell > 0$ and N_ℓ such that

$$m_j^{(\ell)} \geq c_\ell \frac{(\ln j)^{\ell-1}}{j}, \qquad j \geq N_\ell. \tag{14}$$

For $\ell = 1$, equation equation 13 gives $m_j^{(1)} = \eta/j$ for $j > 1$, so the claim holds with $c_1 = \eta$.

Suppose it holds at layer $\ell - 1$, where $2 \leq \ell \leq L$. Enlarging $N_{\ell-1}$ if necessary, we may assume that $t \mapsto (\ln t)^{\ell-2}/t$ is decreasing on $[N_{\ell-1}, \infty)$. Since all coefficients are nonnegative,

$$m_j^{(\ell)} \geq \frac{c_{\ell-1}}{j} \sum_{w=N_{\ell-1}}^{j} \frac{(\ln w)^{\ell-2}}{w}.$$

For all sufficiently large j, integral comparison gives

$$\sum_{w=N_{\ell-1}}^{j} \frac{(\ln w)^{\ell-2}}{w} \geq \frac{(\ln j)^{\ell-1} - (\ln N_{\ell-1})^{\ell-1}}{\ell - 1} \geq \frac{(\ln j)^{\ell-1}}{2(\ell-1)}.$$

Hence

$$m_j^{(\ell)} \geq \frac{c_{\ell-1}}{2(\ell-1)} \frac{(\ln j)^{\ell-1}}{j},$$

which proves the induction claim.

Step 3: Conclude the sensitivity lower bound.

Taking $\ell = L$ and $j = n$ in equation 14, for all sufficiently large n,

$$m_n^{(L)} \geq c_D \frac{(\ln n)^{L-1}}{n}.$$

Choose the linear output map $u = \frac{1}{2}(1, -1)$, so that $\langle me, u \rangle = m$. Thus $F(x_n) = m_n^{(L)}$ and $F(x'_n) = 0$, and therefore

$$\operatorname{sen}(F, x_n) \geq \|F(x_n) - F(x'_n)\|_2 \geq c_D \frac{(\ln n)^{L-1}}{n}.$$

Now let $M \in \mathbb{N}_+$ be arbitrary. Choosing $n \geq M$ sufficiently large and setting $x = x_n$, we obtain

$$\operatorname{sen}(F, x) \geq c_F \frac{\ln^{L-1}(\operatorname{len}(x))}{\operatorname{len}(x)},$$

where $c_F := c_D > 0$ depends only on the constructed Transformer F. Thus the upper bound in Theorem 4.2 is tight up to a multiplicative constant. □

E Incompleteness of Infinite-Precision Transformers under Positive Confidence

In this section, we will prove Theorem 4.5. We first repeat the theorem below.

Theorem E.1. *A non-tame Turing machine $\mathbb{T}$ cannot be memorized by any fixed transformer $\mathcal{F}$ with any given confidence $c > 0$.*

Proof. Suppose, toward a contradiction, that there exists a fixed transformer F of depth L that memorizes T with confidence $c > 0$.

Step 1: Construct two nearby inputs with different outputs.

Since T is non-tame, there exist constants $C > 0$ and $0 \le \alpha < 1$ such that T has infinitely many (C, α)-sensitive inputs. Since the input alphabet Σ_1 is finite, there are only finitely many strings of bounded length. Hence, we may choose a sequence of sensitive stoppable inputs $(x_n)_{n\ge1}$ such that $\ell_n := \mathrm{len}(x_n) \to \infty$. By the definition of (C, α)-sensitivity, for every n there exists another stoppable input $x'_n \in B(x_n, \lceil C\ell_n^\alpha \rceil)$ such that $T(x_n) \neq T(x'_n)$. Let $K_n := \lceil C\ell_n^\alpha \rceil$. Then $K_n \le (C+1)\ell_n^\alpha$.

Let $y_n := \widehat{F}(x_n) \oplus \sigma_0$ and $y'_n := \widehat{F}(x'_n) \oplus \sigma_0$, where σ_0 is the end-of-output token. Since F memorizes T, we have $\widehat{F}_{\mathrm{out}}(x_n) = T(x_n)$ and $\widehat{F}_{\mathrm{out}}(x'_n) = T(x'_n)$. Hence $y_n \neq y'_n$.

Let $j_n := \min\{j : y_n[j] \neq y'_n[j]\}$ be the first position at which the two complete autoregressive generations disagree, and let $z_n := y_n[1 : j_n - 1] = y'_n[1 : j_n - 1]$ be their common prefix. Write $a_n := y_n[j_n]$ and $b_n := y'_n[j_n]$, so that $a_n \neq b_n$. At least one of a_n and b_n is not σ_0. Exchanging the roles of x_n and x'_n if necessary, we may assume $a_n \neq \sigma_0$.

Consider the two autoregressive states $s_n := x_n \oplus z_n$ and $s'_n := x'_n \oplus z_n$. They have the same length $N_n := \mathrm{len}(s_n) = \mathrm{len}(s'_n) \ge \ell_n$. Since appending the same prefix z_n introduces no new disagreement, we also have $s'_n \in B(s_n, K_n)$.

Step 2: Derive a contradiction from sensitivity and confidence.

By Corollary 4.3,

$$\|F(s_n) - F(s'_n)\|_2 \le K_n C_F \frac{(\ln N_n)^{L-1}}{N_n},$$

where C_F depends only on the fixed transformer F. For all sufficiently large t, the function $t \mapsto (\ln t)^{L-1}/t$ is decreasing. Since $N_n \ge \ell_n$ and $K_n \le (C+1)\ell_n^\alpha$, for all sufficiently large n,

$$\|F(s_n) - F(s'_n)\|_2 \le (C+1) C_F \frac{(\ln \ell_n)^{L-1}}{\ell_n^{1-\alpha}} \longrightarrow 0,$$

where the convergence follows from $\alpha < 1$.

We now use the positive-confidence assumption. By construction, a_n is the next token generated by F at state s_n. Since $a_n \neq \sigma_0$ and F memorizes T with confidence c,

$$F_{a_n}(s_n) - F_{b_n}(s_n) > c.$$

On the other hand, b_n is the token selected at state s'_n, so $F_{b_n}(s'_n) \ge F_{a_n}(s'_n)$. Therefore,

$$\big(F_{a_n}(s_n) - F_{a_n}(s'_n)\big) + \big(F_{b_n}(s'_n) - F_{b_n}(s_n)\big) > c.$$

Hence at least one of the two terms on the left is larger than $c/2$, and thus

$$\|F(s_n) - F(s'_n)\|_2 > \frac{c}{2}.$$

This contradicts $\|F(s_n) - F(s'_n)\|_2 \to 0$. Therefore, no fixed transformer F can memorize a non-tame Turing machine T with any fixed confidence $c > 0$. □

F Almost-Everywhere Incompleteness of Transformers

In this section, we will prove Theorem 4.6 and Corollary 4.7.

F.1 Parameter Space of Fixed-Architecture Transformers

Let the tuple of all trainable parameters be denoted by θ:

$$\theta = \Bigg(\underbrace{(V_{\text{emb}}, \{Q_{l,h}, K_{l,h}, V_{l,h}\}_{l,h})}_{\text{Embedding and attention}}, \underbrace{(\{E_{l,1}, E_{l,2}, b_l\}_l, W_{\text{out}}, c)}_{\text{Feedforward and output}}, \underbrace{\left(\{C_i\}_{i=1}^{\lfloor W/2 \rfloor}\right)}_{\text{Position embedding}} \Bigg).$$

Let $T = |\Sigma|$ denote the vocabulary size. The total number of parameters is determined by the depth L, width W, number of attention heads H, and vocabulary size $|\Sigma|$, which we denote by $P(W, H, L, |\Sigma|)$.

For a transformer with a fixed architecture (W, H, L, Σ), the total number of real-valued parameters is

$$P(W, H, L, T) = \underbrace{TW}_{\text{Embedding Layer}} + \underbrace{3LHW^2}_{\text{attention Layers}} + \underbrace{L(2W^2 + W)}_{\text{Feedforward Network}}$$
$$+ \underbrace{W(T+1) + (T+1)}_{\text{Output Layer}} + \underbrace{\left\lfloor \frac{W}{2} \right\rfloor}_{\text{RoPE}}.$$

Equivalently,

$$P(W, H, L, T) = TW + LW(3HW + 2W + 1) + (W + 1)(T + 1) + \lfloor \frac{W}{2} \rfloor.$$

Therefore, the parameter space corresponding to a fixed architecture (W, H, L, Σ) is defined as

$$\Theta_{W,H,L,\Sigma} := \{\theta : \theta \text{ is a parameter set of a transformer with architecture } (W, H, L, \Sigma)\}.$$

Without imposing any additional constraints on parameter magnitudes, this parameter space is naturally isomorphic to a finite-dimensional Euclidean space $\mathbb{R}^{P(W,H,L,T)}$

F.2 Proof of Theorem 4.6

Before proceeding to the formal proof of the theorem, we first need to establish that the parameter set of interest is Borel measurable.

Definition F.1. *Let $G : \Sigma^* \to 2^{\Sigma^*}$ be an f-number of rule, and let $x \in \Sigma^*$ be a stoppable input of $\mathbb{T}$. Define*

$$\Gamma_G^{\mathbb{T}}(x) := \{r \oplus \sigma_T \oplus \mathbb{T}(x) \oplus \sigma_0 : r \in G(x)\},$$

where σ_T is the CoT delimiter and σ_0 is the end-of-output symbol. Thus, for every $\mathcal{F} \in M_G(\mathbb{T})$, since $\mathrm{CoT}(\mathcal{F}(x)) \in G(x)$ and $\mathcal{F}$ memorizes $\mathbb{T}$, the complete generation sequence of $\mathcal{F}$ on x belongs to $\Gamma_G^{\mathbb{T}}(x)$.Then $|\Gamma_G^{\mathbb{T}}(x)| = |G(x)| \le f(\mathrm{len}(x))$, so $\Gamma_G^{\mathbb{T}}(x)$ is finite.

Definition F.2. *For two sequences $x, x' \in \Sigma^*$ of the same length, define*

$$d_{\mathrm{nf}}(x, x') := \#\{j \in [\mathrm{len}(x) - 1] : x[j] \neq x'[j]\},$$

the number of positions at which they differ excluding the final position.

Lemma F.1 (Measurability of the memorization parameter set)**.** *Fix a Turing machine $\mathbb{T}$, a transformer architecture with depth L, width W, number of heads H, and finite vocabulary Σ. Let $\Theta_{L,W,H,\Sigma} \cong \mathbb{R}^P$ be its parameter space. Let $G : \Sigma^* \to 2^{\Sigma^*}$ be an f-bounded CoT rule, i.e., $|G(x)| \le f(\mathrm{len}(x))$ for every stoppable input x of $\mathbb{T}$.*

Assume that generation is performed using a fixed deterministic $\arg\max$ *tie-breaking rule. Then*

$$M_G(\mathbb{T}) \subseteq \Theta_{L,W,H,\Sigma}$$

is a Borel measurable set.

Proof. Fix a finite autoregressive state s and an output token a, and let $L_{s,a}(\theta) := F_\theta(s)_a$ denote the output logit corresponding to token a. For fixed s and a, the map $\theta \mapsto L_{s,a}(\theta)$ is continuous.

Fix a deterministic tie-breaking rule that chooses the token with the smallest index among all tokens attaining the maximal logit. Then the set of parameters for which the next generated token at state s is a is

$$\mathcal{E}_{s,a} := \bigcap_{j<a} \{\theta : L_{s,a}(\theta) - L_{s,j}(\theta) > 0\} \cap \bigcap_{j>a} \{\theta : L_{s,a}(\theta) - L_{s,j}(\theta) \geq 0\}.$$

Hence $\mathcal{E}_{s,a}$ is Borel.

For a fixed complete sequence $y = (y_1, \ldots, y_m) \in \Gamma_G^{\mathbb{T}}(x)$, write $y_{<j} := (y_1, \ldots, y_{j-1})$ and define

$$\mathcal{E}_{x,y} := \bigcap_{j=1}^{m} \mathcal{E}_{x \oplus y_{<j},\, y_j}.$$

Since $m < \infty$, the set $\mathcal{E}_{x,y}$ is Borel.

The set of parameters that generate an admissible complete output on input x is therefore

$$\mathcal{E}_x^G := \bigcup_{y \in \Gamma_G^{\mathbb{T}}(x)} \mathcal{E}_{x,y}.$$

Since $\Gamma_G^{\mathbb{T}}(x)$ is finite, $\mathcal{E}_x^G$ is a finite union of Borel sets and hence is Borel.

Finally,

$$M_G(\mathbb{T}) = \bigcap_{x \in \mathcal{H}_{\mathbb{T}}} \mathcal{E}_x^G,$$

where $\mathcal{H}_{\mathbb{T}}$ denotes the set of stoppable inputs of $\mathbb{T}$. Since $\mathcal{H}_{\mathbb{T}} \subseteq \Sigma_1^*$ is countable, $M_G(\mathbb{T})$ is a countable intersection of Borel sets. Therefore $M_G(\mathbb{T})$ is Borel measurable. □

For convenience, we first restate Theorem 4.6.

Theorem F.2. *For arbitrary S, L, W, H, Σ, assume that the following conditions are satisfied.*

(1) $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$ is a (C, α) non-tame Turing machine and $\Sigma_2 \subset \Sigma_0$.

(2) $\mathbf{A}(S, L, W, H, \Sigma)$ is an absolutely continuous distribution.

(3) G is a f-number of rules, and f satisfies $\lim_{n\to\infty} \frac{f(n)(\ln n)^{(L-1)/2}}{\sqrt{n^{1-\alpha}}} = 0$.

Then, $\Pr_{\mathcal{F}\sim\mathbf{A}(S,L,W,H,\Sigma)}(\mathcal{F} \in M_G(\mathbb{T})) = 0$. That is, if fewer than $\widetilde{o}(\sqrt{n^{1-\alpha}})$ CoTs are allowed for inputs of length n, then the probability for $\mathcal{F}$ trained by algorithm $\mathbf{A}$ to memorize $\mathbb{T}$ is zero.

Proof. Let $\Theta_{L,W,H,\Sigma} \cong \mathbb{R}^P$ be the parameter space of the fixed transformer architecture, equipped with the P-dimensional Lebesgue measure λ_P. By Lemma F.1, $M_G(\mathbb{T})$ is Borel measurable.

We first prove $\lambda_P(M_G(\mathbb{T})) = 0$, and then use the absolute continuity of the randomized training distribution.

Step 1: Choose a sequence of sensitive input pairs.

Since $\mathbb{T}$ is non-tame with coefficients $C > 0$ and $0 \leq \alpha < 1$, it has infinitely many (C, α)-sensitive stoppable inputs. Since the input alphabet Σ_1 is finite, there are only finitely many strings whose lengths are bounded by any fixed constant. Therefore, we may choose a sequence of sensitive inputs $(x_n)_{n\geq 1}$ such that$\ell_n := \mathrm{len}(x_n) \to \infty$.

By the definition of (C, α)-sensitivity, for every n there exists another stoppable input $x_n' \in B(x_n, \lceil C\ell_n^\alpha \rceil)$ such that $\mathbb{T}(x_n) \neq \mathbb{T}(x_n')$.

By the definition of $B(x, k)$, $\mathrm{len}(x_n') = \mathrm{len}(x_n) = \ell_n$, and $d_{\mathrm{nf}}(x_n, x_n') \leq (C+1)\ell_n^\alpha$.

Step 2: Fix all parameters except the output bias.

Let K denote the size of the output vocabulary. Separate the output bias from all remaining transformer parameters and write $\theta = (\vartheta, c) \in \mathbb{R}^{P-K} \times \mathbb{R}^K$. Then, for every finite autoregressive state s, $F_{\vartheta,c}(s) = G_\vartheta(s) + c$, where $G_\vartheta(s)$ does not depend on c.

Fix $\vartheta \in \mathbb{R}^{P-K}$ and define the corresponding output-bias section

$$M_{G,\vartheta}(\mathbb{T}) := \left\{c \in \mathbb{R}^K : (\vartheta, c) \in M_G(\mathbb{T})\right\}.$$

We shall prove that $\lambda_K(M_{G,\vartheta}(\mathbb{T})) = 0$.

Step 3: Every pair of admissible CoTs produces a thin slab.

Fix n and take arbitrary outputs $y \in \Gamma_G^{\mathbb{T}}(x_n)$ and $y' \in \Gamma_G^{\mathbb{T}}(x'_n)$. Since the outputs of $\mathbb{T}$ on x_n and x'_n are different we have $y \neq y'$.

Let $j(y, y') := \min\{j : y[j] \neq y'[j]\}$ be their first disagreement position, and let $z(y, y') := y[1 : j(y, y') - 1] = y'[1 : j(y, y') - 1]$ be the common prefix preceding their first disagreement. Write $a(y, y') := y[j(y, y')]$ and $b(y, y') := y'[j(y, y')]$, so that $a(y, y') \neq b(y, y')$.

Define$s_{n,y,y'} := x_n \oplus z(y, y')$, $s'_{n,y,y'} := x'_n \oplus z(y, y')$, and let $N_{n,y,y'} := \mathrm{len}(s_{n,y,y'})$. Then

$$N_{n,y,y'} = \mathrm{len}(s'_{n,y,y'}) \geq \ell_n. \tag{15}$$

Appending the same prefix to x_n and x'_n does not introduce any new disagreement. Hence $d_{\mathrm{nf}}(s_{n,y,y'}, s'_{n,y,y'}) \leq (C+1)\ell_n^\alpha$. Let $k_n := d_{\mathrm{nf}}(x_n, x'_n) \leq (C+1)\ell_n^\alpha$. We may construct a sequence $s^{(0)} = s_{n,y,y'}, s^{(1)}, \ldots, s^{(k_n)} = s'_{n,y,y'}$ such that every consecutive pair differs at exactly one non-final position.

Applying the single-token sensitivity bound to the transformer with parameters $(\vartheta, 0)$, and by Corollary 4.3,

$$\left\|G_\vartheta(s_{n,y,y'}) - G_\vartheta(s'_{n,y,y'})\right\|_2 \leq C_{(\vartheta,0)} k_n \frac{(\ln N_{n,y,y'})^{L-1}}{N_{n,y,y'}}.$$

Since $k_n \leq (C+1)\ell_n^\alpha$ and the function $t \mapsto (\ln t)^{L-1}/t$ is decreasing for all sufficiently large t, it follows from equation 15 that, for all sufficiently large n,

$$\left\|G_\vartheta(s_{n,y,y'}) - G_\vartheta(s'_{n,y,y'})\right\|_2 \leq (C+1) C_{(\vartheta,0)} \frac{(\ln \ell_n)^{L-1}}{\ell_n^{1-\alpha}}. \tag{16}$$

Now suppose that an output bias c makes the transformer generate y on x_n and y' on x'_n. For simplicity, write $a := a(y, y')$ and $b := b(y, y')$.

At state $s_{n,y,y'}$, the next generated token must be a, so

$$c_a - c_b \geq G_\vartheta(s_{n,y,y'})_b - G_\vartheta(s_{n,y,y'})_a.$$

Similarly, at state $s'_{n,y,y'}$, the next generated token must be b, so

$$c_a - c_b \leq G_\vartheta(s'_{n,y,y'})_b - G_\vartheta(s'_{n,y,y'})_a.$$

Thus $c_a - c_b \in I_{n,y,y'}(\vartheta)$, where

$$I_{n,y,y'}(\vartheta) := \left[G_\vartheta(s_{n,y,y'})_b - G_\vartheta(s_{n,y,y'})_a,\ G_\vartheta(s'_{n,y,y'})_b - G_\vartheta(s'_{n,y,y'})_a\right].$$

If the left endpoint is larger than the right endpoint, then no output bias c can generate this pair of CoTs, and the corresponding feasible set is empty.

Whenever $I_{n,y,y'}(\vartheta)$ is nonempty,

$$\begin{aligned} |I_{n,y,y'}(\vartheta)| &\leq \left|G_\vartheta(s'_{n,y,y'})_b - G_\vartheta(s_{n,y,y'})_b\right| \\ &\quad + \left|G_\vartheta(s'_{n,y,y'})_a - G_\vartheta(s_{n,y,y'})_a\right| \\ &\leq 2\left\|G_\vartheta(s_{n,y,y'}) - G_\vartheta(s'_{n,y,y'})\right\|_2. \end{aligned} \tag{17}$$

Combining equation 16 and equation 17, we obtain

$$|I_{n,y,y'}(\vartheta)| \leq 2(C+1) C_{(\vartheta,0)} \frac{(\ln \ell_n)^{L-1}}{\ell_n^{1-\alpha}} \tag{18}$$

for all sufficiently large n, uniformly over all admissible pairs (y, y').

Step 4: Cover the feasible section by finitely many thin slabs.

For distinct tokens a, b and an interval $I \subseteq \mathbb{R}$, define $\mathcal{H}_{a,b}(I) := \{c \in \mathbb{R}^K : c_a - c_b \in I\}$.

Let $\mathcal{S}_{n,\vartheta}$ be the set of output biases for which the transformer generates some admissible complete output on both x_n and x'_n. By the first-disagreement argument above,

$$\mathcal{S}_{n,\vartheta} \subseteq \bigcup_{\substack{y \in \Gamma_G^{\mathbb{T}}(x_n) \\ y' \in \Gamma_G^{\mathbb{T}}(x'_n)}} \mathcal{H}_{a(y,y'),b(y,y')}\left(I_{n,y,y'}(\vartheta)\right). \tag{19}$$

Fix $R > 0$ and let $Q_R := [-R, R]^K$. For every $a \neq b$ and every finite interval I, Fubini's theorem gives

$$\lambda_K\left(\mathcal{H}_{a,b}(I) \cap Q_R\right) \leq (2R)^{K-1}|I|. \tag{20}$$

By the f-boundedness of G and $\mathrm{len}(x_n) = \mathrm{len}(x'_n) = \ell_n$,

$$|\Gamma_G^{\mathbb{T}}(x_n)|\,|\Gamma_G^{\mathbb{T}}(x'_n)| \leq f(\ell_n)^2. \tag{21}$$

Combining equation 19, equation 20, equation 18, and equation 21, we obtain, for all sufficiently large n,

$$\begin{aligned}\lambda_K\left(\mathcal{S}_{n,\vartheta} \cap Q_R\right) &\leq 2(C+1)C_{(\vartheta,0)}(2R)^{K-1} f(\ell_n)^2 \frac{(\ln \ell_n)^{L-1}}{\ell_n^{1-\alpha}} \\ &= 2(C+1)C_{(\vartheta,0)}(2R)^{K-1}\left[\frac{f(\ell_n)(\ln \ell_n)^{(L-1)/2}}{\sqrt{\ell_n^{1-\alpha}}}\right]^2. \end{aligned} \tag{22}$$

By assumption, $\lim_{n\to\infty} \frac{f(n)(\ln n)^{(L-1)/2}}{\sqrt{n^{1-\alpha}}} = 0$. Let $\ell_n \to \infty$, then we have $\lambda_K\left(\mathcal{S}_{n,\vartheta} \cap Q_R\right) \to 0$

Every $c \in M_{G,\vartheta}(\mathbb{T})$ must correctly handle both x_n and x'_n for every n. Hence $M_{G,\vartheta}(\mathbb{T}) \subseteq \mathcal{S}_{n,\vartheta}$ for every n. Therefore, for every n, $\lambda_K\left(M_{G,\vartheta}(\mathbb{T}) \cap Q_R\right) \leq \lambda_K\left(\mathcal{S}_{n,\vartheta} \cap Q_R\right)$, which leads to $\lambda_K\left(M_{G,\vartheta}(\mathbb{T}) \cap Q_R\right) = 0$ with $n \to \infty$.

Since $\mathbb{R}^K = \bigcup_{R=1}^{\infty} Q_R$, the countable additivity of λ_K yields

$$\lambda_K\left(M_{G,\vartheta}(\mathbb{T})\right) = 0 \qquad \text{for every } \vartheta \in \mathbb{R}^{P-K}. \tag{23}$$

Step 5: The full memorization parameter set has measure zero.

By Lemma F.1, $M_G(\mathbb{T})$ is Borel measurable. Hence Tonelli's theorem and equation 23 imply

$$\lambda_P(M_G(\mathbb{T})) = \int_{\mathbb{R}^{P-K}} \lambda_K\left(M_{G,\vartheta}(\mathbb{T})\right)\, d\vartheta = 0$$

Let $\mu := \mathbf{A}(S, L, W, H, \Sigma)$ be the probability distribution on transformer parameters produced by the randomized training algorithm. By assumption, $\mu \ll \lambda_P$. Since$\lambda_P(M_G(\mathbb{T})) = 0$, absolute continuity implies $\mu(M_G(\mathbb{T})) = 0$. Therefore,

$$\Pr_{\mathcal{F}\sim\mathbf{A}(S,L,W,H,\Sigma)}\left(\mathcal{F} \in M_G(\mathbb{T})\right) = 0.$$

This completes the proof. □

Remark F.1. *The key idea of the above proof is to bridge the gap between sensitivity and memorization. The sensitivity bound controls how much the logits can change between nearby inputs, but by itself it does not determine whether the generated tokens are different, since the output bias can still affect the argmax decision. We therefore fix all parameters responsible for the sensitivity bound and isolate the additive output bias in the architecture defined in Appendix A.2. For each fixed choice of the remaining parameters, memorizing two nearby inputs with different outputs forces a difference between two bias coordinates to lie in a narrow interval. This reduces the memorization problem to a measure estimate in the output-bias space. Consequently, the present proof relies on the existence of an additive output bias; extending the result to architectures with bias-free output heads requires a separate argument.*

F.3 Proof of Corollary 4.7

We now restate Corollary 4.7 and provide a proof.

Corollary F.3. *For $\mathbb{T}_{\mathbf{Arith}}$, we restrict the generation rules G to performing one calculation at a time in CoT. That is, at each CoT step, the leftmost currently calculable innermost operation is evaluated and replaced by its value in a fixed format (shown in Example 3.1). Then, if $\mathbf{A}(S, L, W, H, \Sigma)$ is an absolutely continuous distribution, then $\Pr_{\mathcal{F}\sim\mathbf{A}(S,L,W,H,\Sigma)}(\mathcal{F} \in M_G(\mathbb{T}_{\mathbf{Arith}})) = 0$.*

Proof. First, note that $\mathbb{T}_{\mathbf{Arith}}$ is non-tame since it contains infinitely many $(1, 0)$-sensitive inputs.

Let $\Theta_{L,W,H,\Sigma} \cong \mathbb{R}^P$ be the parameter space of the fixed Transformer architecture, and let K be the size of the output vocabulary. We first show that $\lambda_P(M_G(\mathbb{T}_{\mathbf{Arith}})) = 0$.

We construct a sequence of arithmetic input pairs for which the admissible CoT is unique. Let $A_1 = 0$ and $A_{n+1} = (A_n + 0)$, and define $x_n = (A_n + 1)$ and $x'_n = (A_n * 1)$. Then x_n and x'_n have the same length, differ at exactly one non-final token, and satisfy $\mathbb{T}_{\mathbf{Arith}}(x_n) = 1$ and $\mathbb{T}_{\mathbf{Arith}}(x'_n) = 0$. Moreover, $\mathrm{len}(x_n) \to \infty$.

Under the generation rule G, each CoT step computes one currently calculable operator. By the nested structure above, at every step, there is exactly one such operator. Hence, both x_n and x'_n admit a unique admissible CoT. Let y_n and y'_n denote the corresponding complete generation sequences. Since their final outputs differ, $y_n \neq y'_n$.

As in the proof above, separate the output bias and write $\theta = (\vartheta, c) \in \mathbb{R}^{P-K} \times \mathbb{R}^K$, so that $F_{\vartheta,c}(s) = G_\vartheta(s) + c$. Fix $\vartheta \in \mathbb{R}^{P-K}$ and let

$$M_{G,\vartheta}(\mathbb{T}_{\mathbf{Arith}}) := \left\{c \in \mathbb{R}^K : (\vartheta, c) \in M_G(\mathbb{T}_{\mathbf{Arith}})\right\}.$$

For each n, let j_n be the first position at which y_n and y'_n differ, let z_n be their common prefix before j_n, and write $a_n := y_n[j_n]$ and $b_n := y'_n[j_n]$. Since the output vocabulary is finite, after passing to an infinite subsequence, we may assume that $a_n = a$ and $b_n = b$ correspond to two fixed tokens $a \neq b$.

Set $s_n := x_n \oplus z_n$ and $s'_n := x'_n \oplus z_n$. These two states still differ at exactly one non-final position and have the same length N_n, where $N_n \geq \mathrm{len}(x_n) \to \infty$. By Theorem 4.2 applied to the Transformer with parameters $(\vartheta, 0)$,

$$\|G_\vartheta(s_n) - G_\vartheta(s'_n)\|_2 \leq C_\vartheta \frac{(\ln N_n)^{L-1}}{N_n} \longrightarrow 0.$$

Now let $c \in M_{G,\vartheta}(\mathbb{T}_{\mathbf{Arith}})$. Since the admissible CoTs are unique, the Transformer must generate y_n on x_n and y'_n on x'_n. At their first disagreement, it must therefore choose a at s_n and b at s'_n. Hence $c_a - c_b \in I_n(\vartheta)$.where

$$I_n(\vartheta) := \left[G_\vartheta(s_n)_b - G_\vartheta(s_n)_a,\, G_\vartheta(s'_n)_b - G_\vartheta(s'_n)_a\right].$$

As in the preceding proof, whenever $I_n(\vartheta)$ is nonempty,

$$|I_n(\vartheta)| \leq 2\,\|G_\vartheta(s_n) - G_\vartheta(s'_n)\|_2 \longrightarrow 0.$$

Thus, for any $c, \widetilde{c} \in M_{G,\vartheta}(\mathbb{T}_{\mathbf{Arith}})$, both $c_a - c_b$ and $\widetilde{c}_a - \widetilde{c}_b$ belong to $I_n(\vartheta)$ for every n in the chosen subsequence. Therefore

$$|(c_a - c_b) - (\widetilde{c}_a - \widetilde{c}_b)| \leq |I_n(\vartheta)| \longrightarrow 0,$$

and hence $c_a - c_b = \widetilde{c}_a - \widetilde{c}_b$.

Consequently, either $M_{G,\vartheta}(\mathbb{T}_{\mathbf{Arith}}) = \varnothing$, or there exists $r_\vartheta \in \mathbb{R}$ such that

$$M_{G,\vartheta}(\mathbb{T}_{\mathbf{Arith}}) \subseteq \left\{c \in \mathbb{R}^K : c_a - c_b = r_\vartheta\right\}.$$

The set on the right is an affine hyperplane in $\mathbb{R}^K$ and hence has K-dimensional Lebesgue measure zero. Thus $\lambda_K(M_{G,\vartheta}(\mathbb{T}_{\mathbf{Arith}})) = 0$ for every $\vartheta \in \mathbb{R}^{P-K}$.

By Lemma F.1, $M_G(\mathbb{T}_{\mathbf{Arith}})$ is Borel measurable. Hence, by Tonelli's theorem,

$$\lambda_P(M_G(\mathbb{T}_{\mathbf{Arith}})) = \int_{\mathbb{R}^{P-K}} \lambda_K(M_{G,\vartheta}(\mathbb{T}_{\mathbf{Arith}}))\, d\vartheta = 0.$$

Finally, since $\mathbf{A}(S, L, W, H, \Sigma) \ll \lambda_P$, absolute continuity gives

$$\Pr_{F\sim\mathbf{A}(S,L,W,H,\Sigma)}(F \in M_G(\mathbb{T}_{\mathbf{Arith}})) = 0.$$

This completes the proof. □

G Turing Completeness of Agents

In this section, we will prove Theorem 5.1 and a version of Theorem 5.1 for agents with one transformer.

G.1 Preliminaries

To prove the theorem, we first introduce several auxiliary functions.

Definition G.1 (Auxiliary functions)**.**

Search function. *For $x, z \in \Sigma^*$ with $|z| \geq 3$, let*

$$i = \operatorname*{argmin}_r\{r : x[r] = z[2]\}, \qquad j = \operatorname*{argmin}_r\{r : x[r] = z[3]\}.$$

We use the convention that $x[r] = B$ whenever $r \leq 0$ or $r \geq |x| + 1$. Define

$$f_{\text{Search}}(x, z) = x[i - 1 : j + 1].$$

If either i or j does not exist, set $f_{\text{Search}}(x, z) = B \oplus x[1]$.

Judge function. *For a fixed symbol $\sigma \in \Sigma$, define*

$$f_{\text{Judge},\sigma}(x) = \begin{cases} 1, & x[|x|] = \sigma, \\ 0, & x[|x|] \neq \sigma. \end{cases}$$

Delete function. *For $x, z \in \Sigma^*$, define*

$$f_{\text{Delete}}(x, z) = x/z,$$

where x/z denotes the sequence obtained from x by deleting every symbol $x[r]$ that coincides with at least one symbol occurring in z.

Replace function. *For $x, y, z \in \Sigma^*$ with $|y| \geq 3$ and $|z| \geq 3$, let*

$$i = \arg\min_r\{r : x[r] = y[2]\}, \qquad j = \arg\min_r\{r : x[r] = y[3]\}.$$

If either index does not exist, define

$$f_{\text{Replace}}(x, y, z) = x.$$

Otherwise, define

$$f_{\text{Replace}}(x, y, z) = x[1 : i - 2] \oplus z[2 : |z| - 1] \oplus x[j + 2 : |x|].$$

Each of the four functions above can be computed in linear time.

Definition G.2. *Let*

$$\mathcal{T} = (Q, \Sigma_1, \Sigma_2, \delta)$$

be a Turing machine. For an input x, let $\mathcal{T}(x)[i]$ denote the finite tape segment recorded after the i-th transition as follows. Let l_1 and r_1 be the leftmost and rightmost non-blank tape positions, and let l_2 and r_2 be the leftmost and rightmost positions ever visited by the head up to that time. Then $\mathcal{T}(x)[i]$ is the tape segment between

$$\min\{l_1, l_2\} \qquad \textit{and} \qquad \max\{r_1, r_2\}.$$

In particular,

$$\mathcal{T}(x)[0] = x.$$

Definition G.3. *For $z \in \mathbb{Z}_+$, define*

$$B_z : \mathbb{Z}_+ \longrightarrow \{-1, 0, 1\}^{\lceil \log_2 z \rceil}$$

as follows. For $x \in \{1, \ldots, z-1\}$, $B_z(x)$ is the binary representation of x with length $\lceil \log_2 z \rceil$. Moreover,

$$B_z(z) = -\mathbf{1}_{\lceil \log_2 z \rceil},$$

and for all remaining values of x we set

$$B_z(x) = \mathbf{0}_{\lceil \log_2 z \rceil}.$$

Definition G.4. *Assume that*

$$\Sigma_2 \subset \Sigma \setminus \{\sigma_b, \sigma_e, 0, 1, -1\}.$$

Let $\mathcal{T}(x)[i]$ be as in Definition G.2. Suppose that, after the i-th transition, the machine is in state q_k and its head is at position j of $\mathcal{T}(x)[i]$. Then define

$$\begin{aligned} \mathcal{T}'(x)[i] = \mathcal{T}(x)[i][1 : j-1] \oplus \sigma_b \oplus B_{|Q|}(k) \oplus \sigma_e \\ \oplus \mathcal{T}(x)[i][j : |\mathcal{T}(x)[i]|]. \end{aligned} \tag{24}$$

If $j = 0$, we use the convention

$$\mathcal{T}(x)[i][-1] = B.$$

Without loss of generality, let $\sigma_{|\Sigma_2|} = B$.

Thus $\mathcal{T}'(x)[i]$ explicitly records both the head position and the current state.

We also use the following notation.

Definition G.5. *For indices $i_1, \ldots, i_r$, let*

$$I_{i_1, \ldots, i_r}$$

denote the vector whose coordinates $i_1, \ldots, i_r$ are equal to 1 and whose remaining coordinates are 0.

Similarly,

$$I_{(i_1, j_1), \ldots, (i_r, j_r)}$$

denotes the matrix whose entries at $(i_1, j_1), \ldots, (i_r, j_r)$ are equal to 1 and whose remaining entries are 0.

We now establish two auxiliary lemmas.

Lemma G.1. *Let $\Sigma = \{\sigma_i\}_{i=1}^T$. For any $k, j \in \{1, \ldots, T\}$ and any $q \geq 6$, there exists a Transformer $\mathcal{F}$ of width 3, depth 2, and one attention head such that the following holds for every $x \in \Sigma^*$. If $1 \leq \#\{r : x[r] = \sigma_k\} \leq \lfloor e^q \rfloor$, then $\mathcal{F}(x) = I_j$. If σ_k does not occur in x, then $\mathcal{F}(x) = 0$.*

Proof. We construct the Transformer explicitly. For the embedding layer, embed each symbol σ_i as

$$v_i = \begin{cases} (0, 1, q), & i = k, \\ (0, 0, q), & i \neq k. \end{cases}$$

In the first hidden layer, choose $A_{m-n} = I$, $QK = qI_{(3,2)}$, $V = I_{(2,1)}$, and $\mathrm{FNN} = 0$. Then

$$v_i QK v_r^\top = \begin{cases} q^2, & \sigma_r = \sigma_k, \\ 0, & \sigma_r \neq \sigma_k. \end{cases}$$

Suppose that σ_k appears exactly m times, where $1 \leq m \leq \lfloor e^q \rfloor$. After subtracting the maximum attention score, the exponentiated score is 1 at positions containing σ_k and e^{-q^2} elsewhere. Since $q \geq 6$, $[e^{-q^2}]_q = 0$. Hence only the m occurrences of σ_k receive nonzero attention weight.

Let $\beta_x := m[1/m]_q$. Since $|[1/m]_q - 1/m| \leq \frac{1}{2} 10^{-q}$, we obtain

$$|\beta_x - 1| \leq \frac{m}{2} 10^{-q} \leq \frac{1}{2} \left(\frac{e}{10}\right)^q \leq \frac{1}{4}.$$

Therefore $\beta_x \in [\frac{3}{4}, \frac{5}{4}]$.

Hence, the first coordinate of the hidden state at the final position satisfies

$$h_1(x)[1] = \begin{cases} \beta_x, & \sigma_k \text{ occurs in } x, \\ 0, & \text{otherwise.} \end{cases}$$

Finally, choose the FNN $\phi(t) = 2\,\mathrm{ReLU}(t - \frac{1}{4}) - 2\,\mathrm{ReLU}(t - \frac{3}{4}) - \mathrm{ReLU}(t)$. Then $\phi(0) = 0$ and $\phi(\beta) = 1$ for $\beta \in [\frac{3}{4}, \frac{5}{4}]$. Choosing all output coordinates except j to be zero therefore gives $\mathcal{F}(x)[j] = 1$. Hence

$$\mathcal{F}(x) = \begin{cases} I_j, & 1 \le \#\{r : x[r] = \sigma_k\} \le \lfloor e^q \rfloor, \\ 0, & \sigma_k \text{ does not occur.} \end{cases}$$

This proves the lemma. □

Next, an important lemma will be presented for mapping a symbol back to its index within the alphabet.

Lemma G.2. *Let $L \ge 2$, let $|\Sigma|$ be the vocabulary size, and set $P = 10^{\lceil \ln_{10}(16L) \rceil}$, $\omega = P^{-1}$, and $A = 64(q+1)P^2$. Suppose $q \ge 6$ and*

$$10^q \ge 10^4 (q+1) P^2. \tag{25}$$

There is one attention layer of width $O(L|\Sigma|)$ with L heads such that, for every $1 \le n = |x| \le L$, its final-position hidden state contains blocks $V_0, \dots, V_{L-1}$ satisfying

$$V_h = e_{x[n - \min\{h, n-1\}]}. \tag{26}$$

In particular, $V_h = e_{x[n-h]}$ for every $0 \le h \le n-1$. For inputs of arbitrary length, the same layer also satisfies

$$(V_h)_i \ge 0, \qquad \|V_h\|_1 \le 2, \qquad 0 \le h \le n-1. \tag{27}$$

Proof. For the embedding layer, embed each token $a \in \Sigma$ as $v_a = (1, e_a, 0, \dots, 0)$, where $e_a \in \mathbb{R}^K$ is its one-hot representation. For each attention head, reserve a disjoint block of K coordinates in the hidden state, initialized to zero.

For the positional encoding, choose a two-dimensional RoPE plane with frequency $\omega = P^{-1}$ and set all other required frequencies to zero. Define $c_u = \cos_q(u\omega)$ and $s_u = \sin_q(u\omega)$ for $u \ge 0$.

The attention layer contains L heads. For the h-th head, $0 \le h \le L-1$, choose the query corresponding to the constant coordinate as $(Ac_h, -As_h)$ and the key as $(1, 0)$. Then, under the RoPE convention used in this paper, the attention score for a token at relative distance u from the final position is

$$\lambda_h(u) = \big[[Ac_h c_u]_q + [As_h s_u]_q\big]_q.$$

The value matrix of the h-th head copies the one-hot representation of the selected token into the h-th reserved block.

First consider an input $x = (x[1], \dots, x[n])$ with $n \le L$. Since $L/P \le 1/16$, there is no phase wrap for $0 \le h, u < L$. Let $\eta = 10^{-q}$. By the q-precision rounding of the trigonometric values and the two multiplications,

$$\left| \lambda_h(u) - A \cos\left(\frac{u-h}{P}\right) \right| \le 4A\eta.$$

For the h-th head, the ideal score $A\cos((u-h)/P)$ is maximized at $u_* = \min\{h, n-1\}$. We show that this maximum is sufficiently separated from every other visible position.

If $u_* = h$, then $|(u-h)/P| \ge 1/P$ for every $u \ne h$. Since $1 - \cos t \ge t^2/4$ for $|t| \le 1$, the ideal score gap is at least $A/(4P^2)$.

If $u_* = n - 1 < h$, the maximum is attained at the boundary of the visible positions. Writing $a = (h - (n-1))/P$ and using that all relevant phases lie in $[0, 1]$,

$$\cos a - \cos(a + P^{-1}) = \int_a^{a+P^{-1}} \sin t \, dt \ge \frac{1}{4P^2}.$$

Hence the ideal score gap is again at least $A/(4P^2)$.

Taking the rounding errors into account, the computed score gap between the maximal position and any competing position is at least

$$\frac{A}{4P^2} - 8A10^{-q}.$$

By the definition $A = 64(q+1)P^2$ and assumption equation 25,

$$\begin{aligned}\frac{A}{4P^2} - 8A10^{-q} &\geq 16(q+1) - \frac{512}{10^4} \\ &\geq 15(q+1) > q\ln 10 + \ln 2.\end{aligned}$$

Therefore, after subtracting the maximal attention score, every nonmaximal exponential satisfies $[e^{-\Delta}]_q = 0$, where Δ is the corresponding score gap.

The exponential corresponding to the maximal score is exactly one, so the softmax denominator is also one. Thus the selected position receives attention weight exactly one and all other positions receive weight zero. Hence the h-th head exactly copies the token at relative position $u_* = \min\{h, n-1\}$. Since the corresponding destination block is zero in the embedding layer, the residual connection does not affect it. Therefore

$$V_h = e_{x[n-\min\{h,n-1\}]}.$$

In particular, $V_h = e_{x[n-h]}$ for every $0 \leq h \leq n-1$, proving equation 26.

It remains to consider inputs whose lengths may exceed L. Exact copying is no longer required, but the attention computation must remain well defined.

For an arbitrary relative position u, instead of directly computing the possibly unbounded product $u\omega$, set $r_0 = 0$ and recursively define

$$r_{u+1} = \begin{cases} r_u + \omega, & r_u + \omega < 2[\pi]_q, \\ r_u + \omega - 2[\pi]_q, & r_u + \omega \geq 2[\pi]_q. \end{cases}$$

Since both ω and $2[\pi]_q$ are q-precision numbers, every r_u remains a q-precision number and satisfies $0 \leq r_u < 2[\pi]_q < 7$. This gives the same reduced phase as in Definition C.2 without forming an unbounded intermediate value. Negative relative positions are handled analogously by modular negation. Hence all trigonometric values and attention scores remain in the permitted numerical range.

Finally, consider an arbitrary attention row. Let E_i denote its rounded exponentials and set $Z = [\sum_i E_i]_q$. At least one E_i equals one. If $Z = \mathrm{Inf}$, then by the finite-precision division convention all attention weights are zero, so the corresponding output block is zero.

Otherwise, Z is finite. Let $p_i = E_i/Z$, so that $\sum_i p_i = 1$. If $[p_i]_q > 0$, then $p_i \geq \eta/2$, and therefore

$$[p_i]_q \leq p_i + \frac{\eta}{2} \leq 2p_i.$$

Consequently, $\sum_i [p_i]_q \leq 2$. Since the value vectors are one-hot vectors, all coordinates of the head output are nonnegative and its ℓ_1 norm is at most 2. Thus equation 27 holds for inputs of arbitrary length.

The construction uses L attention heads and L disjoint blocks of size K, so its width is $O(LK)$. Moreover, neither the precision condition nor the bound in equation 27 depends on K. □

G.2 Proof of Theorem 5.1

We restate Theorem 5.1 below.

Theorem G.3. *There are functions v_d, v_{out}, v_{end}, and v_r computable in linear time, such that for any Turing machine $\mathbb{T} = (Q, \Sigma_1, \Sigma_2, \delta)$, one can construct an agent A_q that is able to memorize $\mathbb{T}$. Moreover, A_q satisfies the following:*

(1) The transformers $\mathcal{F}_d, \mathcal{F}_c$ have constant depth $O(1)$, width $O(|Q| \cdot |\Sigma_2|)$, number of heads $O(\ln |Q|)$, and numerical precision $q = O(\ln |Q|)$.

(2) The agent executes step (b2) at most $t_{\mathbb{T}}(x)+1$ times; the memory module requires at most $O(t_{\mathbb{T}}(x)+\mathrm{len}(x)+\log_2(|Q|))$ space; the length of output for $\mathcal{F}_d, \mathcal{F}_c$ in each reasoning is $O(\ln|Q|)$.

As a consequence, the agent A_q is computationally the same as $\mathbb{T}$.

We now prove Theorem 5.1.

Proof. Let $\Sigma = \Sigma_2 \cup \{\sigma_b, \sigma_e, 0, 1, -1, \sigma_T\}$. Without loss of generality, let $\sigma_{|\Sigma_2|} = B$ and $T = |\Sigma|$.

We construct the agent module by module.

Step 1: the decision Transformer $\mathcal{F}_d$.

The role of $\mathcal{F}_d$ is only to decide whether the agent should continue the simulation or terminate. Its required behavior is $\sigma_T \oplus \sigma_b \oplus \sigma_e$ when -1 does not occur in the input, and $\sigma_T \oplus \sigma_b \oplus \sigma_e \oplus B \oplus (-1)$ when -1 does occur.

Let $\mathcal{F}_1$ be the Transformer from Lemma G.1 that detects -1 and outputs $\mathcal{F}_1(x) = I_{|\Sigma_2|}$ when -1 occurs, and 0 otherwise. Let $\mathcal{F}_2$ be the Transformer from Lemma G.1 that detects σ_T and outputs $\mathcal{F}_2(x) = I_{|\Sigma_2|+5}$ when σ_T occurs, and 0 otherwise.

Finally, construct a Transformer $\mathcal{F}_3$ with an embedding dimension 5. Use $v_i = (1,0,0,0,0)$ for $\sigma_i \in \Sigma_2$, and

$$v_T = (0,1,0,0,0), \qquad v_b = (0,0,1,0,0), \qquad v_e = (0,0,0,1,0), \qquad v_{-1} = (0,0,0,0,1).$$

In its hidden layer, set $Q = K = V = 0$ and $\mathrm{FNN} = 0$. Let $h = x_{\mathrm{last}}$ denote the final-token embedding. Choose the output map so that

$$\begin{aligned} &\mathcal{F}_3(x)[|\Sigma_2|+6] = 3h[1], \qquad &&\mathcal{F}_3(x)[|\Sigma_2|+1] = 10h[2], \\ &\mathcal{F}_3(x)[|\Sigma_2|+2] = 10h[3], \qquad &&\mathcal{F}_3(x)[0] = h[4] + 10h[5], \\ &\mathcal{F}_3(x)[|\Sigma_2|+5] = -10h[4], \end{aligned}$$

with all remaining coordinates equal to zero.

Set

$$\mathcal{F}_d = \frac{5}{4}\mathcal{F}_1 + 4\mathcal{F}_2 + \mathcal{F}_3.$$

By direct inspection of the output logits, $\mathcal{F}_d$ has the desired decision behavior.

Define the stopping rule by $v_{\mathrm{end}}(x) = f_{\mathrm{Judge},-1}(x)$. Thus $v_{\mathrm{end}}(x) = 1$ exactly when the last symbol is -1.

Step 2: the data-selection function v_d.

Let $v_d = f_{\mathrm{Search}}$. For an encoded memory state,

$$v_d\big(\mathcal{T}'(x)[i], \mathcal{F}_d(\mathcal{T}'(x)[i])\big) = v_d\big(\mathcal{T}'(x)[i], \sigma_T \oplus \sigma_b \oplus \sigma_e\big) = \ell_i(x),$$

where

$$\ell_i(x) := a_i \oplus \sigma_b \oplus B_{|Q|}(k) \oplus \sigma_e \oplus b_i,$$

with $a_i = \mathcal{T}(x)[i][j-1]$ and $b_i = \mathcal{T}(x)[i][j]$. Thus $\ell_i(x)$ is the local segment surrounding the current head position, where b_i is the symbol currently scanned by the head and a_i is its left neighbor.

For the initial raw memory $M = x$, the symbols σ_b and σ_e have not yet been inserted. By the no-match branch of f_{Search},

$$v_d\big(x, \sigma_T \oplus \sigma_b \oplus \sigma_e\big) = B \oplus x[1].$$

The execution Transformer interprets this two-symbol input as $a = B$, $k = 1$, and $b = x[1]$, corresponding to the initial state q_1 of the Turing machine.

Step 3: the execution Transformer $\mathcal{F}_c$.

We construct $\mathcal{F}_c$ so that it memorizes one transition of the Turing machine. Let $\delta(q_k, b_i) = (q_{k'}, c_i, m_i)$ with $m_i \in \{-1, 0, 1\}$, where $q_{k'}$ is the next state, c_i is the symbol written at the

current head position, and m_i is the head-movement direction. Regard $\tau_i(x) := (k', c_i, m_i)$ as the transition information computed in Part 5.

Part 6 converts this transition information, together with the local symbol a_i, into the local memory-update block

$$R_{m_i} = \begin{cases} S_{k'} \oplus a_i \oplus c_i, & m_i = -1, \\ a_i \oplus S_{k'} \oplus c_i, & m_i = 0, \\ a_i \oplus c_i \oplus S_{k'}, & m_i = 1. \end{cases}$$

Define $u_i(x) := R_{m_i}(x) \oplus \sigma_{m_i}$. The required autoregressive behavior of the execution Transformer is therefore

$$\widehat{\mathcal{F}}_c(\ell_i(x)) = \sigma_T \oplus R_{m_i}(x) \oplus \sigma_{m_i}.$$

If σ_T has not yet appeared, Lemma G.1 allows $\mathcal{F}_c$ to output σ_T first. It therefore remains to construct the one-step transition map after the marker has appeared.

Let $L = \lceil \log_2 |Q| \rceil$ and $d = L + 4$. Apply Lemma G.2 with its window parameter set to $2L + 10$. Then all relevant tokens in the bounded execution context can be recovered exactly into fixed coordinate blocks of the last hidden state. Indeed, every block used below is of the form V_{t+j} with $0 \le t \le L + 5$ and $j \le L + 4$, and hence

$$t + j \le 2L + 9.$$

Moreover, every valid autoregressive execution context has length at most $2L + 10$, so Lemma G.2 applies at every generation step. The remaining computation is implemented by a constant-depth ReLU network, divided into the following seven parts.

Part 0: indicator gadgets.

For integer-valued inputs, define

$$h_a(u) = \mathrm{ReLU}(u - a + 1) - 2\,\mathrm{ReLU}(u - a) + \mathrm{ReLU}(u - a - 1).$$

Then $h_a(u) = 1$ if $u = a$ and 0 otherwise. We also use $\mathrm{clip}(u) = \mathrm{ReLU}(u) - \mathrm{ReLU}(u - 1)$ and, for $v \in [0, 1]$ and $g \in \{0, 1\}$, $\mathrm{Gate}(v, g) = \mathrm{ReLU}(v + g - 1)$. These gadgets implement the case distinctions used below.

Part 1: distinguish the initial configuration.

Let V_j denote the copied token at relative offset j given by Lemma G.2. Define $E_t = \mathrm{clip}((V_t)_{\sigma_T})$ and

$$U_j = \mathrm{clip}\left(\sum_{t=0}^{d+1} \mathrm{Gate}\big(\mathrm{clip}(V_{t+j}), E_t\big)\right).$$

Thus U_j is the j-th token preceding the last σ_T. Set $\nu = (U_2)_{\sigma_e}$. On every valid execution input, $\nu = 0$ exactly in the initial case and $\nu = 1$ once the state block has been explicitly encoded.

Part 2: isolate the current local configuration.

In the normal case, $b = U_1$ and $a = U_{L+4}$, while the j-th bit of the current state code is obtained from U_{L+3-j} for $1 \le j \le L$. When $\nu = 0$, the execution input represents the implicit initial configuration, so we instead use $a = B$ and $k = 1$, while still taking $b = U_1$.

Part 3: compute the generation phase.

Retain the indicators $E_0, E_1, \ldots, E_{d+1}$. On every valid autoregressive prefix, exactly one E_t equals 1. It indicates that exactly t tokens have already been generated after σ_T and will be used in Part 6 to select the next output token.

Part 4: decode the current state.

Let $u_1, \ldots, u_L$ be the recovered state bits and define $k_{\mathrm{raw}} = \sum_{j=1}^{L} 2^{L-j} u_j$. Using ν, select

$$k_{\mathrm{current}} = \begin{cases} 1, & \nu = 0, \\ k_{\mathrm{raw}}, & \nu = 1. \end{cases}$$

This selection is implemented by the gating gadgets from Part 0.

Part 5: apply the transition function.

For each $(k, s) \in Q \times \Sigma_2$, define

$$T_{k,s} = \mathrm{ReLU}\big(h_k(k_{\mathrm{current}}) + b_s - 1\big),$$

where b_s is the s-coordinate of the one-hot representation of b. Exactly one such indicator is active. If $\delta(q_k, s) = (q_{k'}, c, m)$, the corresponding transition-table entries directly produce the encoded next state $B_{|Q|}(k')$, the written symbol c, and the direction m. Since there are $O(|Q||\Sigma_2|)$ state-symbol pairs, this part has width $O(|Q||\Sigma_2|)$ and constant depth.

Part 6: reconstruct and serialize the encoded configuration.

Let $S_{k'} = \sigma_b \oplus B_{|Q|}(k') \oplus \sigma_e$. As in the original construction, define

$$R_m = \begin{cases} S_{k'} \oplus a \oplus c, & m = -1, \\ a \oplus S_{k'} \oplus c, & m = 0, \\ a \oplus c \oplus S_{k'}, & m = 1. \end{cases}$$

The transition indicators select the appropriate R_m.

Finally, the phase indicators E_t serialize this block: when $E_t = 1$, the network outputs the $(t+1)$-st token of R_m; after all tokens of R_m have been generated, it outputs the direction token τ_m, followed by σ_0. If σ_T has not yet appeared, the marker detector instead outputs σ_T first. Therefore $\widehat{F}_c(\ell) = \sigma_T \oplus R_m \oplus \tau_m$.

Step 4: memory replacement and final output.

Define

$$v_r(x, z) := \begin{cases} B \oplus f_{\mathrm{Replace}}\big(x, \sigma_T \oplus \sigma_b \oplus \sigma_e, z\big) \oplus B, & \text{if both } \sigma_b \text{ and } \sigma_e \text{ occur in } x, \\ B \oplus z[2 : |z| - 1] \oplus x[2 : |x|] \oplus B, & \text{otherwise.} \end{cases}$$

The second case occurs only at the first execution step, when the memory still consists of the unencoded input. Since $\widehat{F}_c(\ell_0(x)) = \sigma_T \oplus R_{m_0}(x) \oplus \tau_{m_0}$, we have

$$v_r\big(x, \widehat{F}_c(\ell_0(x))\big) = B \oplus R_{m_0}(x) \oplus x[2 : |x|] \oplus B,$$

which is an encoded representation of the configuration after the first transition.

For every subsequent step, both σ_b and σ_e occur in the memory. By Step 2, $f_{\mathrm{Search}}(T'(x)[i], \sigma_T \oplus \sigma_b \oplus \sigma_e)$ locates the current local configuration

$$\ell_i(x) = a_i \oplus \sigma_b \oplus B_{|Q|}(k) \oplus \sigma_e \oplus b_i.$$

By the construction in Part 6, $\widehat{F}_c(\ell_i(x)) = \sigma_T \oplus R_{m_i}(x) \oplus \tau_{m_i}$. Hence, by the definition of f_{Replace}, the local block $\ell_i(x)$ is replaced by $\widehat{F}_c(\ell_i(x))[2 : |\widehat{F}_c(\ell_i(x))| - 1] = R_{m_i}(x)$, while all symbols outside this block are left unchanged.

Therefore, after each execution cycle, the memory is a blank-padded encoding of the next Turing-machine configuration, namely, $B^{a_{i+1}} \oplus T'(x)[i+1] \oplus B^{b_{i+1}}$ for some $a_{i+1}, b_{i+1} \geq 1$. These additional boundary blanks do not change the represented tape configuration and ensure that the head can always move one position to the left or to the right.

Finally, let

$$v_{\mathrm{out}}(x) = f_{\mathrm{Delete}}\big(x, \sigma_b \oplus \sigma_e \oplus (-1) \oplus B\big),$$

After these auxiliary symbols are removed, the remaining sequence is exactly the output of the memorized Turing machine.

It remains to verify the computational-equivalence statement. Each execution cycle of the agent memorizes exactly one transition of $\mathbb{T}$. Hence, on every stoppable input x, the agent executes step (b2) at most $t_{\mathbb{T}}(x) + 1$ times.

Throughout the computation, the memory length is at most $O(t_{\mathbb{T}}(x)+\mathrm{len}(x)+\log_2|Q|)$. When $|Q|$ and $|\Sigma_2|$ are treated as constants, the depth, width, number of attention heads, numerical precision, and per-invocation output length of $\mathcal{F}_d$ and $\mathcal{F}_c$ are all constants independent of x. Therefore, each invocation of the fixed Transformers can be carried out in time polynomial in the current memory length. Since v_d, v_r, v_{end}, and v_{out} are computable in linear time, and there are at most $t_{\mathbb{T}}(x)+1$ execution cycles, the total running time of A_q is polynomial in $t_{\mathbb{T}}(x)+\mathrm{len}(x)$.

Thus A_q memorizes $\mathbb{T}$ with at most polynomial overhead. Conversely, the entire computation of A_q consists only of finite-precision Transformer evaluations, linear-time tool calls, and updates of finite strings, and hence can itself be memorized by a Turing machine with at most polynomial overhead. Together with $A_q(x)=\mathbb{T}(x)$ for every stoppable input x, this shows that when $|Q|$ and $|\Sigma_2|$ are constants, A_q is computationally equivalent to $\mathbb{T}$.

Moreover, v_{end} returns 1 if and only if the simulated configuration reaches the halting state q_s. Hence, if T does not halt on x, the agent loop also runs indefinitely. □

G.3 Agent with One Transformer

In this section, we show that the two transformers $\mathcal{F}_c, \mathcal{F}_d$ in the agent can be the same one. We can consolidate the functions of $\mathcal{F}_c$ and $\mathcal{F}_d$ into a single transformer, $\mathcal{F}_s$, and include a suitable prompt so that $\mathcal{F}_s$ can distinguish whether it is performing decision-making or execution.

Write $A^s=(\mathcal{F}_s,\mathbf{M},v_{\mathrm{end}},v_d,v_r,v_{\mathrm{out}})$ to indicate that the agent uses the same transformer $\mathcal{F}_s$ in both the decision module and the execution module. Then we have that:

Proposition G.4. *If $|\Sigma|\geq|\Sigma_2|+7$ holds, there exist linear time computable functions v_d, v_{out}, v_{end}, v_r such that for any Turing machine $\mathbb{T}=(Q,\Sigma_1,\Sigma_2,\delta)$, there is an agent A_q^s such that A_q^s can memorize the Turing machine $\mathbb{T}$, and A_q^s satisfies that:*

(1) $\mathcal{F}_s$ has constant depth $O(1)$, width $O(|Q|\cdot|\Sigma_2|)$, number of heads $O(\ln|Q|)$, and numerical precision $q=O(\ln|Q|)$.

(2) The agent executes step (b2) at most $t_{\mathbb{T}}(x)+1$ times, memory module requires at most $O(t_{\mathbb{T}}+\mathrm{len}(x)+\log_2(|Q|))$ space, the length of output for F_s in each reasoning is $O(ln|Q|)$.

The proof follows the same construction as Theorem 5.1, with an additional prompt symbol used to distinguish the decision module from the execution module.

We first prove a simple combination lemma.

Lemma G.5. *Let $q\geq 6$ be fixed, and let $\mathcal{Z}\subseteq\Sigma^*$ be a class of nonempty contexts, closed under taking nonempty prefixes, in which a distinguished token σ_k occurs at most once. Suppose that two q-precision Transformers $\mathcal{F}^1$ and $\mathcal{F}^2$ are well-defined on every $z\in\mathcal{Z}$, and that*

$$\mathcal{F}^1(z),\mathcal{F}^2(z)\in[0,1]^{|\Sigma|+1}.$$

Assume that their positional-encoding blocks can be embedded into disjoint blocks of a common RoPE matrix.

Then there exists a q-precision Transformer

$$\mathcal{F}=U(\mathcal{F}^1,\mathcal{F}^2,\sigma_k)$$

such that, for every $z\in\mathcal{Z}$,

$$\mathcal{F}(z)=\begin{cases}\mathcal{F}^1(z), & \sigma_k \text{ does not occur in } z,\\ \mathcal{F}^2(z), & \sigma_k \text{ occurs in } z.\end{cases}$$

Moreover, the construction introduces only constant-factor/additive overhead in depth, width, and number of heads, and does not change the numerical precision q.

Proof. By Lemma G.1, there exists a q-precision Transformer D_{σ_k} such that

$$D_{\sigma_k}(z)=\begin{cases}0, & \sigma_k \text{ does not occur in } z,\\ 1, & \sigma_k \text{ occurs in } z.\end{cases}$$

Write $D(z) = D_{\sigma_k}(z)$.

Run $\mathcal{F}^1$, $\mathcal{F}^2$, and D in disjoint hidden-state blocks, and pad the shallower networks by identity layers if necessary. Let $h^1(z)$ and $h^2(z)$ denote the final hidden states of $\mathcal{F}^1$ and $\mathcal{F}^2$, so that

$$\mathcal{F}^i(z) = h^i(z)W_i + c_i, \qquad i \in \{1, 2\}.$$

Append one additional Transformer block whose value matrices are all zero, so its attention output is identically zero.The feedforward network computes as following,

$$\mathrm{ReLU}\big(h^1(z)W_1 + c_1 - D(z)\mathbf{1}\big) + \mathrm{ReLU}\big(h^2(z)W_2 + c_2 - (1 - D(z))\mathbf{1}\big)$$

and writes the result into the reserved coordinates. The final affine output layer simply reads these coordinates.

If σ_k is absent, then $D(z) = 0$, and since $\mathcal{F}^1(z), \mathcal{F}^2(z) \in [0, 1]^{|\Sigma|+1}$, $\mathcal{F}(z) = \mathcal{F}^1(z)$.

If σ_k is present, then $D(z) = 1$, and similarly $\mathcal{F}(z) = \mathcal{F}^2(z)$.

□

Proof. Let

$$\Sigma = \Sigma_2 \cup \{\sigma_p, \sigma_b, \sigma_e, 0, 1, -1, \sigma_T\}.$$

Use the same constructions of the execution Transformer $\mathcal{F}_c$, v_{end}, v_r, and v_{out} as in the proof of Theorem 5.1, while modifying only the output layer of decision Transformer $\mathcal{F}_d$, with the same common precision $q = O(\ln |Q|)$ chosen from the outset sufficiently large so that $q \geq 6$

The only change is that the local input supplied to the execution module is prefixed by σ_p. Define

$$v_d(x, z) = \sigma_p \oplus f_{\mathrm{Search}}(x, z).$$

Enlarge the copying window in the construction of $\mathcal{F}_c$ by one position. Prefixing σ_p shifts all original tokens by one position and therefore preserves all relative offsets among them. For every copying block used by the execution construction in the normal branch, the selected relative offset still points to the same original token, whereas σ_p occupies a new non-target position. By Lemma G.2, every non-target position receives exactly zero attention weight under the q-precision softmax. In the initial branch, only the copied blocks corresponding to U_1 and U_2 are used, and these are likewise unchanged. Consequently, for every valid local input ℓ and every autoregressive prefix $y[1:j]$,

$$\mathcal{F}_c\big(\sigma_p \oplus \ell \oplus y[1:j]\big) = \mathcal{F}_c\big(\ell \oplus y[1:j]\big).$$

Let $\mathcal{Z}$ contain all decision contexts, prompted execution contexts, and their nonempty autoregressive prefixes.

Recall that $\mathcal{F}_d = \frac{5}{4}\mathcal{F}_1 + 4\mathcal{F}_2 + \mathcal{F}_3$. By direct inspection of the construction in Theorem 5.1, every output coordinate satisfies $-10 \leq [\mathcal{F}_d(z)]_a \leq 10$ for every $z \in \mathcal{Z}$. Define

$$\widetilde{\mathcal{F}}_d(z) = \frac{\mathcal{F}_d(z) + 10\mathbf{1}}{20}.$$

Then $\widetilde{\mathcal{F}}_d(z) \in [0, 1]^{|\Sigma|+1}$, and, since this is a strictly increasing affine transformation of all output logits, $\arg\max_a [\widetilde{\mathcal{F}}_d(z)]_a = \arg\max_a [\mathcal{F}_d(z)]_a$. Thus $\widetilde{\mathcal{F}}_d$ has exactly the same autoregressive output as $\mathcal{F}_d$.

By the construction of the execution Transformer, its final output coordinates are clipped to $[0, 1]$, and hence

$$\mathcal{F}_c(z) \in [0, 1]^{|\Sigma|+1} \qquad \text{for every } z \in \mathcal{Z}.$$

Therefore Lemma G.5 applies. Set

$$\mathcal{F}_s = U(\widetilde{\mathcal{F}}_d, \mathcal{F}_c, \sigma_p).$$

On every decision call, σ_p is absent, so $\mathcal{F}_s = \widetilde{\mathcal{F}}_d$ on the corresponding autoregressive contexts. Since the affine rescaling preserves the argmax, $\mathcal{F}_s$ has the same autoregressive output as $\mathcal{F}_d$. On every execution call, σ_p occurs exactly once, so $\mathcal{F}_s$ has the same autoregressive output as $\mathcal{F}_c$.

The prompt symbol is not written into persistent memory. Hence v_r, v_{end}, and v_{out} are unchanged, and every cycle produces the same memory update as in Theorem 5.1. Therefore, the shared-transformer agent memorizes the same Turing-machine computation.

Finally, Lemma G.5 changes the depth, width, and head count only by constant factors and does not change q. Thus, all resource bounds of Theorem 5.1 remain unchanged up to absolute constants. □

H Application Scopes of Results for Finite Precision transformers, Infinite Precision transformers, and Agents

In this section, we will clarify the scope within which the theorems presented in this paper are valid.

Full Conclusion.

This article clearly specifies the structure of the transformer, defines position encoding, and defines the Turing machine structure we are considering. Although these structures simplify the LLMs used in reality, they retain the core structure. In this setting, we have demonstrated their completeness and incompleteness.

For some work such as Jiang et al. (2026), introduce Heaviside activation function and $\ln n$ scaling factor to prove the Turing completeness for transformer, this approach changes the structure of the transformer, so our conclusion cannot be applied to this type of structure, and therefore there is no contradiction with them.

Results in Section 4.2 do not hold for finite precision transformers.

Section 4.2 considers the Turing incompleteness of infinite precision transformers on non-tame Turing machines, it holds for any $|\Sigma|$ such that $\Sigma_2 \subset \Sigma_0$, but we must point out that these conclusions cannot be directly extended to finite precision transformers. Naturally, they cannot be applied to agents with finite transformers.

Results in Section 4.2.1 do not hold for a finite precision transformer. Precision loss does not arise in an infinite-precision transformer and is therefore not taken into account in the sensitivity analysis of Theorem 4.2. Once the loss of precision is taken into account, that bound will no longer hold.

Results in Section 4.2.2 are not valid for finite precision transformers. Under the precision limit, if the transformer can correctly output, the maximum weight is at least 10^{-q} larger than the others. By using several amplification layers, such a value can naturally reach any large amount not exceeding the precision limitation, so there is no confidence problem for a finite precision transformer.

Results in Section 4.2.3 do not hold for a finite precision transformer. Under the precision limit, when the structure of the transformer and precision are fixed, the number of parameters in the hypothetical space is limited. Therefore, as long as there is at least one parameter that can make the transformer memorize the target Turing machine, the probability of memorizing such a Turing machine is not 0.

Comparing the results in Section 4.1 and Section 5.1.

Section 4.1 considers the Turing incompleteness of finite precision transformers on non-converging Turing machines, it holds for any $|\Sigma|$ such that $\Sigma_2 \subset \Sigma_0$. Since our conclusion in section 4.1 holds for all finite precision transformers, we do not need to consider where the transformers come from, such as whether they are constructed or trained.

In Section 5.1, the transformer can complete the task through multiple cycles and calls to tools. In essence, it is a method to solve long tasks by dividing them into shorter problems, thereby avoiding having the transformer directly solve particularly difficult problems, such as non-converging Turing machines, which greatly reduces the difficulty of the tasks undertaken in the operation of the single-step transformer and ultimately realizes Turing completeness. So the results in Section 4.1 do not contradict the Turing completeness of agents that use finite precision transformers.

In the results of agent, we require that $|\Sigma_2| + 6 \leq |\Sigma|$ as shown in the proof, this is not a hard requirement: for $0-1$ Turing machine, $\Sigma_2 = \{0, 1, B\}$; for $\mathbb{T}_{\mathbf{Arith}}$, $|\Sigma_2| = 19$; on the other hand, $|\Sigma|$ is always a large value in most of transformer.

I Additional Experimental Details

I.1 Implementation Details

Datasets The benchmark contains 100 expressions for each of the 13 operator counts: $(1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 25, 30)$. Reference answers are generated by a deterministic evaluator and are represented as integers or reduced fractions. A prediction is correct only if its integer value, or both its numerator and denominator, exactly match the reference answer. The following table collects several examples from our test datasets.

id	input	answer	length	num_operands	operator_family	operators
test_short-mixed-l2-0017	$14 - 10/1$	4	2	3	mixed	["-", "/"]
test_short-subtract-l3-0008	$79 - 15 - 30 - 81$	-47	3	4	subtract	["-", "-", "-"]
test_short-mixed-l3-0019	$15/5 + 19/3$	28/3	3	4	mixed	["/", "+", "/"]
test_medium-mixed-l6-0016	$4 + 9 * 19 - 14 * 6 * 4 - 6$	-167	6	7	mixed	["+", "*", "-", "*", "*", "-"]

Models and Inference Configuration We use Llama3.1-8B, Qwen3-8B, and GLM4.5-air as our test models and evaluate instruction-tuned models served with vLLM through its OpenAI-compatible API. For each evaluated model, the decision component $\mathcal{F}_d$ and execution module $\mathcal{F}_c$ in the agent use identical model weights, numerical precision, chat template, decoding parameters, and structured-output backend, which is the same as the single model. The primary evaluation uses greedy decoding with temperature 0 and pass@1. A single-model response is limited to 1,024 generated tokens, while each decision or execution response is limited to 512 generated tokens. To avoid the influence of random factors, we add a supplementary analysis using independent pass@k sampling, which is presented in Appendix I.3.

Evaluation Metrics Our primary metric is exact-answer accuracy. Predictions are parsed as integers or reduced fractions before comparison, allowing mathematically equivalent fraction forms to match. We also report the completion rate, defined as the fraction of examples for which a system returns a parseable final answer. Both metrics are reported by checkpoint, operator count, length split, and operator family. Data and analysis are detailed in Appendix I.3.

I.2 Example Agent Trajectory

The following table presents a successful trajectory sampled from the experiment using GLM-4.5-Air. The input expression is $15/5 + 19/3$, whose reference answer is $28/3$. The table omits the models' textual reasoning and retains the actions that affect execution.

Round	Memory $\mathbf{M}_t$	$\mathcal{F}_d$	v_d	$\mathcal{F}_c$	Memory $\mathbf{M}_{t+1}$
1	$15 \underset{o_1}{/} 5 \underset{o_2}{+} 19 \underset{o_3}{/} 3$	o_1	$(15, /, 5)$	3	$3 \underset{o_2}{+} 19 \underset{o_3}{/} 3$
2	$3 \underset{o_2}{+} 19 \underset{o_3}{/} 3$	o_3	$(19, /, 3)$	$19/3$	$3 \underset{o_2}{+} 19/3$
3	$3 \underset{o_2}{+} 19/3$	o_2	$(3, +, 19/3)$	$28/3$	$28/3$

In the first round, the decision invocation F_d selects o_1. The read function v_d locates this stable identifier and returns only the local operation $15/5$. An independent computation invocation F_c returns 3, which v_r writes back to memory. The same procedure next reduces $19/3$ and finally adds the two intermediate values. The trajectory therefore uses three decision invocations and three computation invocations.

The identifiers remain attached to the surviving operators as the expression is reduced. The deterministic functions v_d and v_r only retrieve and replace memory entries: they do not select an operator, perform arithmetic, or verify that a replacement is correct. In this trajectory, the final memory state contains $28/3$, which matches the reference answer.

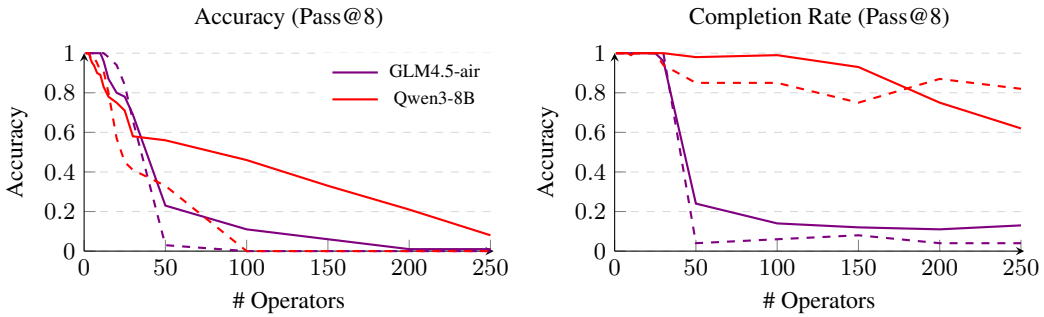


Figure 2: Accuracy rate of single model and agent as the number of required operations grows.

I.3 Detailed Result Analysis

For the supplementary Pass@8 analysis, we additionally generate 500 longer expressions using the same data-generation setting as the primary benchmark, with 100 expressions at each of 50, 100, 150, 200, and 250 operators. Each operator count contains 20 expressions from each of the five operator families.

Figure 2 reports Pass@8 accuracy and completion rate as functions of the number of operators for GLM4.5-air and Qwen3-8B. As this experiment serves only as a supplementary sampling analysis, we restrict Pass@8 evaluation to these two representative models. Pass@8 reduces the influence of single-sample stochasticity by evaluating eight sampled attempts per problem. In the accuracy panel, performance declines with expression length for all curves; at larger operator counts, however, the agent remains more accurate than the single-model baseline, which collapses toward zero. The completion-rate panel also exhibits length-dependent degradation, though its severity varies across models and settings.

This comparison reflects two opposing effects: iterative execution reduces the state-management burden of each model invocation, but each additional decision and execution call creates another opportunity for error. An incorrect operator choice or replacement is written to memory and affects all subsequent rounds. External memory therefore does not eliminate model error; its benefit depends on whether local decomposition outweighs the accumulation of per-step errors. Accuracy and completion rate diagnose different failure modes: low completion indicates protocol errors, malformed actions, or generation truncation, whereas high completion with low accuracy points to semantic errors in operator selection or arithmetic. Consequently, the agent may match or underperform the single model on short expressions without contradicting the theoretical result. The theory concerns the computational capability enabled by repeated invocation and external memory, rather than uniform empirical dominance on every finite input.