RETHINKING THE ROLE OF POSITIONAL ENCODING: SLIDING-WINDOW TRANSFORMERS WITHOUT PE REMAIN TURING COMPLETE

Anonymous authors
Paper under double-blind review

ABSTRACT

Positional encoding (PE) is widely viewed as necessary for transformers to process ordered sequences: without them, the next-token map appears permutation-invariant in its context tokens. This intuition underlies all prior universality results, which rely on positional information to prove that transformers with chain-of-thought can perform arbitrary computation, i.e., they are Turing complete. We revisit this belief in the regime most relevant to long-form reasoning, where generation proceeds through a finite sliding context window. Our opening perception is that the window mechanism itself (mildly) breaks the permutation symmetry. To distill and precisely capture the degree of this added expressiveness, we introduce an abstract autoregressive model, the HIST model, in which each update depends only on constant-size internal state and the token-count histogram within the current window. We prove that this HIST model is Turing complete by showing that the evolution of the window can reveal the token that has just left the window, which suffices to simulate Turing-complete Post machines. We then construct a sliding-window transformer over a constant-size token alphabet, without PE, and show that it can simulate the HIST model. Our result demonstrates that positional encodings are not indispensable for transformers to perform universal computation: The window sliding itself already breaks permutation symmetry and captures sufficient positional information.

1 INTRODUCTION

Transformer-based large language models with chain-of-thought (CoT) reasoning have demonstrated exceptional performance across a wide range of complex reasoning tasks, such as mathematical problem solving (Luong & Lockhart, 2025) and code generation (OpenAI, 2025; Anthropic, 2026). These developments have motivated a growing body of theoretical work studying the reasoning abilities of Transformers through the lens of expressiveness, e.g. (Pérez et al., 2021; Merrill & Sabharwal, 2023a; 2024; Li et al., 2024; Nowak et al., 2024; Qiu et al., 2024; Zubić et al., 2024; Merrill & Sabharwal, 2025; Yang et al., 2025; Li & Wang, 2025; 2026). A central conclusion of this line of research is that, once CoT reasoning is allowed, transformers can achieve universal computation, that is, they are Turing complete.

Yet existing universality results rely crucially on positional encoding (PE), which augments each token with an explicit representation of its position in the sequence, thereby enabling the model to distinguish different orderings of the same context. This view is natural when the model attends to the entire past. Transformer models consist of interleaved self-attention and feed-forward layers, which are both *order-invariant* and thus cannot distinguish permutations of the same context. Therefore, without positional information, the model lacks access to order-dependent information required for structured memory and sequential computation as in Turing machines. From this perspective, positional encodings in transformer models appear necessary for universal computation.

However, this perspective may overlook a key feature of autoregressive reasoning with CoT. In earlier settings without CoT, the context window can effectively cover the entire past. In contrast, CoT, particularly in long-form reasoning, introduces extended intermediate computations that quickly exceed the context capacity, forcing the model to operate with a relatively short sliding window rather

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.

than attend to the full history. As the generative process proceeds, this window slides forward: new tokens enter, old tokens leave, and the model attends only to the current window. With sliding-window processing, the context is no longer a stack of tokens. The motion of the window itself introduces a temporal asymmetry, however small, providing a new source of sequential information. Even if the next token generation is permutation-invariant within each individual window, the temporal evolution of the window is order-sensitive: it depends on which tokens enter and exit at each step. This leads to the central question of the paper:

Are PEs truly necessary for universal computation in sliding-window transformers?

Besides theoretical curiosity, we are interested in the problem because PE is not merely a neutral implementation detail, but can influence practical efficiency. Depending on the design, they may add extra parameters (e.g., learned absolute PEs (Vaswani et al., 2017)), incur additional overhead in the attention computation (e.g., relative PEs (Shaw et al., 2018; Su et al., 2024)), and complicate length extrapolation and context extension (Kazemnejad et al., 2023). Besides practical considerations, strong PEs can also act as an additional source of computational power, making it less clear whether the source of universal computation lies in the transformers content-based computation or in the PE attached to it.

1.1 Our contribution

We prove that the sliding-window transformer model without PE remains Turing-complete. Our results demonstrate that positional encoding is not necessary for transformers to retrieve sequential structure and achieve universal computation. Conceptually, our result separates two notions that are often conflated: although a transformer with PE is permutation-invariant with respect to the tokens inside a single window, the FIFO-like motion of a sliding window is dynamic and breaks permutation-invariance. Temporal motion can itself provide sufficient sequential structure for universal computation.

The HIST model. To frame this viewpoint more precisely, we introduce an abstract autoregressive HIST model that can only use permutation-invariant summary of data in its context window. A HIST machine repeatedly generates the next token using only a finite control state and the histogram of the current window, namely the vector of token counts indexed by the alphabet. It has no access to the positions of individual tokens: conditioned on its control state, the machines next-token rule is a permutation-invariant function of the tokens in the current window.

Theorem 1 (High level)**.** *The HIST model is Turing complete (under suitable input encoding).*

Our proof proceeds by simulating Post machines (Post, 1936) with HIST machines. A Post machine is an automaton equipped with a queue, and it is Turing complete. We view the sliding window of a HIST machine as a queue-like data structure and show that it can implement the queue operations needed for simulating Turing machines, provided that the input is given in the following form: To simulate a Post machine on input $(x_1, x_2, \ldots, x_k)$, the input to the HIST simulator is set to be $((x_1, x_2), (x_2, x_3), \ldots, (x_{k-1}, x_k))$. We refer to this input encoding as the edge encoding.

The key point is that the "dequeue" operation can be simulated without ever addressing the position of the front token explicitly. Although a HIST machine only observes a permutation-invariant summary of the window under the edge encoding, namely the parity of the number of occurrences of each token type, this information is sufficient to recover the token that has just left the window. Indeed, by comparing the parity histograms before and after a window update, while temporarily storing in its local state the newest tokens appended at the rear, the machine can infer the dropped front token. This allows the sliding window to emulate the queue of a Post machine while using only parity information about the current window.

From HIST to Transformers without PE. We then show that a sliding-window transformer without positional encodings can simulate the HIST model under the edge encoding. See Figure 1 for an illustration of the main ideas. The construction uses a constant-size token alphabet and a constant-size set of learnable parameters. The only non-constant numerical resource used in our implementation is the arithmetic precision, which we isolate in Remark 4.

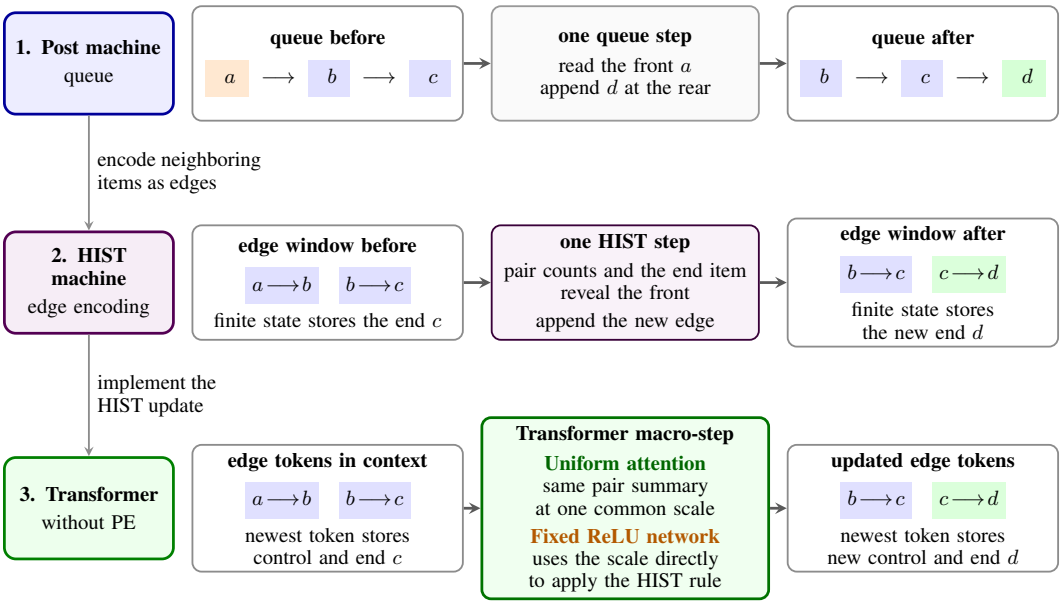

Figure 1: Three levels of the simulation, shown for a one-symbol append. The Post machine changes its queue from abc to bcd. The HIST simulator stores neighboring edges, whose counts together with the remembered end item reveal the front. A Transformer realizes the same HIST update: uniform attention produces the pair summary at a common scale, and a fixed ReLU network uses that scaled signal directly to select and emit the next pair.

Theorem 2 (High Level)**.** *Every HIST machine can be simulated by a sliding-window transformer without using PEs. Consequently, sliding-window transformers without positional encodings are Turing complete. Quantitatively, a Turing-machine run using time t and space s requires $O(ts)$ generated tokens, $O(s)$ active context, and $O(\log s)$-bit precision implementation.*

This improves upon previous universality constructions for transformers by removing the need for positional encodings; see Table 1 for comparison. Conceptually, the result separates order information from positional encodings. Universal computation does require some source of sequential structure, but this structure need not be provided by an externally supplied positional address system. In the sliding-window regime, it can instead arise from the temporal dynamics of the window itself. A few remarks are in order.

Remark 3 (Adaptive window updates)**.** Our construction uses a simple adaptive window-update mechanism. The window size is initialized to be the input length. Each generated token is augmented with a control bit that determines how the context window is updated. One value of the bit implements the usual sliding update: the new token enters the window and the oldest token leaves. The other value expands the window size by one: the new token enters the window while the oldest token is retained. Thus, the model can decide during generation whether to preserve or increase the window size.

This adaptive mechanism avoids fixing a worst-case window size in advance. A fixed-window construction typically requires a window length chosen as a function of the input length, such as an assumed space bound $s(n)$ (Li & Wang, 2025; 2026). Such a bound may be unknown a priori, and even when it is known, it reflects a worst-case requirement over all inputs of length n. By contrast, an adaptive window can grow according to the needs of the particular computation being simulated, making the required context length instance-dependent rather than worst-case over the input length.

Remark 4 (Precision)**.** Our construction uses a constant-size token alphabet and a constant number of learnable parameters, but a non-constant numerical precision. The unbounded precision is only required during generation. The description of the transformer is finite and fixed.

Table 1: Comparing Transformers' complexity to simulate $(t(n), s(n))$-bounded Turing machines. 'Dim.' denotes embedding dimension, 'Window' means effective window size, and 'CoT per TM step' denotes the CoT length needed to simulate one TM step. This paper focuses on the nontrivial case where $t(n), s(n) \geq n$.

Source	Precision	Dim.	Window	CoT per TM-step	PE
Pérez et al. (2021)	$O(\log t(n))$	$O(1)$	$n + t(n)$	1	✓
Bhattamishra et al. (2020)	unbounded	$O(1)$	$n + t(n)$	1	✓
Merrill & Sabharwal (2024)	$O(\log t(n))$	$O(1)$	$n + t(n)$	1	✓
Li et al. (2024)*	$O(1)$	$O(\log t(n))$	$O(t(n) \log t(n))$	$O(\log t(n))$ (amortized)	✓
Qiu et al. (2024)	$O(\log t(n))$	$O(1)$	$O(t(n) \log t(n))$	$O(\log t(n))$ (amortized)	✓
Yang et al. (2025)†	$O(\log t(n))$	$O(1)$	$s(n)$	1	✓
Li & Wang (2025)	$O(1)$	$O(1)$	$s(n)$	$s(n)$	✓
Li & Wang (2026)	$O(1)$	$O(1)$	$s(n)$	$s^c(n)$‡	✓
This work	$O(\log s(n))$	$O(1)$	$O(s(n))$	$O(s(n))$	No

* Any multi-tape TM running in time $t(n)$ can be simulated by a Boolean circuit of size $O(t(n) \log t(n))$ (Arora & Barak, 2009).
† The construction incorporates a reduction mechanism to recursively erase intermediate CoT steps.
‡ The exponent $c > 0$ can be made arbitrarily small by letting the precision and embedding size sufficiently large constants.

1.2 Related Work

Positional information in transformers. The original Transformer architecture uses positional encodings to inject token-order information into an otherwise permutation-equivariant attention stack (Vaswani et al., 2017). Common positional mechanisms for transformers include absolute positional encodings, such as sinusoidal positional encodings and learned positional embeddings (Vaswani et al., 2017), as well as relative positional mechanisms, such as relative positional representations and T5-style relative position biases (Shaw et al., 2018; Raffel et al., 2020); rotary position embeddings (Su et al., 2024); and attention-bias methods such as ALiBi (Press et al., 2022). PE affect both empirical behavior and formal expressiveness. For example, Sälzer et al. (2025) studied the counting ability of transformers without neither positional information nor sliding window mechanism. PE is also closely related to length generalization: extrapolating beyond the training length can change model behavior in ways that depend on the positional-encoding scheme (Kazemnejad et al., 2023). Our work is complementary to this line of research. Instead of designing a new positional mechanism, we show that PEs are not necessary for universality when the model uses a sliding context window. The key point is that, although the computation inside each window can be permutation-invariant, the FIFO-style motion of the window over time provides a sequential signal.

Transformer universality and chain-of-thought. Attention-based architectures are known to be Turing complete under some architectural assumptions (Pérez et al., 2021). Recent work has refined this picture by studying how chain-of-thought, precision, depth, and window size affect the computational power of transformers (Merrill & Sabharwal, 2023b; Qiu et al., 2024; Merrill & Sabharwal, 2025; Yang et al., 2025). Closest in spirit are universality constructions for transformers with constant precision and short window by simulating Post machines (Li & Wang, 2025; 2026). All these constructions use PE as an indispensable resource, and in some cases, PE also act as an additional source of computational power, making it less clear whether the source of universal computation lies in the transformers content-based computation or in the PE attached to it. For instance, Li et al. (2024) uses PE to store the structure of the circuit to be computed, and (Li & Wang, 2025; 2026) employ PE, depending on the simulated space bound, to single out the unique token that should be attended to. Our contribution is to remove explicit PE from the universality proof by routing the simulation through a permutation-invariant window-summary model.

2 Preliminaries

We use deterministic single-tape Turing machines with a tape bounded on the left, initially $\triangleright x \perp\perp \cdots$, and no transition moving left of that boundary. This standard universal model fixes the tape convention for all statements below.

For a finite alphabet Σ and word $w = w_1 \cdots w_m$, define

$$|w| = m, \qquad \mathsf{hist}(w) = \big(\#\{j : w_j = \sigma\}\big)_{\sigma \in \Sigma} \in \mathbb{N}^{\Sigma}.$$

The empty word ϵ has zero histogram. We write $u \parallel v$ or uv for concatenation and $\mathsf{tail}(w)$ for the word obtained by deleting the first symbol of w. All uses of tail below have a nonempty argument. The notation $\mathbf{e}_\sigma^d$ denotes the standard basis vector indexed by σ in $\mathbb{R}^d$; we omit d when the context is clear.

2.1 Post machine

A Post machine (PM) (Post, 1936) is defined as a tuple $\langle \Sigma, Q, \delta \rangle$ where

- Σ is the tape alphabet, including a designated "blank" symbol $\perp$, a designated "start" symbol $\triangleright$, and numbers 0 and 1.
- Q is a finite set consisting of possible states. We assume Q contains a designated start state q_{start} and a designated halting state q_{halt}.
- $\delta : Q \times \Sigma \to Q \times (\Sigma \cup \Sigma^2)$ is the transition function, specifying how the machine updates the tape and state.

The machine has access to an unbounded queue whose entries are symbols from Σ. The queue is initialized to contain the start symbol $\triangleright$ and a finite input string $x \in \{0,1\}^*$. The machine starts in state q_{start}. The computation repeatedly performs the following step until the machine enters q_{halt}. Suppose the current state is $q \in Q$, and the symbol at the front of the queue is $\sigma \in \Sigma$. If

$$\delta(q, \sigma) = (q', \sigma')$$

for some $q' \in Q$ and $\sigma' \in \Sigma \cup \Sigma^2$, then the machine updates its state from q to q', removes the front symbol σ from the queue, and appends the string σ' to the rear of the queue.

Turing completeness of PM The model of PM is Turing complete, i.e., equivalent to TMs in terms of computational power. Specifically,

Theorem 5. *Given any single-tape TM running in t time and s space, there is an equivalent PM running in $O(t \times s)$ time and using a s-size queue.*

This theorem has been implicitly proved in several places, see, e.g., Appendix A in (Li & Wang, 2025).

2.2 Transformers without PE

In the following definition of transformers without PE, for notational simplicity, we only consider the single-head, single-layer transformer, which suffices for our construction.

Let $\mathcal{V}$ be a finite vocabulary. A decoder-only transformer is an autoregressive sequence model mapping $\mathcal{V}^* \to \mathcal{V}$. Given a sequence $(v_1, \ldots, v_i) \in \mathcal{V}^i$, it computes the next token v_{i+1} via the following layers:

1. **Token Embedding:** Maps each $v_j \in \mathcal{V}$ to a vector $\mathrm{emb}(v_j) \in \mathbb{R}^D$, where D is the embedding dimension.
2. **Positional Encoding:** There is no PE, and thus the initial input representation is exactly $h_j^0 := \mathrm{emb}(v_j)$.
3. **Self-Attention Layer:** Using parameter matrices $Q, K, V \in \mathbb{R}^{D \times D}$, the unmasked attention logits are computed as $\mathsf{logit}_{i,j} = \langle Qh_i^0, Kh_j^0 \rangle$. The attention weights $\alpha_{i,j}$ are obtained via the standard softmax function *over the context window* $C = (v_{i-|C|+1}, \ldots, v_i)$; namely,

$$\alpha_{i,j} := \frac{\exp(\mathsf{logit}_{i,j})}{\sum_{j=i-|C|+1}^{i} \exp(\mathsf{logit}_{i,j})} \quad \text{for } j = i - |C| + 1, \ldots, i.$$

   The output is $a_i = \sum_{j=i-|C|+1}^{i} \alpha_{i,j} V h_j^0$, and the residual stream becomes $h_i^{0.5} = a_i + h_i^0$.

4. **Feed-Forward Network (MLP):** A multi-layer ReLU network f_{mlp} is applied, yielding $h_i^1 = f_{\text{mlp}}(h_i^{0.5}) + h_i^{0.5}$.
5. **Output Layer:** The next token is generated deterministically by

$$v_{i+1} = \operatorname*{argmax}_{v \in \mathcal{V}} (\text{emb}(v)^\top h_i^1).$$

Adaptive context window and CoT generation. We consider a model where the transformer uses an adaptive sliding-window update during generation. In the beginning, on input sequence $v = (v_1, \ldots, v_n)$, the context window is set to be the entire prompt sequence. During generation, every vocabulary token w carries an update bit $\gamma(w) \in \{0, 1\}$. After generating w, the context window is updated by

$$C \leftarrow \begin{cases} C \parallel w, & \gamma(w) = 1, \\ \mathsf{tail}(C) \parallel w, & \gamma(w) = 0. \end{cases}$$

Thus $\gamma = 0$ is the usual sliding update, while $\gamma = 1$ appends a token without evicting an old token and hence increases the window length $|C|$ by one.

Positional encoding. In an *absolute* PE scheme, each absolute position j is assigned a vector p_j, and the embedding layer is replaced by $h_j^0 = \mathsf{emb}(v_j) + p_j$; standard examples include sinusoidal absolute encodings and learned absolute position embeddings (Vaswani et al., 2017; Radford et al., 2019). In a *relative* PE scheme, the attention computation is modified by a quantity depending on the offset $i - j$, for example, by adding a relative bias or a relative key/value vector to the score or value computation (Shaw et al., 2018). Rotary position embeddings implement a related relative mechanism by applying position-dependent rotations to queries and keys, so that their inner products encode relative offsets (Su et al., 2024).

3 HIST Model and Its Turing Completeness

A *HIST machine* has a finite alphabet Γ, a finite state set Q_H, an initial state, designated halting states, and a fixed computable transition map

$$\delta_H : Q_H \times \mathbb{N}^\Gamma \longrightarrow Q_H \times (\Gamma \cup \Gamma^2).$$

Its configuration consists of a state u and a finite window $W \in \Gamma^*$. If $\delta_H(u, \mathsf{hist}(W)) = (u', d)$, its next configuration is

$$(u', \mathsf{tail}(W \parallel d)). \tag{1}$$

Thus it appends one or two symbols and deletes the first symbol of the resulting word. This definition also covers $W = \epsilon$, since d is nonempty. Equivalently, the front tape head advances once and the rear head advances by $|d|$. The window may be supplied in an explicitly specified finite-alphabet encoding of the input. We specify that encoding below; the machine does not have to construct it from raw bits. Computability of the transition rule excludes an oracle encoded in an arbitrary function on unbounded histograms.

Edge encoding. For a nonempty word $C = x_1 \cdots x_k$ over Σ, define the edge encoding of C as

$$E(C) = ((x_1, x_2), \ldots, (x_{k-1}, x_k)) \in (\Sigma^2)^*, \qquad E(x_1) = \epsilon.$$

For an edge histogram $h \in \mathbb{N}^{\Sigma^2}$ define its divergence by

$$(\partial h)_\sigma = \sum_{a \in \Sigma} h(\sigma, a) - \sum_{a \in \Sigma} h(a, \sigma). \tag{2}$$

The contributions of internal queue symbols cancel, giving the identity

$$\partial \mathsf{hist}(E(C)) = \sum_{j=1}^{k-1} (\mathbf{e}_{x_j} - \mathbf{e}_{x_{j+1}}) = \mathbf{e}_{x_1} - \mathbf{e}_{x_k}. \tag{3}$$

Self-loops and repeated symbols make no difference. In particular, if the rear symbol $b = x_k$ is stored in the finite control, then

$$\partial \mathsf{hist}(E(C)) + \mathbf{e}_b = \mathbf{e}_{x_1}.$$

The queue front is therefore determined by a linear histogram operation and one finite-state symbol.

Theorem 6 (HIST simulates Post machines using edge encoding)**.** *For every Post machine* PM *with alphabet* Σ*, there is a HIST machine with alphabet* $\Gamma = \Sigma^2$ *that simulates it one transition at a time. If the queue after* t *transitions is* C_t*, the HIST window is* $W_t = E(C_t)$ *and its finite state stores the queue-automaton state and the rear symbol of* C_t*. For initial queue* $\triangleright x\ \bot$*, the initial HIST window is* $E(\triangleright x\ \bot)$ *and its initial state is the fixed pair* $(q_{\text{start}}, \bot)$*.*

Proof. Take $Q_H = Q_{\mathsf{PM}} \times \Sigma$, with all (q_{halt}, b) designated as halting states. Given state (q, b) and edge histogram h, recover the unique σ with $\partial h + \mathbf{e}_b = \mathbf{e}_\sigma$. On inputs where this is not a basis vector, define the transition arbitrarily. Write $\delta_{\mathsf{PM}}(q, \sigma) = (q', a)$. If $a = a_1$, output the single edge (b, a_1) and enter (q', a_1). If $a = a_1 a_2$, output $(b, a_1)(a_1, a_2)$ and enter (q', a_2). In either case, writing $d = E(ba)$ gives

$$E(C) \parallel d = E(Ca), \qquad \mathsf{tail}(E(Ca)) = E(\mathsf{tail}(Ca)) = E(\mathsf{tail}(C)a).$$

The second identity applies because Ca has at least two symbols. It remains valid when C is a singleton and $E(C)$ is empty. By (3) the recovered symbol is the true queue front, and by (1) the next edge window is exactly the encoding of the next queue. The stored rear and control state are updated correctly. Induction from the prescribed initial configuration proves the claim. □

Corollary 7. *HIST machines are Turing complete under the edge encoding.*

Proof. Combine Theorems 5 and 6. □

4 Simulating the HIST Machines under Edge Encoding

Theorem 8 (Simulating HIST machines under edge encoding)**.** *Fix a queue automaton and the HIST machine constructed in Theorem 6. There is a transformer with one standard softmax attention head, one attention layer, a fixed finite ReLU MLP, no positional encodings, and no layer normalization, that faithfully simulates this HIST machine. Its vocabulary, dimension, MLP depth and width, and all parameters are independent of the input and window length.*

Each HIST transition requires at most four generated tokens, and halting is signaled by one additional token. If all HIST windows in a run have length at most S*, then every transformer context has length at most* $S + 2$*. Using* $O(\log(S + 2))$*-bit precision for the arithmetic during generation, the simulation is exact.*

Proof. We start with the construction of the transformer.

Vocabulary and input encoding. The finite control-label set is

$$R = \{\mathsf{N}(q, b) : q \in Q_{\mathsf{PM}}, b \in \Sigma\} \cup \{\mathsf{P}(q, a_1, a_2) : q \in Q_{\mathsf{PM}}, a_1, a_2 \in \Sigma\}.$$

A normal label $\mathsf{N}(q, b)$ records a completed simulated configuration. A pending label $\mathsf{P}(q', a_1, a_2)$ records that (b, a_1) has been emitted and (a_1, a_2) must be emitted next. Let $\Lambda = \Gamma \cup \{\#_0, \#_1\}$ be the set of *displayed symbols*, where we introduce two markers $\#_0$ and $\#_1$. The vocabulary is

$$\mathcal{V} = (\Lambda \times R \times \{0, 1\}) \cup \{\mathsf{HALT}\}.$$

The bit in an ordinary token is its adaptive update bit. Generation stops upon producing HALT.

For context window $\mathcal{C} \in \mathcal{V}^*$, its *edge projection*, denoted $\mathsf{edge}(\mathcal{C}) \in \Gamma^*$, is the word obtained by concatenating the Λ-component of the tokens in $\mathcal{C}$ and deleting the markers. A *ready* context contains each marker exactly once and has edge projection $E(C)$, where C is the simulated queue. If the input to the Post machine is $E(\triangleright x\ \bot) = e_1 \cdots e_L$, then the prompt is

$$(e_1, N(q_{\text{start}}, \bot), 0), \ldots, (e_L, N(q_{\text{start}}, \bot), 0), (\#_0, N(q_{\text{start}}, \bot), 0), (\#_1, N(q_{\text{start}}, \bot), 0). \quad (4)$$

This prompt is supplied externally.

Embeddings and uniform attention. Set $K = |\mathcal{V}|$, $N = |\Lambda|$, and $D = K + N$. Use the tied input/output embedding

$$\mathsf{emb}(v) = (\mathbf{e}_v^K, \mathbf{0}^N).$$

Set $Q_{\text{att}} = K_{\text{att}} = 0$. The value matrix maps an ordinary token with displayed symbol τ to $(\mathbf{0}^K, \mathbf{e}_\tau^N)$, and maps HALT to zero. Thus, before halting, a context of length m with newest token v_i gives

$$h_i^{0.5} = (\mathbf{e}_{v_i}^K, A), \qquad A(\tau) = \frac{\mathrm{cnt}_{\mathcal{C}}(\tau)}{m}, \tag{5}$$

where $\mathrm{cnt}_{\mathcal{C}}(\tau)$ is the number of times the symbol $\tau \in \Gamma$ appears $\mathcal{C}$. The first block stores the newest control label, and the second is a normalized histogram.

Scaled mode and front indicators. Write $[z]_+ = \max(z, 0)$ and $\min(u, v) = u - [u - v]_+$. With $A_j = A(\#_j)$ define

$$r = \min(A_0, A_1), \qquad r_0 = [A_1 - A_0]_+, \qquad r_1 = [A_0 - A_1]_+. \tag{6}$$

On reachable contexts these are $(\rho, 0, 0)$, $(0, \rho, 0)$, or $(0, 0, \rho)$, where $\rho = 1/m$. They respectively indicate ready form, missing $\#_0$, and missing $\#_1$. For each label $\ell \in R$, its indicator $I_\ell \in \{0, 1\}$ is a fixed sum of coordinates of $\mathbf{e}_{v_i}^K$. Define

$$\begin{aligned} B_\sigma &= \sum_q I_{\mathsf{N}(q,\sigma)}, \qquad D_\sigma = \sum_{a\in\Sigma} A((\sigma, a)) - \sum_{a\in\Sigma} A((a, \sigma)), \\ y_\sigma &= D_\sigma + [r + B_\sigma - 1]_+, \qquad z_\sigma = \min([y_\sigma]_+, r). \end{aligned} \tag{7}$$

In normal ready form, (3) gives $y_\sigma = z_\sigma = \rho \mathbf{1}[x_1 = \sigma]$; otherwise we have $r = 0$, so every $z_\sigma = 0$. The remaining gates select the current finite control label, route the unique active mass ρ to the chosen output token, and cancel the incoming residual. Their complete formulas and proof of fixed ReLU realizability are given in Section A.

Updates. In ready form with label $\mathsf{N}(q, b)$ and recovered front σ, evaluate the finite table $\delta_{\mathsf{PM}}(q, \sigma) = (q', a)$. For $a = a_1$, emit $((b, a_1), \mathsf{N}(q', a_1), 0)$. For $a = a_1 a_2$, first emit $((b, a_1), \mathsf{P}(q', a_1, a_2), 1)$; in pending form emit $((a_1, a_2), \mathsf{N}(q', a_2), 0)$. If a marker is missing, refresh it before any further simulated transition. When a normal halting state is reached, complete all refreshes before emitting HALT.

In Theorem 10 we prove that these updates delete exactly one edge, restore both markers after at most two refresh tokens, and preserve the encoded queue and rear symbol. The two-symbol case uses one expansion and one slide; refreshes never change the length. Consequently the context bound is $S + 2$ and the token bound is four per HIST transition. The numerical argument in Theorem 11 completes the proof. □

Corollary 9 (Universality and resources)**.** *For every single-tape Turing machine there is one fixed transformer of the above architecture that simulates it on all encoded inputs. A run using time t and space s is simulated with $O((t+1)s)$ generated tokens, context length $O(s)$, and $O(\log(s+2))$-bit inference arithmetic. On input $\triangleright x \perp$, the configuration of the Turing machine at each step can be decoded from tokens generated by the CoT steps of the transformer with $E(\triangleright x \perp)$ as the prompt.*

Proof. Combine Theorems 5, 6 and 8. □

5 Conclusions and Limitations

We have shown that positional encoding is not necessary for the computational universality of Transformers in the sliding-window autoregressive regime. Although a Transformer without PE is permutation-invariant with respect to the tokens inside a single window, the temporal motion of the window provides an additional source of sequential structure. This FIFO-like structure is sufficient to simulate the dequeue operation of a Post machine, and hence to achieve Turing completeness.

Our result should not be interpreted as showing that Transformers without PE can recover exact token positions inside the context window. Our construction deliberately avoids positional addressing by introducing the HIST model (with edge encoding), where the transition depends only on a

finite control state and the histogram of the current window. The result, therefore, shows that exact positional access is stronger than what is needed for universal computation: a permutation-invariant window summary, together with finite control and sliding-window dynamics, already suffices.

An interesting future direction is to investigate whether the s-factor slowdown in our construction is inherent. Our result shows that positional encoding is not necessary for universal computation, but it remains unclear whether removing PE necessarily incurs an efficiency cost. Establishing either a faster simulation or a matching lower bound would further clarify the computational role of positional information in transformers. We have shown that positional encoding is not necessary for the computational universality of Transformers in the sliding-window autoregressive regime. Although a Transformer without PE is permutation-invariant with respect to the tokens inside a single window, the temporal motion of the window provides an additional source of sequential structure. This FIFO-like structure is sufficient to simulate the dequeue operation of a Post machine, and hence to achieve Turing completeness.

AI use statement In this work, we used generative AI tools for language polishing, literature search, and help drawing figures. We have reviewed all AI-assisted content and checked all the proofs. We take full responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics statement The work is theoretical and does not involve human subjects, private data, potentially harmful insights, methodologies and applications, or other activities that would require special ethical safeguards.

Reproducibility statement The paper does not include experiments; the main contributions are theoretical results and proofs. All assumptions are stated in the definitions and theorem statements. Complete proofs of the claims are also included in the paper.

REFERENCES

Anthropic. Introducing claude opus 4.7. Anthropic Announcements, 2026. https://www.anthropic.com/news/claude-opus-4-7.

Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

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*, pp. 455–475, 2020.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36:24892–24928, 2023.

Qian Li and Yuyi Wang. Constant bit-size transformers are turing complete. *Advances in Neural Information Processing Systems*, 38, 2025.

Qian Li and Yuyi Wang. Efficient turing machine simulation with transformers. In *The Fourteenth International Conference on Learning Representations, ICLR 2026*, 2026.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Thang Luong and Edward Lockhart. Advanced version of gemini with deep think officially achieves gold-medal standard at the International Mathematical Olympiad. Google DeepMind Blog, July 2025.

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

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023b.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

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

Franz Nowak, Anej Svete, Alexandra Butoi, and Ryan Cotterell. On the representational capacity of neural language models with chain-of-thought reasoning. *arXiv preprint arXiv:2406.14197*, 2024.

OpenAI. How codex ran openai devday 2025. OpenAI Developer Blog, 2025. https://developers.openai.com/blog/topic/codex.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is turing-complete. *J. Mach. Learn. Res.*, 22:75:1–75:35, 2021. URL https://jmlr.org/papers/v22/20-302.html.

Emil L Post. Finite combinatory processesformulation1. *The journal of symbolic logic*, 1(3):103–105, 1936.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

Ruizhong Qiu, Zhe Xu, Wenxuan Bao, and Hanghang Tong. Ask, and it shall be given: Turing completeness of prompting. *CoRR*, abs/2411.01992, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Marco Sälzer, Chris Köcher, Alexander Kozachinskiy, Georg Zetzsche, and Anthony Widjaja Lin. The counting power of transformers. *arXiv preprint arXiv:2505.11199*, 2025.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 464–468, 2018.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Chenxiao Yang, Nathan Srebro, David McAllester, and Zhiyuan Li. Pencil: Long thoughts with short memory. *arXiv preprint arXiv:2503.14337*, 2025.

Nikola Zubić, Federico Soldá, Aurelio Sulser, and Davide Scaramuzza. Limits of deep learning: Sequence modeling through the lens of complexity theory. *arXiv preprint arXiv:2405.16674*, 2024.

A Fixed ReLU Circuit and Correctness of the Transformer Simulation

We give all gates omitted from Section 4, then prove the semantic and numerical invariants. Throughout this section $\mathcal{C}$ denotes a transformer context and C denotes a queue-automaton queue.

A.1 EXPLICIT FINITE RELU REALIZATION

For $0 \le u \le 1$ and $I \in \{0,1\}$, the elementary gate

$$\mathrm{gate}(u, I) = [u + I - 1]_+ = uI \tag{8}$$

uses only an affine map and one ReLU. It multiplies a bounded scalar by a *bit*, not two arbitrary real inputs. Each I_ℓ is obtained by summing the newest-token coordinates whose vocabulary labels equal ℓ, and hence is an exact bit.

Using r, r_0, r_1, z_σ from (6)–(7), define the following finite collections of gates:

$$g_{q,b,\sigma} = [z_\sigma + I_{\mathsf{N}(q,b)} - 1]_+ \qquad (q \neq q_{\mathrm{halt}}), \tag{9}$$

$$p_{q,a_1,a_2} = [r + I_{\mathsf{P}(q,a_1,a_2)} - 1]_+, \tag{10}$$

$$f_{j,\ell} = [r_j + I_\ell - 1]_+ \qquad (j \in \{0,1\},\ \ell \in R), \tag{11}$$

$$h_b = [r + I_{\mathsf{N}(q_{\mathrm{halt}},b)} - 1]_+. \tag{12}$$

In a normal non-halting ready configuration with front σ and label $\mathsf{N}(q,b)$, the sole nonzero gate is $g_{q,b,\sigma} = \rho$. In a pending ready configuration, the sole nonzero gate is the corresponding $p_{q,a_1,a_2} = \rho$. If $\#_j$ is missing, the sole nonzero gate is $f_{j,\ell} = \rho$, where ℓ is the current label. In normal halting ready form the sole nonzero gate is $h_b = \rho$.

To verify these statements, first note that $r, r_j \in \{0, \rho\}$ and $0 \le z_\sigma \le r \le 1$. In normal ready form, the divergence identity gives $z_\sigma = \rho\mathbf{1}[x_1 = \sigma]$. In pending form the normal-label bits all vanish, disabling every g gate regardless of the values of z. In refresh form $r = z_\sigma = 0$, disabling all normal, pending, and halting gates. The remaining assertions follow from (8). The semantic proof below establishes that every context reached before halting is one of these cases.

Routing to the output. For $q \neq q_{\mathrm{halt}}$, define a finite table $T(q,b,\sigma) \in \mathcal{V}$ by

$$T(q,b,\sigma) = \begin{cases} ((b,a_1), \mathsf{N}(q',a_1), 0), & \delta_{\mathsf{PM}}(q,\sigma) = (q',a_1), \\ ((b,a_1), \mathsf{P}(q',a_1,a_2), 1), & \delta_{\mathsf{PM}}(q,\sigma) = (q',a_1a_2). \end{cases}$$

The desired logit vector in $\mathbb{R}^K$ is the fixed linear routing map

$$\begin{aligned} L = &\sum_{q \neq q_{\mathrm{halt}},b,\sigma} g_{q,b,\sigma}\mathbf{e}^K_{T(q,b,\sigma)} \\ &+ \sum_{q,a_1,a_2} p_{q,a_1,a_2}\mathbf{e}^K_{((a_1,a_2),\mathsf{N}(q,a_2),0)} \\ &+ \sum_{j,\ell} f_{j,\ell}\mathbf{e}^K_{(\#_j,\ell,0)} + \sum_b h_b\mathbf{e}^K_{\mathsf{HALT}}. \end{aligned} \tag{13}$$

All sums have a fixed finite number of terms determined by the queue automaton. Exactly one gate is active, so $L = \rho\mathbf{e}^K_{v^*}$, with v^* the prescribed output token. Set

$$f_{\mathrm{mlp}}(h) = (L(h), \mathbf{0}^N) - h. \tag{14}$$

Equations (6)–(14) use only fixed affine maps and finitely many ReLUs. Any needed identity wire through a hidden layer can be realized by $u = [u]_+ - [-u]_+$; a linear output layer realizes the final subtraction. Consequently these equations specify one finite multi-layer ReLU network with fixed integer weights and biases on the whole ambient space.

After the residual addition, $h_i^1 = (\rho\mathbf{e}^K_{v^*}, \mathbf{0}^N)$. With the tied embedding from Section 4,

$$\mathsf{emb}(v)^\top h_i^1 = \rho\mathbf{1}[v = v^*].$$

Thus the next token is uniquely v^*, with argmax gap $\rho > 0$. The construction never rescales this vector to an unscaled basis vector. In particular, as $m \to \infty$ the gap may approach zero, consistently with continuity of a fixed ReLU network. On unreachable contexts the same formulas define a total network; a fixed tie-breaking order defines its output if needed.

A.2 MARKER REFRESH AND SIMULATION INVARIANT

Lemma 10 (Two-marker refresh)**.** *Suppose $\mathcal{C}$ contains one copy of each marker and has edge projection W. Emit either one edge with bit 0, or two edges with bits $1, 0$, and let F be the emitted edge word. Thereafter, whenever marker $\#_j$ is missing, append $\#_j$ with bit 0 and unchanged control label. This requires at most two refresh steps, never loses both markers, and terminates with one copy of each marker and edge projection $\mathsf{tail}(WF)$. Exactly one edge is deleted over the entire operation.*

Proof. Let $\widehat{\mathcal{C}}$ be the old context followed by the one or two emitted edge tokens, before the final bit-0 deletion. The resulting context is $\mathsf{tail}(\widehat{\mathcal{C}})$. Since F is nonempty, $\widehat{\mathcal{C}}$ has a first edge; its preceding prefix consists of zero, one, or two distinct markers.

If the first deletion removes an edge, both markers remain and the operation is finished. Otherwise exactly one marker is missing. Refreshing that marker restores it at the rear while deleting the next symbol of the prefix of $\widehat{\mathcal{C}}$. If this is the other marker, only that other marker is now missing; one more refresh deletes the first edge and restores the second marker. Replacement markers were appended after this first edge, so they cannot be deleted before it. Thus at most two refreshes remove precisely the prefix through its first edge, restoring every removed marker and no others. The remaining edge projection is $\mathsf{tail}(WF)$. This proof also applies to $W = \epsilon$. □

Induction invariant. At the beginning of simulated step t, the context is ready, satisfies

$$\mathsf{edge}(\mathcal{C}) = E(C_t),$$

and has latest label $\mathsf{N}(q_t, b_t)$, where q_t and b_t are the queue-automaton state and rear symbol. The prompt (4) satisfies this invariant at $t = 0$. By (7), the circuit identifies the front at scale ρ and routes the correct transition.

For a one-symbol queue append a_1, the circuit emits (b_t, a_1), sets the label to $\mathsf{N}(q_{t+1}, a_1)$, and slides. For a two-symbol append a_1a_2, it emits (b_t, a_1) with expansion bit 1, preserving both markers, then emits (a_1, a_2) with bit 0. The pending gate ensures that the second edge is emitted without executing another queue transition. In either case the appended edge word is $F = E(b_t a)$.

Theorem 10 now restores both markers, preserves the new normal label, and gives

$$\mathsf{edge}(\mathcal{C}') = \mathsf{tail}(E(C_t)E(b_t a)) = E(C_{t+1}),$$

by Theorem 6. All intermediate contexts therefore have either both markers or exactly one missing; no other mode is reached. This establishes the induction invariant and the gate assumptions together.

If $q_{t+1} = q_{\mathrm{halt}}$, a pending second edge is still emitted and refreshes still have priority because every halting gate is multiplied by the ready mass r. Only in completed normal ready form does the circuit emit HALT. Conversely, no other configuration can emit it. This also covers an initially halting queue automaton.

Resource bounds. A one-symbol append uses at most three tokens and a two-symbol append at most four; add one token for halting. At a ready checkpoint the context length is $|W_t| + 2$. A one-symbol append and every refresh preserve length. A two-symbol append increases length by one at its first emission; already then its length equals $|W_{t+1}| + 2$, since $|W_{t+1}| = |W_t| + 1$. Thus a bound $|W_t| \le S$ at all completed simulated configurations also bounds every intermediate context by $S + 2$. These bounds concern generated tokens and active context, not arithmetic operation count for evaluating attention over that context.

A.3 A PRECISE LOGARITHMIC-PRECISION IMPLEMENTATION

Lemma 11 (Inference precision)**.** *Let $M \ge 2$ bound every context length in the simulation. Keep all model parameters fixed. There is a fixed-point evaluation of standard uniform softmax attention and the above ReLU circuit, using $O(\log M)$ bits per scalar (with constants depending only on the simulated machine), that generates exactly the same tokens as the real-arithmetic construction.*

Proof. Choose $b = \lceil \log_2 M \rceil + 1$ fractional bits and the dyadic grid $2^{-b}\mathbb{Z}$. Evaluate the zero logits and their exponentials exactly, sum the softmax denominator to the integer m, and round each attention weight $1/m$ to the nearest grid point, with a common fixed tie rule. This is a rounding of the standard softmax operation, not an extra division gate in the MLP. Denote the common rounded weight by a_m. Accumulate weighted values exactly on the same grid, with sufficient integer headroom and no saturation. For $2 \le m \le M$,

$$|a_m - 1/m| \le 2^{-b-1} \le \frac{1}{4M} \le \frac{1}{4m}, \qquad \frac{3}{4m} \le a_m \le 1, \qquad m a_m \le \frac{5}{4}.$$

Each value coordinate is zero or one, so the accumulated histogram is exactly

$$\widetilde{A}(\tau) = \mathrm{cnt}_{\mathcal{C}}(\tau) a_m.$$

All coordinates have the same scale, including the two marker coordinates. Thus (6) gives exactly one active mode of mass a_m, and in normal ready form the divergence cancellation gives

$$\widetilde{D}_\sigma + [\widetilde{r} + B_\sigma - 1]_+ = a_m \mathbf{1}[x_1 = \sigma].$$

Every routing gate and the output calculation are consequently exact with a_m in place of ρ.

All MLP weights and biases are integers. Affine operations and ReLUs preserve the dyadic grid, so no further fractional rounding error is introduced. Histogram partial sums are bounded by $5/4$, and all intermediate values in the fixed circuit are bounded by a constant depending only on that circuit. The attention denominator requires $\lceil \log_2(M+1) \rceil$ integer bits; the other computations require only fixed additional headroom. Hence a common signed fixed-point format with $b + O(\log M)$ total bits suffices. The winning output gap is exactly $a_m > 0$.

Finally, generated tokens are discrete and are re-embedded exactly at the next step. No hidden real-valued state or rounding error is propagated from one generation step to the next. The common scale may change with m, without changing the proof. □

The order of rounding and accumulation in Theorem 11 is part of the numerical specification. Independently rounding the final normalized histogram coordinates need not preserve the cancellation identity. The lemma establishes a sufficient $O(\log M)$-bit implementation; it does not claim correctness for every rounding convention or a fixed precision valid for arbitrarily long runs. In exact real arithmetic, one fixed transformer works for every finite prefix without supplying a space bound to its weights or its MLP.