Characterizing the Expressivity of Fixed-Precision Transformer Language Models

Jiaoda Li **Ryan Cotterell**
{jiaoda.li, ryan.cotterell}@inf.ethz.ch
ETH zürich

Abstract

Transformer-based language models (LMs) have achieved widespread empirical success, but their theoretical expressive power remains only partially understood. In this work, we analyze a restricted idealization of fixed-precision transformers with strict future masking, soft attention, and no positional encodings. We establish that this class of models is exactly as expressive as a specific fragment of linear temporal logic that contains only a single temporal operator: the `past` operator. We further connect this fragment to established classes in formal language theory, automata theory, and algebra, yielding a unified framework for understanding transformer expressivity under this idealization. Finally, we present empirical results that align closely with our theory: transformers trained on languages within their characterized expressive capacity generalize reliably across sequence lengths, while they consistently fail to generalize on languages beyond it.[1]

1 Introduction

Transformer-based language models (LMs) have demonstrated remarkable empirical success [46, 36, 12] on a wide variety of natural language tasks [47, 19, 41, *inter alia*]. This success has sparked growing interest in understanding the theoretical expressive power of transformers, i.e., what languages they can and cannot recognize, and, by extension, what tasks they can and cannot perform. A significant body of work approaches this question by relating transformers to well-established frameworks such as formal languages, logic, and circuit complexity [18, 31, 50, 42]. To facilitate their theoretical analysis, theoreticians often propose idealizations of transformers. For instance, while practical implementations of transformers operate under fixed precision, e.g., single (32-bit) or half (16-bit) precision, many authors assume arbitrary [38, 18, 34] or length-dependent precision [32, 7]. Although such idealizations capture key aspects of transformers, they tend to overestimate their expressive power [38].

A recent step toward a more faithful theoretical understanding of the expressive power of transformers comes from Yang et al. [50], who show that fixed-precision transformers with strict future masking and unique hard attention (UHA) are exactly as expressive as linear temporal logic $\mathbf{LTL}[\mathbf{P}, \mathbf{F}, \mathbf{S}, \mathbf{U}]$, which includes four temporal operators: $\mathbf{P}$ (`past`), $\mathbf{F}$ (`future`), $\mathbf{S}$ (`since`), and $\mathbf{U}$ (`until`). However, UHA still deviates from the soft attention used in practice. To address this gap, Yang and Chiang [49] analyze fixed-precision transformers with strict future masking and soft attention, an idealization that most closely reflects the models deployed in real-world applications. Yang and Chiang [49] show that such models are upper bounded by C-RASP, a counting-based programming language, though a precise characterization of these models' expressivity remains open.

In this paper, we close this gap by providing an exact characterization of the expressive power of fixed-precision transformers with soft attention, strict masking, and no positional encodings (NoPE). We show they are precisely characterized by $\mathbf{LTL}[\mathbf{P}]$, a restricted fragment of $\mathbf{LTL}[\mathbf{P}, \mathbf{F}, \mathbf{S}, \mathbf{U}]$ that

[1] Code available at GitHub repository.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

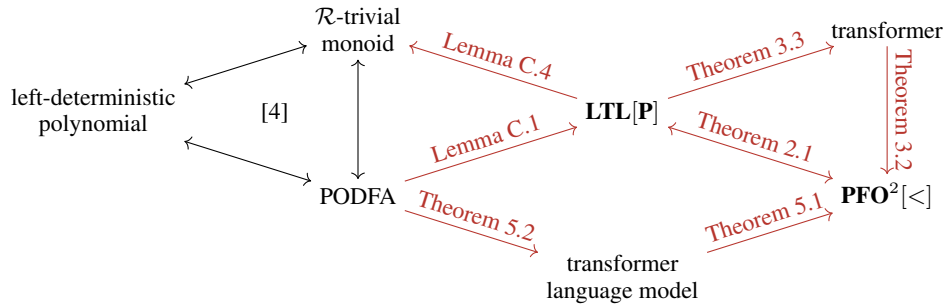


Figure 1: Roadmap of the paper. Red arrows indicate novel results.

uses only the `past` operator (**P**). We further demonstrate that **LTL**[**P**] is equivalent in expressivity to partially ordered deterministic finite automata (PODFAs), which are characterized by $\mathcal{R}$-trivial monoids and recognize left-deterministic polynomials. These results offer a detailed and principled characterization of the expressive power of this idealization, delineating its strengths and limitations. Crucially, our findings imply that many simple languages, e.g., bounded Dyck languages, which have been shown to be recognizable under more permissive idealizations, are beyond the reach of the models we study. We also extend our theoretical results to transformer LMs, showing that their expressivity matches that of transformer recognizers. A visual overview of the theoretical landscape is provided in Fig. 1.

To arrive at a compelling theory, it is essential to show that it faithfully reflects the behavior of models trained under standard machine learning paradigms. To this end, we provide empirical evidence using the length generalization framework, a widely used method for gauging neural network expressivity [10, 6, 22]. We construct a suite of languages spanning a fine-grained hierarchy of formal language classes. Our results (Tab. 1) exhibit strong alignment between theory and practice: for all languages that transformers are predicted to recognize, the models generalize perfectly over lengths (100% accuracy); for languages beyond their theoretical capacity, they consistently make generalization errors, regardless of learning rates or random seeds.

2 Background

In this section, we present the necessary background knowledge that underpins our analysis.

2.1 Strings and Languages

An **alphabet**, denoted as Σ, is a finite, non-empty set of symbols. A **string** over Σ is a *finite* sequence of symbols drawn from Σ. The set of all strings over Σ is denoted by its Kleene star Σ^*. A subset of Σ^* is called a **language**. A **regular expression** is a declarative way to describe a language, defined recursively as follows:

- $\emptyset$ and each $\mathtt{a} \in \Sigma$ are regular expressions;
- If α and β are regular expressions, so are the union $\alpha + \beta$, concatenation $\alpha\beta$, the Kleene star α^*, and complement α^{c}.

A language is **regular** if and only if it can be described by a regular expression [26]. A regular language is said to be **star-free** if it can be described by a regular expression without the Kleene star [28]. As an example, Σ^*, the set of all strings over Σ, is star-free, as it can be described by $\emptyset^{\mathsf{c}}$.

2.2 LTL[P]

Linear temporal logic **LTL**[**P**, **F**, **S**, **U**] [25] is a modal logic, with modalities referring to time. The full definition is given in §A.1. In this paper, we define a fragment of **LTL**[**P**, **F**, **S**, **U**]—denoted as **LTL**[**P**]—that includes only one temporal operator **P** (`past`). **Formulas** in **LTL**[**P**] are composed of **atomic formulas** $\pi_{\mathtt{a}}$ for every $\mathtt{a} \in \Sigma$, Boolean connectives $\wedge, \neg$, and a temporal operator **P**. The

disjunction $\vee$ is definable in terms of $\wedge$ and $\neg$ as:

$$\psi_1 \vee \psi_2 \stackrel{\text{def}}{=} \neg(\neg\psi_1 \wedge \neg\psi_2). \tag{1}$$

When defined over strings, the formulas are interpreted with respect to a string $\mathbf{w} = \mathtt{w}_1 \cdots \mathtt{w}_N$ at a position $n \in \{1, \ldots, N\}$. The semantics are defined inductively. Given a string $\mathbf{w}$, a position n, and a formula ψ, we define $\mathbf{w}, n \models \psi$ ($\mathbf{w}$ satisfies ψ at position n) as follows:

$\mathbf{w}, n \models \pi_{\mathtt{a}}$	if and only if	the n^{th} symbol of $\mathbf{w}$ is an $\mathtt{a}$.
$\mathbf{w}, n \models \psi_1 \wedge \psi_2$	if and only if	$\mathbf{w}, n \models \psi_1$ and $\mathbf{w}, n \models \psi_2$.
$\mathbf{w}, n \models \neg\psi$	if and only if	$\mathbf{w}, n \not\models \psi$.
$\mathbf{w}, n \models \mathbf{P}\psi$	if and only if	there exists m such that $m < n$ and $\mathbf{w}, m \models \psi$.

Example:

- $\mathtt{abb}, 1 \models \pi_{\mathtt{a}}$ as the first symbol is $\mathtt{a}$.
- $\mathtt{abb}, 3 \models \mathbf{P}\pi_{\mathtt{a}}$ as an $\mathtt{a}$ occurs before position 3.

To say a string $\mathbf{w}$ of length N satisfies a formula ψ, written as $\mathbf{w} \models \psi$, we evaluate from position $N + 1$ (the position after the final symbol):

$$\mathbf{w} \models \psi \quad \text{if and only if} \quad \mathbf{w}, N + 1 \models \psi. \tag{2}$$

The language defined by a formula ψ is the set $\mathbb{L}(\psi)$ of all strings that satisfy ψ.

Example:

$\mathbf{P}\mathtt{a}$ defines all the strings that contain at least one occurrence of $\mathtt{a}$, i.e.,

$$\mathbb{L}(\mathbf{P}\mathtt{a}) = \Sigma^* \mathtt{a} \Sigma^*. \tag{3}$$

2.3 $\mathbf{PFO}^2[<]$

First-order logic $\mathbf{FO}[<]$ can also be used to describe formal languages. In contrast to (linear) temporal logic, where formulas are interpreted at a single position, $\mathbf{FO}[<]$ formulas may contain multiple position variables, making it better-suited to simulating mechanisms in transformers that involve more than one position, such as attention; see §§ B.2 and 3.2. In this paper, we introduce a fragment of $\mathbf{FO}[<]$ called $\mathbf{PFO}^2[<]$—the past fragment of first-order logic with two variables. The atomic formulas of $\mathbf{PFO}^2[<]$ consist of unary predicates $\pi_{\mathtt{a}}$ for each $\mathtt{a} \in \Sigma$ and a binary numerical predicate $<$ that can be used to determine the sequential order between variables. Formulas in $\mathbf{PFO}^2[<]$ can have at most two distinct variables, x and y. In addition to the usual Boolean connectives $\wedge$ and $\neg$, there is a bounded existential quantifier $\exists y < x$, where y is the variable bounded by the quantifier, and x is the free variable. In case there is no free variable, the bounded existential quantifier behaves like an ordinary existential quantifier. The formulas in $\mathbf{PFO}^2[<]$ are constructed inductively as follows:

- Every atomic formula is a formula;
- A Boolean combination of formulas is a formula if it does not contain more than two variables;
- If $\phi(x, y)$ is a formula with two variables x, y, $\exists y < x : \phi(x, y)$ and $\exists x < y : \phi(x, y)$ are formulas;
- If $\phi(x)$ is a formula with one variable x, $\exists x < y : \phi(x)$ is a formula;
- If $\phi(x)$ is a formula with one variable x, $\exists x : \phi(x)$ is a formula.

The bounded universal quantifier $\forall y < x$ can be defined using $\exists y < x$ and $\neg$. For instance:

$$\forall y < x : \phi(x, y) \stackrel{\text{def}}{=} \neg\exists y < x : \neg\phi(x, y). \tag{4}$$

A formula with no free variables is called a **sentence**. We write $\mathbf{w} \models \phi$ if ϕ is satisfied for $\mathbf{w}$. A sentence ϕ defines a language, the set of strings over Σ satisfying it, denoted as $\mathbb{L}(\phi)$.

Example:
The sentence below defines all the strings that contain $\texttt{abc}$ as a subsequence (symbols appearing in order, possibly non-contiguously):

$$\exists x : (\pi_{\texttt{c}}(x) \wedge \exists y < x : (\pi_{\texttt{b}}(y) \wedge \exists x < y : \pi_{\texttt{a}}(x))) \,. \tag{5}$$

Note that the variable x is reused, as permitted in $\mathbf{PFO}^2[<]$. The formulas below, however, are not definable in $\mathbf{PFO}^2[<]$:

- $x < y < z$ (more than two variables);
- $\forall x : (\exists y \geq x : \pi_{\texttt{a}}(y))$ (disallowed use of the quantifier $\exists y \geq x$).

$\mathbf{PFO}^2[<]$ is specifically designed to match the expressive power of $\mathbf{LTL}[\mathbf{P}]$, allowing us to leverage both formalisms in our analysis of transformer expressivity.

Theorem 2.1. $\mathbf{PFO}^2[<]$ *and* $\mathbf{LTL}[\mathbf{P}]$ *have the same expressive power.*

Proof. See §A.2.3. ■

3 Transformers

In this section, we formally define the transformer idealization under consideration and establish two central expressivity results: (i) every transformer can be translated into an equivalent $\mathbf{PFO}^2[<]$ formula, and (ii) every $\mathbf{LTL}[\mathbf{P}]$ formula can be simulated by a transformer. Combined with Theorem 2.1, these results imply that transformers are expressively equivalent to both $\mathbf{LTL}[\mathbf{P}]$ and $\mathbf{PFO}^2[<]$.

3.1 The Transformer Architecture

The transformer considered in this section follows the architecture described by Yang et al. [50], with the exception that we use **soft attention** in place of unique hard attention (UHA). Before formally defining the architecture, we first state explicitly the assumptions that will govern our model as follows.

Assumption 3.1. *We assume that the transformer satisfies the following conditions:*

1. ***Fixed precision****: There exists a finite set of values that the floating-point numbers can assume. We denote this set as* $\mathbb{F} = \{f_1, f_2, \ldots\}$.

2. ***No positional encoding*** *(NoPE): Positional encodings are omitted for now, but are addressed in §F.3.*

3. ***Soft attention****: Attention weights are computed using the* $\mathrm{softmax}$ *function. We also treat average hard attention (AHA) in §F.1, showing that AHA is equally expressive as soft attention.*

4. ***Strict future masking****: As in Yang et al. [50], the attention is strictly future-masked. Strict masking is shown to be more expressive than non-strict masking in §F.2.*

5. ***Recognition****: Following standard practice in expressivity studies [42], the transformer is treated as a language recognizer. We will extend the analysis to transformer language models in §5.*

Transformers take strings as input. Given a string $\mathbf{w} = \mathtt{w}_1 \cdots \mathtt{w}_N$ over an alphabet Σ, we append a special end-of-sentence symbol $\textsc{eos} \notin \Sigma$ to form a **completed** string $\overline{\mathbf{w}} \stackrel{\text{def}}{=} \mathtt{w}_1 \cdots \mathtt{w}_N \textsc{eos}$ over the extended alphabet $\overline{\Sigma} \stackrel{\text{def}}{=} \Sigma \cup \{\textsc{eos}\}$. An initial embedding layer $\mathbf{E} \colon \overline{\Sigma}^{N+1} \to \mathbb{F}^{D \times (N+1)}$ maps each input symbol including $\textsc{eos}$ to a D-dimensional representation. We define the embedding layer as the 0^{th} layer of the transformer:

$$\mathbf{H}^{(0)} \stackrel{\text{def}}{=} \mathbf{E}. \tag{6}$$

The embedding is followed by a stack of L hidden layers. Each hidden layer contains two sublayers: a self-attention mechanism $\mathbf{A}^{(\ell)} \colon \mathbb{F}^{D \times (N+1)} \to \mathbb{F}^{D \times (N+1)}$, and a feedforward network $\mathbf{F}^{(\ell)} \colon \mathbb{F}^{D \times (N+1)} \to \mathbb{F}^{D \times (N+1)}$. These components are composed sequentially. For each

$\ell \in \{0, 1, \ldots, L-1\}$, we define:

$$\mathbf{H}^{(\ell+0.5)} \stackrel{\text{def}}{=} \text{LN}\left(\mathbf{A}^{(\ell)}\left(\mathbf{H}^{(\ell)}\right) + \mathbf{H}^{(\ell)}\right), \tag{7a}$$

$$\mathbf{H}^{(\ell+1)} \stackrel{\text{def}}{=} \text{LN}\left(\mathbf{F}^{(\ell)}\left(\mathbf{H}^{(\ell+0.5)}\right) + \mathbf{H}^{(\ell+0.5)}\right). \tag{7b}$$

where LN denotes layer normalization [1]. Finally, a classification head $\text{C}\colon \mathbb{F}^D \to \mathbb{F}$ maps the representation at the EOS position in the final layer to a scalar output:

$$o(\overline{\mathtt{w}}) \stackrel{\text{def}}{=} \text{C}\left(\mathbf{H}^{(L)}(\overline{\mathtt{w}})_{:,N+1}\right). \tag{8}$$

We use $\mathbf{H}^{(L)}(\overline{\mathtt{w}})_{d,n}$ to denote the $(d,n)^{\text{th}}$ entry in the matrix. Similarly, $\mathbf{H}^{(L)}(\overline{\mathtt{w}})_{:,n}$ refers to the column vector at position n, and $\mathbf{H}^{(L)}(\overline{\mathtt{w}})_{d,:}$ refers to the row vector corresponding to dimension d. Consequently, a transformer defines a function of type $\Sigma^{N+1} \to \mathbb{F}$, where N is a parameter of the type. We say a string $\mathtt{w}$ is accepted by the transformer if $o(\overline{\mathtt{w}}) > 0$. A detailed specification of the architecture is provided in §B.1.

3.2 From Transformers to $\mathbf{PFO}^2[<]$

In this subsection, we discuss the following theorem.

Theorem 3.2. *Every transformer can be simulated by* $\mathbf{PFO}^2[<]$.

Proof. See §B.2. ■

Our proof closely follows the approach of Chiang et al. [9, Section 5], with one key difference: the simulation of summation in soft attention. While prior work makes use of counting quantifiers to simulate summation, Li et al. [27] demonstrate that summation with iterative rounding can be simulated with $\mathbf{FO}[<]$. We take this a step further by showing that the specific summation involved in soft attention can be simulated by $\mathbf{PFO}^2[<]$. Here, we provide a high-level overview of the proof. We identify two summations in soft attention. The first occurs in the denominator of the $\mathrm{softmax}$ function, defined for a D-dimensional vector $\mathbf{x}$, as follows:

$$\mathrm{softmax}(\mathbf{x})_d \stackrel{\text{def}}{=} \frac{\exp(x_d)}{\sum_{i=1}^{D} \exp(x_i)}, \text{ for } d \in \{1, \ldots, D\}. \tag{9}$$

$\exp$ is a deterministic function. Under the fixed precision assumption, the possible input and output values of $\exp$ form a finite set, allowing $\mathbf{PFO}^2[<]$ formulas to encode $\exp$ by explicit enumeration of these values. Since $\exp$ outputs only non-negative values, the summation in the denominator reduces to a sum of non-negative terms, which can be simulated by $\mathbf{PFO}^2[<]$ (Lemma B.5). Additionally, the second summation is a weighted sum, in which the weights are produced by the $\mathrm{softmax}$ function. Under fixed precision, the output of a $\mathrm{softmax}$ contains a bounded number of non-zero entries (Proposition B.6). Therefore, the weighted sum involves only a bounded number of terms and can similarly be simulated by $\mathbf{PFO}^2[<]$ (Lemma B.7).

3.3 From $\mathbf{LTL}[\mathbf{P}]$ to Transformers

In this subsection, we discuss the following theorem.

Theorem 3.3. *Every* $\mathbf{LTL}[\mathbf{P}]$ *formula can be simulated by a transformer.*

Proof. See §B.3. ■

Our proof is adapted from that of Yang et al. [50, Appendix C.1]. Intuitively, each $\mathbf{LTL}[\mathbf{P}]$ formula is encoded in a dedicated coordinate $d \in \{1, \ldots, D\}$ of the transformer's hidden state. At each position, this coordinate takes the value 1 if the formula is satisfied at that position, and 0 otherwise. The key challenge in proving Theorem 3.3 lies in simulating the temporal operator $\mathbf{P}$. Prior constructions typically employ uniform attention, which assigns equal weight to all preceding positions, to simulate such operators. However, under fixed precision, soft attention has a limited attention span, as the

output of the $\mathrm{softmax}$ function contains only a bounded number of non-zero entries. We refer to this bound as the maximum attention span, denoted by $N_{\max}$. To overcome this limitation, previous work has relied on non-fixed numerical precision—such as arbitrary precision [9, 49] or log precision [31]—to prevent attention weights from vanishing. In contrast, we present a construction that overcomes this issue without requiring increased numerical precision.

We begin by describing a base construction that applies when attention weights do not vanish, and then extend it to handle cases where vanishing weights may occur.

Base Construction. Suppose a formula ψ is simulated by coordinate d_1 at the ℓ^{th} layer, i.e., by $\mathbf{H}^{(\ell)}_{d_1,:}$ (omitting the input string $\overline{\mathbf{w}}$ for brevity), and let $\mathbf{P}\psi$ be the formula we aim to simulate. We construct an attention sublayer that uniformly attends to all previous positions $m < n$ such that $\mathbf{H}^{(\ell)}_{d_1,m} = 1$. If such positions exist, we set an unused coordinate $d_{\mathbf{P}}$ to 1, i.e., $\mathbf{H}^{(\ell+0.5)}_{d_{\mathbf{P}},n} = 1$ and otherwise to 0.

Handling Vanishing Weights. However, when too many (more than $N_{\max}$) previous positions satisfy ψ, the resulting attention weights will underflow to 0. This results in $\mathbf{H}^{(\ell+0.5)}_{d_{\mathbf{P}},n} = 0$ even when $\mathbf{P}\psi$ is true, leading to incorrect behavior. To address this, we first compute the logical conjunction of d_1 and $d_{\mathbf{P}}$ and store it in $d_\wedge$, i.e.,

$$\mathbf{H}^{(\ell+1)}_{d_\wedge,n} = \begin{cases} 1 & \text{if } \mathbf{H}^{(\ell)}_{d_1,n} = 1 \text{ and } \mathbf{H}^{(\ell+0.5)}_{d_{\mathbf{P}},n} = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

The number of positions n with $\mathbf{H}^{(\ell+1)}_{d_\wedge,n} = 1$ is bounded by $N_{\max}$, since beyond the $(N_{\max}+1)^{\text{th}}$ position, the first attention sublayer will suffer from vanishing attention weights, causing $\mathbf{H}^{(\ell+0.5)}_{d_{\mathbf{P}},n} = 0$.

Next, we construct a second layer in which the attention sublayer uniformly attends to all positions $m < n$ where $\mathbf{H}^{(\ell+1)}_{d_\wedge,m} = 1$. This produces a coordinate $d_{\mathbf{PP}}$ that simulates $\mathbf{P}(\psi \wedge \mathbf{P}\psi)$. Since the number of positions satisfying $\mathbf{H}^{(\ell+1)}_{d_\wedge,m} = 1$ is bounded, this second attention sublayer is not subject to vanishing attention weights and therefore computes its output reliably.

The formula $\mathbf{P}(\psi \wedge \mathbf{P}\psi)$ effectively asks whether there are at least two positions before n that satisfy ψ. Thus, if there is exactly one position $m < n$ satisfying ψ, then $\mathbf{P}(\psi \wedge \mathbf{P}\psi)$ differs from the target formula $\mathbf{P}\psi$. In this case, however, the first attention sublayer (producing $d_{\mathbf{P}}$) correctly simulates $\mathbf{P}\psi$, since the attention only needs to cover one position. We therefore take the logical disjunction of $d_{\mathbf{P}}$ and $d_{\mathbf{PP}}$, implemented via the subsequent feedforward sublayer. The resulting coordinate $d_\vee$ correctly simulates the formula $\mathbf{P}\psi$. This completes the proof sketch.

Example:
Assume the attention span is limited to 1 position. The following example illustrates the simulation process, with defective simulations highlighted in red.

formula	coordinate	position				
ψ	d_1	1	0	1	0	1
$\mathbf{P}\psi$	$d_{\mathbf{P}}$	0	1	1	0	0
$\psi \wedge \mathbf{P}\psi$	$d_\wedge$	0	0	1	0	0
$\mathbf{P}(\psi \wedge \mathbf{P}\psi)$	$d_{\mathbf{PP}}$	0	0	0	1	1
$\mathbf{P}\psi = \mathbf{P}(\psi \wedge \mathbf{P}\psi) \vee \mathbf{P}\psi$	$d_\vee$	0	1	1	1	1

4 Characterizations of LTL[P]

We have established the expressive equivalence between transformers and **LTL**[**P**]. This connection becomes especially compelling when paired with precise and rich characterizations of **LTL**[**P**]. To that end, we prove the following theorem.

Theorem 4.1. *Let $\mathbb{L} \subseteq \Sigma^*$ be a regular language, $\mathbb{M}$ be its syntactic monoid, and $\mathcal{A}$ be the DFA accepting it. The following assertions are equivalent: (1) $\mathbb{L}$ is a left-deterministic polynomial, (2) $\mathbb{M}$ is $\mathcal{R}$-trivial, (3) $\mathcal{A}$ is partially ordered, and (4) $\mathbb{L}$ is definable by* **LTL**[**P**].

Proof. See §C. ■

In the remainder of this section, we focus on the PODFA characterization, which will be central to later developments. The other characterizations are defined in §C, which also includes a discussion of superclasses of $\mathbf{LTL}[\mathbf{P}]$, summarized in Tab. 2.

Definition 4.2. *A **deterministic finite automaton** (DFA) is a 5-tuple $\mathcal{A} = (\Sigma, Q, q_I, F, \delta)$ where*

- *Σ is an alphabet;*
- *Q is a finite set of states;*
- *$q_I \in Q$ is the initial state;*
- *$F \subseteq Q$ is the set of final states;*
- *$\delta \colon Q \times \Sigma \to Q$ is a total transition function;*
- *$q_R \in Q \setminus F$ is a rejecting sink state, i.e., $\delta(q_R, \mathtt{a}) = q_R$ for every $\mathtt{a} \in \Sigma$.*

Given an automaton $\mathcal{A} = (\Sigma, Q, q_I, F, \delta)$, we say $\mathcal{A}$ is in state q upon reading a string $\mathbf{w} = \mathtt{w}_1 \cdots \mathtt{w}_N \in \Sigma^*$ if and only if there exists a sequence of states $q_0, \ldots, q_N$ such that

- $q_0 = q_I$;
- $q_{n+1} = \delta(q_n, \mathtt{w}_{n+1})$ for $n \in [N-1]$;
- $q_N = q$.

A string $\mathbf{w}$ is accepted by $\mathcal{A}$ if it reaches one of the final states upon reading $\mathbf{w}$.

Definition 4.3. *A DFA is said to be **partially ordered** if there exists a partial order $\preceq$ on Q such that for every state $q \in Q$ and symbol $\mathtt{a} \in \Sigma$, we have $q \preceq \delta(q, \mathtt{a})$.*

Informally, in a partially ordered DFA, once the automaton leaves a state, it never revisits it. An example of a non-partially ordered DFA is provided in §C.3.

5 Transformer Language Models

In this section, we extend our analysis to transformer LMs. Formally, an LM constitutes a distribution over Σ^*, typically factorized autoregressively as follows:

$$p(\mathbf{w}) \stackrel{\text{def}}{=} \overrightarrow{p}(\text{EOS} \mid \mathbf{w}) \prod_{n=1}^{N} \overrightarrow{p}(\mathtt{w}_n \mid \mathbf{w}_{<n}). \tag{11}$$

where $\mathbf{w}_{<n} \stackrel{\text{def}}{=} \mathtt{w}_1 \cdots \mathtt{w}_{n-1}$. We define an LM p's prefix probability p_{prefix} as:

$$p_{\text{prefix}}(\mathbf{w}) \stackrel{\text{def}}{=} \sum_{\mathbf{w}' \in \Sigma^*} p(\mathbf{w}\mathbf{w}'). \tag{12}$$

To adapt a transformer for language modeling, we feed the input string $\mathbf{w}$ into the model autoregressively, and the classification head is replaced with a language modeling head $\mathbf{L} \colon \mathbb{F}^D \to \mathbb{F}^{|\overline{\Sigma}|}$, which maps the representation in the last layer L at the most recent position $n-1$ to a probability distribution over $\overline{\Sigma}$. This defines the local distribution $\overrightarrow{p}$ as follows:

$$\overrightarrow{p}\left(\mathtt{w}_n \mid \mathbf{w}_{<n}\right) \stackrel{\text{def}}{=} \mathbf{L}\left(\mathbf{H}^{(L)}(\mathbf{w}_{<n})_{:,n-1}\right)_{\mathtt{w}_n}. \tag{13}$$

where $\mathbf{L}(\mathbf{H}^{(L)}(\mathbf{w}_{<n})_{:,n-1})_{\mathtt{w}_n}$ denotes the probability assigned to the symbol $\mathtt{w}_n$. Under fixed precision, this distribution may not be perfectly normalized, as the sum of its components can deviate from 1 due to rounding errors.

As transformer LMs do not introduce any new computational components beyond those already present in transformer recognizers, the same constructions and proofs apply. Therefore, every transformer LM can be simulated by $\mathbf{PFO}^2[<]$.

Theorem 5.1. *Every transformer LM can be simulated by* $\mathbf{PFO}^2[<]$.

Proof. See §D. ■

The converse direction is more subtle. Yang and Chiang [49] show how to compile a transformer LM from C-RASP. Here, we present an alternative approach that is more intuitive. Our construction leverages the PODFA characterization of $\mathbf{LTL}[\mathbf{P}]$. Given a PODFA $\mathcal{A} = (\Sigma, Q, q_I, F, \delta)$, we can define a corresponding LM $p_{\mathcal{A}}$ by specifying the local distribution $\overrightarrow{p}_{\mathcal{A}}$ as follows: Suppose $\mathcal{A}$ is in state q upon reading the prefix $\mathbf{w}_{<n}$:

- If $q \notin F$ and $q \neq q_R$, $\overrightarrow{p}_{\mathcal{A}}(\mathtt{a} \mid \mathbf{w}_{<n})$ assigns uniform probability over all symbols in $\{\mathtt{a} \mid \delta(q, \mathtt{a}) \in Q \setminus \{q_R\}\}$;
- If $q \in F$, $\overrightarrow{p}_{\mathcal{A}}(\mathtt{a} \mid \mathbf{w}_{<n})$ assigns uniform probability over all symbols in $\{\mathtt{a} \mid \delta(q, \mathtt{a}) \in Q \setminus \{q_R\}\} \cup \{\text{EOS}\}$;
- If $q = q_R$, $\overrightarrow{p}_{\mathcal{A}}(\mathtt{a} \mid \mathbf{w}_{<n}) = 0$ for all $\mathtt{a} \in \overline{\Sigma}$, i.e., $p_{\text{prefix}}(\mathbf{w}_{<n}) = 0$.

Note that in practice, a $\mathrm{softmax}$ cannot produce an all-zero vector. For technical convenience, we therefore extend the $\overline{\Sigma}$ with a special symbol $\text{UNK} \notin \overline{\Sigma}$, which receives all the probability mass when $\mathcal{A}$ reaches the rejecting state, i.e., $\overrightarrow{p}_{\mathcal{A}}(\text{UNK} \mid \mathbf{w}_{<n}) = 1$ if $q = q_R$. Accordingly, we extend the LM head $\mathbf{L}$ to accommodate this special symbol: $\mathbf{L} \colon \mathbb{F}^D \to \mathbb{F}^{|\overline{\Sigma} \cup \{\text{UNK}\}|}$.

Theorem 5.2. *Every PODFA LM can be simulated by a transformer LM.*

Proof. See §D. ■

Intuitively, for each state $q \in Q$, we can construct a $\mathbf{LTL}[\mathbf{P}]$ formula ψ_q such that $\mathbf{w}_{<n}, n-1 \models \psi_q$ if and only if the automaton is in state q after reading the prefix $\mathbf{w}_{<n}$. By Theorem 3.3, each formula ψ_q can be encoded into a designated coordinate d_q of the transformer's hidden state. The language modeling head, extended to include the special token UNK, then maps these coordinates to the corresponding probability distribution specified by $\overrightarrow{p}_{\mathcal{A}}$.

6 Experiments

The experiments are divided into two parts: language recognition and language modeling.

6.1 Language recognition

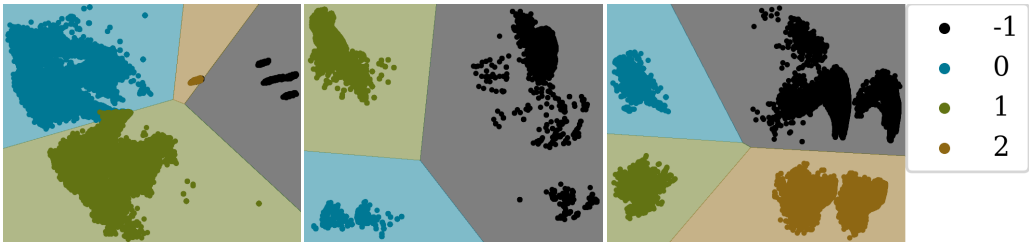
We have shown that a transformer can recognize only left-deterministic polynomials. In this section, we conduct a series of language classification experiments to empirically validate this claim. We consider five language classes, arranged in a strict inclusion hierarchy—each class is a proper subset of the one preceding it. For each class, we select one or more representative languages. These languages are listed in Tab. 1, and their detailed definitions are provided in §E.1.2.

6.1.1 Experimental setup

We implement the transformer that we theorize about (Assumption 3.1). For comparison, we also train a long short-term memory (LSTM) [20]. Models are trained on strings up to length 40, and tested on strings of length $41 - -500$. Each experiment is run with 5 different random seeds and 3 learning rates. We consider a transformer to have successfully recognized a language if it achieves 100% accuracy in at least one of the runs. Details of the experimental setup and model configurations are provided in §E.1.1.

6.1.2 Results

We compute classification accuracy and report both the maximum and mean values across all runs in Tab. 1. The LSTM achieves perfect accuracy on all tasks, consistent with previous work showing that LSTMs can recognize regular languages [29] and implement counting mechanisms [48]. This confirms that the tasks are learnable given the available training data. Results on transformers align



(a) RDP-1 (b) LDP-1 (c) LDP-2

Figure 2: Visualization of the transformer's representations for different DFA states. -1 denotes the rejecting state q_R. The filled contours represent the decision boundaries of a linear classifier.

precisely with our theoretical predictions: under fixed precision, transformers with soft attention and NoPE can recognize exactly the class of left-deterministic polynomials. They achieve perfect accuracy on all $\mathbf{LTL}[\mathbf{P}]$-definable languages but consistently fail on tasks outside this class, even though prior work has shown that some of these languages are theoretically recognizable under more permissive idealizations [9, 31, 49, 51, 44]. Notably, we use single-precision (32-bit) floating-point numbers, and the string lengths never exceed the maximum attention span of the transformer. That is, attention can uniformly cover all prior positions without numerical underflow or overflow. Yet, despite these favorable conditions, the transformer exhibits no expressive power beyond what is predicted by our formal characterization. A more detailed breakdown of the results can be found in §E.1.3.

6.2 Language modeling

We now turn to experiments on language modeling, focusing on three representative languages: RDP-1, LDP-2, and LDP-1. The corresponding automata are illustrated in Fig. 3. We train the transformer LMs using the standard cross-entropy loss. For evaluation, a predicted symbol is considered correct if it has non-zero probability under the target distribution $\overrightarrow{p}_{\mathcal{A}}$ induced by the DFA. Per-token accuracies are reported in Tab. 3. The transformer LM successfully learns LDP-1 and LDP-2 with perfect accuracy. For RDP-1, the best performance reaches 98.3%, but the model consistently falls short of achieving 100.0%. This gap becomes more evident upon inspecting the hidden states of the model.

Table 1: Language recognition experiments. Language classes are ordered by decreasing complexity. For each class, examples are chosen to be minimally more complex than those in the immediately lower class. Maximum and mean accuracies (± standard deviation) are reported. Exact values of 100.0% accuracy are highlighted in bold.

Class	Language	Transformer		LSTM	
		max (%)	mean (%)	max (%)	mean (%)
Counter languages	CNT	83.3	53.6 ± 8.6	**100.0**	86.9 ± 20.9
Regular languages	PARITY	52.1	50.6 ± 0.8	**100.0**	**100.0** ± 0.0
Star-free	DYCK-$(1,2)$	83.4	64.2 ± 8.0	**100.0**	99.3 ± 1.0
	DYCK-$(1,1)$	87.7	71.5 ± 9.3	**100.0**	88.8 ± 17.3
	LT-2	62.1	57.9 ± 2.3	**100.0**	**100.0** ± 0.0
Unambiguous polynomials	RDP-1	90.0	71.1 ± 11.5	**100.0**	**100.0** ± 0.0
	LAST	64.8	57.3 ± 2.7	**100.0**	**100.0** ± 0.0
Left-deterministic polynomials	PT-2	**100.0**	98.3 ± 3.5	**100.0**	99.7 ± 1.1
	LT-1	**100.0**	88.8 ± 11.8	**100.0**	93.0 ± 14.2
	LDP-2	**100.0**	100.0 ± 0.0	**100.0**	**100.0** ± 0.0
	LDP-1	**100.0**	97.3 ± 6.0	**100.0**	**100.0** ± 0.0
	FIRST	**100.0**	99.4 ± 1.4	**100.0**	**100.0** ± 0.0

We extract the representation $\mathbf{H}$ at each position n by concatenating the outputs of all self-attention and feedforward sublayers:

$$\mathbf{H}_{:,n} = \left[\mathbf{H}^{(0.5)}(\overline{\mathbf{w}})_{:,n}{}^{\top} \cdots \mathbf{H}^{(L)}(\overline{\mathbf{w}})_{:,n}{}^{\top}\right]^{\top}. \tag{14}$$

To assess whether the model internally tracks the states in the DFAs, we train a linear classifier to predict the state the DFA is in upon reading $\mathbf{w}_{<n}$, given $\mathbf{H}_{:,n-1}$. The classifier consists of two linear layers with no intermediate nonlinearity: the first projects to $\mathbb{F}^2$ for visualization, and the second performs classification. In Fig. 2, we plot the 2D projections and overlay the decision boundaries of the classifier.

The visualizations reveal that, for LDP-1 and LDP-2, the transformer does distinguish the states, as their representations are linearly separable. In contrast, for RDP-1, state representations are intermixed. Notably, even when we train a neural probe with nonlinearity on the representations, perfect probing accuracy remains elusive. This suggests that the transformer does not learn to fully separate the states in the non-partially ordered DFA corresponding to RDP-1.

Acknowledgments

This publication was made possible by an ETH AI Center doctoral fellowship to Jiaoda Li.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. In *NIPS Deep Learning Symposium*, 2016. URL https://arxiv.org/abs/1607.06450.

[2] Pablo Barcelo, Alexander Kozachinskiy, Anthony Widjaja Lin, and Vladimir Podolskii. Logical languages accepted by transformer encoders with hard attention. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gbrHZq07mq.

[3] Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the Ability and Limitations of Transformers to Recognize Formal Languages. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.576. URL https://aclanthology.org/2020.emnlp-main.576.

[4] J.A. Brzozowski and Faith E. Fich. Languages of $\mathcal{R}$-trivial monoids. *Journal of Computer and System Sciences*, 20(1):32–49, 1980. ISSN 0022-0000. doi: https://doi.org/10.1016/0022-0000(80)90003-3. URL https://www.sciencedirect.com/science/article/pii/0022000080900033.

[5] J. Richard Büchi. *On a Decision Method in Restricted Second Order Arithmetic*, pages 425–435. Springer New York, New York, NY, 1990. ISBN 978-1-4613-8928-6. doi: 10.1007/978-1-4613-8928-6_23. URL https://doi.org/10.1007/978-1-4613-8928-6_23.

[6] Alexandra Butoi, Ghazal Khalighinejad, Anej Svete, Josef Valvoda, Ryan Cotterell, and Brian DuSell. Training neural networks as recognizers of formal languages. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=aWLQTbfFgV.

[7] David Chiang. Transformers in uniform TC^0. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=ZA7D4nQuQF.

[8] David Chiang and Peter Cholak. Overcoming a theoretical limitation of self-attention. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7654–7664, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.527. URL https://aclanthology.org/2022.acl-long.527.

[9] David Chiang, Peter Cholak, and Anand Pillay. Tighter bounds on the expressivity of transformer encoders. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 5544–5562. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/chiang23a.html.

[10] Gregoire Deletang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A Ortega. Neural networks and the chomsky hierarchy. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WbxHAzkeQcn.

[11] Volker Diekert, Paul Gastin, and Manfred Kufleitner. A survey on small fragments of first-order logic over finite words. *International Journal of Foundations of Computer Science*, 19(03): 513–548, 2008. doi: 10.1142/S0129054108005802. URL https://doi.org/10.1142/S0129054108005802.

[12] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita

Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models. *Computing Research Repository*, arXiv:2407.21783, 2024. URL `https://arxiv.org/abs/2407.21783`. Version 2.

[13] K. Etessami and T. Wilke. An until hierarchy for temporal logic. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 108–117, 1996. doi: 10.1109/LICS.1996.561310. URL `https://ieeexplore.ieee.org/document/561310`.

[14] Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Information and Computation*, 179(2):279–295, 2002. ISSN 0890-5401. doi: https://doi.org/10.1006/inco.2001.2953. URL `https://www.sciencedirect.com/science/article/pii/S0890540101929530`.

[15] Patrick C. Fischer, Albert R. Meyer, and Arnold L. Rosenberg. Counter machines and counter languages. *Mathematical systems theory*, 2:265–283, 1968. URL https://link.springer.com/article/10.1007/BF01694011.

[16] Dov Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal analysis of fairness. POPL '80, page 163–173, New York, NY, USA, 1980. Association for Computing Machinery. ISBN 0897910117. doi: 10.1145/567446.567462. URL https://doi.org/10.1145/567446.567462.

[17] Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020. doi: 10.1162/tacl_a_00306. URL https://aclanthology.org/2020.tacl-1.11.

[18] Yiding Hao, Dana Angluin, and Robert Frank. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810, 2022. doi: 10.1162/tacl_a_00490. URL https://aclanthology.org/2022.tacl-1.46.

[19] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9: 1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://dl.acm.org/doi/10.1162/neco.1997.9.8.1735.

[21] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. In *The Seventh International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJe4ShAcF7.

[22] 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 *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=U49N5V51rU.

[23] IEEE. Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, 2008. doi: 10.1109/IEEESTD.2008.4610935.

[24] Neil Immerman and Dexter Kozen. Definability with bounded number of bound variables. *Information and Computation*, 83(2):121–139, 1989. ISSN 0890-5401. doi: https://doi.org/10.1016/0890-5401(89)90055-2. URL https://www.sciencedirect.com/science/article/pii/0890540189900552.

[25] Johan Anthony Wilem Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968. URL https://www.proquest.com/docview/302320357.

[26] S. C. Kleene. *Representation of Events in Nerve Nets and Finite Automata*, pages 3–42. Princeton University Press, Princeton, 1956. ISBN 9781400882618. doi: 10.1515/9781400882618-002. URL https://doi.org/10.1515/9781400882618-002.

[27] 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*, 2024. URL https://openreview.net/forum?id=3EWTEy9MTM.

[28] R. McNaughton and S. Papert. *Counter-free Automata*. M.I.T. Press research monographs. M.I.T. Press, 1971. ISBN 9780262130769. URL https://mitpress.mit.edu/9780262130769/counter-free-automata/.

[29] William Merrill. Sequential neural networks as automata. In Jason Eisner, Matthias Gallé, Jeffrey Heinz, Ariadna Quattoni, and Guillaume Rabusseau, editors, *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 1–13, Florence, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3901. URL https://aclanthology.org/W19-3901.

[30] William Merrill. On the linguistic capacity of real-time counter automata. *Computing Research Repository*, arXiv:2004.06866, 2020. URL https://arxiv.org/abs/2004.06866. Version 2.

[31] William Merrill and Ashish Sabharwal. A logic for expressing log-precision transformers. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 52453–52463. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a48e5877c7bf86a513950ab23b360498-Paper-Conference.pdf.

[32] William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023. doi: 10.1162/tacl_a_00562. URL https://aclanthology.org/2023.tacl-1.31.

[33] William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NjNGlPh8Wh.

[34] 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. doi: 10.1162/tacl_a_00493. URL https://aclanthology.org/2022.tacl-1.49.

[35] Franz Nowak, Anej Svete, Alexandra Butoi, and Ryan Cotterell. On the representational capacity of neural language models with chain-of-thought reasoning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12510–12548, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.676. URL https://aclanthology.org/2024.acl-long.676.

[36] OpenAI. GPT-4 technical report. *Computing Research Repository*, arXiv:2303.08774, 2023. URL https://doi.org/10.48550/arXiv.2303.08774.

[37] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57, 1977. doi: 10.1109/SFCS.1977.32. URL https://ieeexplore.ieee.org/document/4567924.

[38] Jorge Pérez, Javier Marinković, and Pablo Barceló. On the turing completeness of modern neural network architectures. In *The Seventh International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyGBdo0qFm.

[39] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074. URL https://aclanthology.org/N18-2074.

[40] Imre Simon. Piecewise testable events. In H. Brakhage, editor, *Automata Theory and Formal Languages*, pages 214–222, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg. ISBN 978-3-540-37923-2. URL https://link.springer.com/chapter/10.1007/3-540-07407-4_23.

[41] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain,

Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Johan Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew M. Dai, Andrew La, Andrew Kyle Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, Cesar Ferri, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Christopher Waites, Christian Voigt, Christopher D Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, C. Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodolà, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germàn Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Xinyue Wang, Gonzalo Jaimovitch-Lopez, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Francis Anthony Shevlin, Hinrich Schuetze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B Simon, James Koppel, James Zheng, James Zou, Jan Kocon, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh Dhole, Kevin Gimpel, Kevin Omondi, Kory Wallace Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros-Colón, Luke Metz, Lütfi Kerem Senel, Maarten Bosma, Maarten Sap, Maartje Ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramirez-Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael Andrew Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan Andrew Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter W Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan Le Bras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Russ Salakhutdinov, Ryan Andrew Chi, Seungjae Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruet-

ter, Samuel R. Bowman, Samuel Stern Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima Shammie Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven Piantadosi, Stuart Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsunori Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Venkatesh Ramasesh, vinay uday prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Sophie Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj. Featured Certification.

[42] Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 05 2024. ISSN 2307-387X. doi: 10.1162/tacl_a_00663. URL https://doi.org/10.1162/tacl_a_00663.

[43] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.127063. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

[44] Anej Svete and Ryan Cotterell. Transformers can represent n-gram language models. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6845–6881, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.381. URL https://aclanthology.org/2024.naacl-long.381/.

[45] Pascal Tesson and Denis Thérien. *Diamonds are forever: The variety DA*, pages 475–499. 2002. doi: 10.1142/9789812776884_0021. URL https://www.worldscientific.com/doi/abs/10.1142/9789812776884_0021.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[47] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf.

[48] Gail Weiss, Yoav Goldberg, and Eran Yahav. On the practical computational power of finite precision RNNs for language recognition. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745, Melbourne, Australia, July 2018. Association for

Computational Linguistics. doi: 10.18653/v1/P18-2117. URL https://aclanthology.org/P18-2117.

[49] Andy Yang and David Chiang. Counting like transformers: Compiling temporal counting logic into softmax transformers. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=FmhPg4UJ9K.

[50] Andy Yang, David Chiang, and Dana Angluin. Masked hard-attention transformers recognize exactly the star-free languages. *Computing Research Repository*, arXiv:2310.13897, 2024. URL https://arxiv.org/abs/2310.13897. Version 3.

[51] Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. Self-attention networks can process bounded hierarchical languages. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3770–3785, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.292. URL https://aclanthology.org/2021.acl-long.292.

A Background

In this section, we provide background information that was omitted from the main text for brevity. We include it here for the sake of completeness.

A.1 Temporal logic

Temporal logic [37] is a special case of modal logic, with modalities referring to time.

A.1.1 LTL[P, F, S, U]

Linear temporal logic **LTL**[**P**, **F**, **S**, **U**] includes four temporal operators: **P** (`past`), **F** (`future`), **S** (`since`), and **U** (`until`). Notably, **P**, **F**, and one of **S** or **U** can be omitted without loss of expressive power [16]. We write $\top$ to denote TRUE and $\bot$ to denote FALSE. The semantics of the formulas are defined inductively as follows:

- $\mathbf{w}, n \models \pi_{\mathtt{a}}$ iff the n^{th} symbol of $\mathbf{w}$ is an $\mathtt{a}$.
- $\mathbf{w}, n \models \psi_1 \wedge \psi_2$ iff $\mathbf{w}, n \models \psi_1$ and $\mathbf{w}, n \models \psi_2$.
- $\mathbf{w}, n \models \neg\psi$ iff $\mathbf{w}, n \not\models \psi$.
- $\mathbf{w}, n \models \mathbf{P}\psi$ iff there exists m such that $m < n$ and $\mathbf{w}, m \models \psi$.
- $\mathbf{w}, n \models \mathbf{F}\psi$ iff there exists m such that $m > n$ and $\mathbf{w}, m \models \psi$.
- $\mathbf{w}, n \models \psi_1 \mathbf{S} \psi_2$ iff there exists m such that $m < n$, $\mathbf{w}, m \models \psi_2$, and for every k such that $m < k < n$, $\mathbf{w}, k \models \psi_1$.
- $\mathbf{w}, n \models \psi_1 \mathbf{U} \psi_2$ iff there exists m such that $m > n$, $\mathbf{w}, m \models \psi_2$, and for every k such that $n < k < m$, $\mathbf{w}, k \models \psi_1$.

Example:

- $\mathtt{abb}, 1 \models \pi_{\mathtt{a}}$ as the first symbol is $\mathtt{a}$.
- $\mathtt{abb}, 3 \models \mathbf{P}\pi_{\mathtt{a}}$ as an $\mathtt{a}$ occurs before position 3.
- $\mathtt{abb}, 3 \models \pi_{\mathtt{b}} \mathbf{S} \pi_{\mathtt{a}}$ as $\pi_{\mathtt{a}}$ holds at position 1 (before position 3), and $\pi_{\mathtt{b}}$ holds at every position between 1 and 3.

To say a string $\mathbf{w}$ of length N satisfies a formula ψ, written as $\mathbf{w} \models \psi$, we imagine we start at a position outside the string, e.g., position 0 or $N + 1$.

Example:

- $\mathtt{abb} \models \pi_{\mathtt{b}} \mathbf{S} \pi_{\mathtt{a}}$ as the formula is satisfied at position 4, outside the string.
- $\mathtt{abb} \models \pi_{\mathtt{a}} \mathbf{U} \pi_{\mathtt{b}}$ as the formula is satisfied at position 0, outside the string.

The language defined by a formula ψ is the set of all strings that satisfy ψ, denoted by $\mathbb{L}(\psi)$:

$$\mathbb{L}(\psi) \stackrel{\text{def}}{=} \{\mathtt{w} \in \Sigma^* \mid \mathtt{w} \models \psi\}. \tag{15}$$

Example:

$$\mathbb{L}\left(\pi_{\mathtt{b}} \mathbf{S} \pi_{\mathtt{a}}\right) = \Sigma^* \mathtt{a} \mathtt{b}^* \tag{16a}$$
$$\mathbb{L}\left(\pi_{\mathtt{a}} \mathbf{U} \pi_{\mathtt{b}}\right) = \mathtt{a}^* \mathtt{b} \Sigma^* \tag{16b}$$

We define the **operator depth** of a formula ψ, denoted $\mathrm{od}(\psi)$, as the maximum number of nested temporal operators in ψ. It is defined recursively as follows:

$$\begin{aligned}
\mathrm{od}(\pi_{\mathtt{a}}) &= 0 \\
\mathrm{od}(\psi_1 \wedge \psi_2) &= \max(\mathrm{od}(\psi_1), \mathrm{od}(\psi_2)) \\
\mathrm{od}(\neg\psi) &= \mathrm{od}(\psi) \\
\mathrm{od}(\mathbf{P}\psi) &= \mathrm{od}(\psi) + 1 \\
\mathrm{od}(\mathbf{F}\psi) &= \mathrm{od}(\psi) + 1 \\
\mathrm{od}(\psi_1 \mathbf{S} \psi_2) &= \max(\mathrm{od}(\psi_1), \mathrm{od}(\psi_2)) + 1 \\
\mathrm{od}(\psi_1 \mathbf{U} \psi_2) &= \max(\mathrm{od}(\psi_1), \mathrm{od}(\psi_2)) + 1
\end{aligned}$$

A.1.2 LTL[P, F]

Unary temporal logic **LTL**[**P**, **F**] is the fragment of **LTL**[**P**, **F**, **S**, **U**] that excludes the binary operators **S** and **U**. This restriction renders **LTL**[**P**, **F**] strictly less expressive than **LTL**[**P**, **F**, **S**, **U**] [13, 14].

Example:
Consider the language consisting of all words that begin with $\mathtt{a}$ and end with $\mathtt{b}$. To enforce that a word starts with $\mathtt{a}$, we require that $\mathtt{a}$ occurs at the beginning of the word, i.e., it is preceded by nothing. This can be expressed as $\mathbf{P}(\pi_{\mathtt{a}} \wedge \neg \mathbf{P} \top)$. Symmetrically, ending with $\mathtt{b}$ can be expressed using the dual condition involving **F**. Thus, the formula

$$\mathbb{L}(\mathbf{P}(\pi_{\mathtt{a}} \wedge \neg \mathbf{P} \top) \wedge \mathbf{P}(\pi_{\mathtt{b}} \wedge \neg \mathbf{F} \top)) = \mathtt{a} \Sigma^* \mathtt{b} \tag{17}$$

defines the desired language.

A.1.3 LTL[P]

We define **LTL**[**P**] as the past fragment of **LTL**[**P**, **F**], which excludes the **F** operator.

Proposition A.1. **LTL**[**P**] *is strictly less expressive than* **LTL**[**P**, **F**].

Proof. Since **LTL**[**P**] is a syntactic fragment of **LTL**[**P**, **F**], every formula expressible in **LTL**[**P**] is also expressible in **LTL**[**P**, **F**]. To establish strictness, it suffices to exhibit a language definable in **LTL**[**P**, **F**] that is not definable in **LTL**[**P**]. Consider again the language $\mathtt{a}\Sigma^*\mathtt{b}$. It is not definable in **LTL**[**P**], as expressing the end of a string requires the ability to look forward, which **LTL**[**P**] lacks. ■

A.2 First-order logic

The seminal work of Büchi [5] demonstrated how properties of languages can be described using logical formulas, while McNaughton and Papert [28] was the first to restrict these formulas to first-order logic (**FO**[$<$]).

A.2.1 FO[<]

The atomic formulas of **FO**[<] consist of unary predicates $\pi_{\mathtt{a}}$ for each $\mathtt{a} \in \Sigma$ and a binary numerical predicate $<$ that can be used to determine the sequential order between variables. Formulas in **FO**[<] may use arbitrarily many variables. However, it has been shown that three variables are both necessary and sufficient to define all first-order definable languages [25, 24]. In addition to the usual Boolean connectives $\wedge$ and $\neg$, there is an **existential quantifier** $\exists$. The formulas in **FO**[<] are constructed inductively as follows:

- Every atomic formula is a formula;
- A Boolean combination of formulas is a formula;
- If $\phi(x, \ldots)$ is a formula, so is $\exists x : \phi(x, \ldots)$;

The **universal quantifier** $\forall$ can be defined as follows:

$$\forall x : \phi(x, \ldots) \stackrel{\text{def}}{=} \neg \exists x : \neg \phi(x, \ldots). \tag{18}$$

A formula with no free variables is called a **sentence**. We write $\mathtt{w} \models \phi$ if ϕ is satisfied for $\mathtt{w}$ under the interpretation, where variables range over the positions in $\mathtt{w}$, unary predicates $\pi_{\mathtt{a}}(x)$ hold if the symbol at position x is $\mathtt{a}$, and $<$ is interpreted as the natural order on positions. A sentence ϕ can define a language, the set of strings over Σ satisfying it, denoted as $\mathbb{L}(\phi)$.

Example:
The formula below defines the language $\mathtt{a}\Sigma^*\mathtt{b}$:

$$\exists x \forall y \exists z : (\pi_{\mathtt{a}}(x) \wedge x \leq y \wedge y \leq z \wedge \pi_{\mathtt{b}}(z)) \tag{19}$$

where $x \leq y$ is shorthand for $x < y \vee x = y$.

The **quantifier depth** of a formula ϕ, denoted $\mathrm{qd}(\phi)$, is defined recursively as follows:

$$\begin{aligned} \mathrm{qd}(\pi_{\mathtt{a}}) &= 0 \\ \mathrm{qd}(\phi_1 \wedge \phi_2) &= \max(\mathrm{qd}(\phi_1), \mathrm{qd}(\phi_2)) \\ \mathrm{qd}(\neg\phi) &= \mathrm{qd}(\phi) \\ \mathrm{qd}(\exists x : \phi(x, \ldots)) &= \mathrm{qd}(\phi) + 1 \end{aligned}$$

It is well known that **FO**[<] and **LTL**[**P**, **F**, **S**, **U**] are equivalent in expressive power [25] and they both define exactly the class of star-free languages [28].

A.2.2 FO^2[<]

FO^2[<] is the fragment of **FO**[<] restricted to using only two distinct variables (typically reused via quantification).

Example:
The formula Eq. (19) can be written as a formula in **FO**^{2}[<] as follows:

$$\exists x : (\pi_{\mathtt{a}}(x) \wedge \forall y : (x \leq y \wedge \exists x : (y \leq x \wedge \pi_{\mathtt{b}}(x)))), \tag{20}$$

which uses only two distinct variables x and y.

It is shown that **LTL**[**P**, **F**] and **FO**^{2}[<] recognize precisely the same languages [14].

A.2.3 PFO^2[<]

We define **PFO**^{2}[<] as the past fragment of **FO**^{2}[<], in which formulas are restricted so that whenever a free variable x is present, every existential quantifier must be of the form $\exists y < x$.. This ensures that all quantification is limited to positions at or before x, i.e., only past positions relative to the free variable are accessed.

$\mathbf{PFO}^2[<]$ has precisely the same expressive power as $\mathbf{LTL}[\mathbf{P}]$. We establish this equivalence by providing translations in both directions.

Lemma A.2. *Every formula ψ in $\mathbf{LTL}[\mathbf{P}]$ can be translated into an equivalent single-variable formula $\phi(x)$ in $\mathbf{PFO}^2[<]$.*

Proof. We proceed by structural induction on the formula ψ.

Base case. If $\psi = \pi_{\mathtt{a}}$, we translate it to $\phi(x) = \pi_{\mathtt{a}}(x)$.

Induction step. Assume ψ_1 and ψ_2 can be translated into $\phi_1(x)$ and $\phi_2(x)$ respectively. We translate compound formulas as follows:

- If $\psi = \psi_1 \wedge \psi_2$, then $\phi(x) = \phi_1(x) \wedge \phi_2(x)$.
- If $\psi = \neg\psi_1$, then $\phi(x) = \neg\phi_1(x)$.
- If $\psi = \mathbf{P}\psi_1$, then $\phi(x) = \exists y < x : \phi_1(y)$

In each case, the resulting formula $\phi(x)$ belongs to $\mathbf{PFO}^2[<]$, since it uses at most two variables (the free variable x and a bound variable y), both of which can be reused. ■

Lemma A.3. *Every single-variable formula $\phi(x)$ in $\mathbf{PFO}^2[<]$ can be translated into an equivalent formula ψ in $\mathbf{LTL}[\mathbf{P}]$.*

Proof. We proceed by induction on the quantifier depth $\mathrm{qd}(\phi)$.

Base case. If $\mathrm{qd}(\phi) = 0$, then $\phi(x)$ is a Boolean combination of atomic formulas such as $\pi_{\mathtt{a}}(x)$, which are directly translatable into $\mathbf{LTL}[\mathbf{P}]$ formulas.

Induction step. Assume that every formula of quantifier depth at most k can be translated into an equivalent $\mathbf{LTL}[\mathbf{P}]$ formula. Consider a formula $\phi(x)$ of depth $k+1$. The formula can be written as a Boolean combination of:

- Subformulas of quantifier depth at most k, which can be translated into $\mathbf{LTL}[\mathbf{P}]$ by the inductive hypothesis;
- Subformulas of the form $\exists y < x : \phi_1(y)$, where ϕ_1 has depth at most k. By the inductive hypothesis, $\phi_1(y)$ translates to a $\mathbf{LTL}[\mathbf{P}]$ formula ψ_1, and the whole subformula translates to $\psi = \mathbf{P}\psi_1$.

■

Combining the two lemmas, we obtain the following theorem.

Theorem 2.1. $\mathbf{PFO}^2[<]$ *and* $\mathbf{LTL}[\mathbf{P}]$ *have the same expressive power.*

Proof. The result follows directly from Lemma A.2 and Lemma A.3. To define languages, $\mathbf{LTL}[\mathbf{P}]$ formulas are interpreted at position $N+1$. Similarly, $\mathbf{PFO}^2[<]$ formulas with a single variable x can be converted into sentences by prefixing them with $\exists x$, which is semantically equivalent to $\exists x < N+1$. ■

B Transformers

In this section, we describe the transformer idealization under consideration in more detail. The formal proofs for the following two results are also given: (i) every transformer can be translated into an equivalent $\mathbf{PFO}^2[<]$ formula, and (ii) every $\mathbf{LTL}[\mathbf{P}]$ formula can be simulated by a transformer.

B.1 Transformer architecture

The assumptions we make are restated as follows:

Assumption 3.1. *We assume that the transformer satisfies the following conditions:*

*1. **Fixed precision**: There exists a finite set of values that the floating-point numbers can assume. We denote this set as $\mathbb{F} = \{f_1, f_2, \ldots\}$.*

*2. **No positional encoding** (NoPE): Positional encodings are omitted for now, but are addressed in §F.3.*

*3. **Soft attention**: Attention weights are computed using the* $\mathrm{softmax}$ *function. We also treat average hard attention (AHA) in §F.1, showing that AHA is equally expressive as soft attention.*

*4. **Strict future masking**: As in Yang et al. [50], the attention is strictly future-masked. Strict masking is shown to be more expressive than non-strict masking in §F.2.*

*5. **Recognition**: Following standard practice in expressivity studies [42], the transformer is treated as a language recognizer. We will extend the analysis to transformer language models in §5.*

We denote the set of non-negative floating-point numbers as $\mathbb{F}_{\geq 0} \subset \mathbb{F}$. Similar notation is used for positive numbers, negative numbers, and other subsets. The set $\overline{\mathbb{F}}$ includes two special values, ∞ and $-\infty$, representing positive and negative infinity. Their behaviors are defined as follows:

- $\forall f \in \mathbb{F} \setminus \{-\infty\}, \infty + f \stackrel{\text{def}}{=} \infty$;
- $\forall f \in \mathbb{F} \setminus \{-\infty\}, -\infty + f \stackrel{\text{def}}{=} -\infty$;
- $\forall f \in \mathbb{F}_{>0}, f \cdot \infty \stackrel{\text{def}}{=} \infty$ and $f \cdot (-\infty) \stackrel{\text{def}}{=} -\infty$;
- $\forall f \in \mathbb{F}_{<0}, f \cdot \infty \stackrel{\text{def}}{=} -\infty$ and $f \cdot (-\infty) \stackrel{\text{def}}{=} \infty$;
- $\forall f \in \mathbb{F} \setminus \{\infty, -\infty\}, f/\infty \stackrel{\text{def}}{=} 0$;
- $\exp(\infty) \stackrel{\text{def}}{=} \infty$ and $\exp(-\infty) \stackrel{\text{def}}{=} 0$.

Transformers take strings as inputs. Given a string $\mathtt{w} = \mathtt{w}_1 \cdots \mathtt{w}_N$ over an alphabet Σ, we append a special end-of-sentence symbol $\textsc{eos} \notin \Sigma$ to form the extended string $\overline{\mathtt{w}} \stackrel{\text{def}}{=} \mathtt{w}_1 \cdots \mathtt{w}_N \textsc{eos}$ over the extended alphabet $\overline{\Sigma} \stackrel{\text{def}}{=} \Sigma \cup \{\textsc{eos}\}$.

The transformer consists of an input layer, followed by a stack of L hidden layers and a final output layer.

B.1.1 Input layer

An embedding function $\mathbf{e}$ of type $\overline{\Sigma} \to \mathbb{F}^D$ maps each symbol to a D-dimensional column vector. The embedding layer applies $\mathbf{e}$ to each symbol in $\overline{\mathtt{w}}$ to produce the input representation: $\mathbf{E}(\overline{\mathtt{w}}) \in \mathbb{F}^{D \times (N+1)}$, i.e.,

$$\mathbf{E}(\overline{\mathtt{w}})_{:,n} \stackrel{\text{def}}{=} \mathbf{e}(\mathtt{w}_n), \quad n \in \{1, \ldots, N+1\}. \tag{21}$$

where $\mathtt{w}_{N+1} \stackrel{\text{def}}{=} \textsc{eos}$.

B.1.2 Hidden layers

Each hidden layer contains two sublayers: a self-attention mechanism and a feedforward network.

The self-attention mechanism is a function of type $\mathbb{F}^{D \times (N+1)} \to \mathbb{F}^{D \times (N+1)}$. Given input $\mathbf{H}(\overline{\mathtt{w}}) \in \mathbb{F}^{D \times (N+1)}$, we compute:

$$\mathbf{Q} \stackrel{\text{def}}{=} \mathbf{\Theta}^{\mathsf{Q}} \mathbf{H}(\overline{\mathtt{w}}), \tag{22a}$$

$$\mathbf{K} \stackrel{\text{def}}{=} \mathbf{\Theta}^{\mathsf{K}} \mathbf{H}(\overline{\mathtt{w}}), \tag{22b}$$

$$\mathbf{V} \stackrel{\text{def}}{=} \mathbf{\Theta}^{\mathsf{V}} \mathbf{H}(\overline{\mathtt{w}}), \tag{22c}$$

where $\mathbf{\Theta}^{\mathsf{Q}}, \mathbf{\Theta}^{\mathsf{K}}, \mathbf{\Theta}^{\mathsf{V}} \in \mathbb{F}^{D \times D}$ are learnable parameter matrices.

A pairwise compatibility score $\mathbf{S} \in \mathbb{F}^{(N+1) \times (N+1)}$ is computed via scaled dot product:

$$\mathbf{S}_{n,m} \stackrel{\text{def}}{=} \frac{\mathbf{Q}_{:,n} \cdot \mathbf{K}_{:,m}}{\sqrt{D}}. \tag{23}$$

We then compute attention weights $\boldsymbol{\alpha} \in \mathbb{F}^{(N+1)\times(N+1)}$ using $\mathrm{softmax}$:

$$\boldsymbol{\alpha}_{n,m} \stackrel{\text{def}}{=} \frac{\exp(\mathbf{S}_{n,m})}{\sum_{i<n} \exp(\mathbf{S}_{n,i})}. \tag{24}$$

In practice, the $\mathrm{softmax}$ is numerically stabilized by subtracting the maximum score from all scores before exponentiation:

$$\boldsymbol{\alpha}_{n,m} \stackrel{\text{def}}{=} \frac{\exp(\mathbf{S}_{n,m} - \max_{i<n} \mathbf{S}_{n,i})}{\sum_{j<n} \exp(\mathbf{S}_{n,j} - \max_{i<n} \mathbf{S}_{n,i})}. \tag{25}$$

The attention output at position n is then computed as a weighted sum of the value vectors:

$$\mathbf{A}(\mathbf{H}(\overline{\mathbf{w}}))_{:,n} \stackrel{\text{def}}{=} \sum_{m<n} \boldsymbol{\alpha}_{n,m} \mathbf{V}_{:,m}. \tag{26}$$

We assume a single attention head, since multiple heads do not increase expressive power [27].

The feedforward sublayer is a function of type $\mathbb{F}^{D\times(N+1)} \to \mathbb{F}^{D\times(N+1)}$ and consists of two linear transformations with a ReLU nonlinearity:

$$\mathbf{F}(\mathbf{H}(\overline{\mathbf{w}}))_{:,n} \stackrel{\text{def}}{=} \boldsymbol{\Theta}^{\text{F2}} \mathrm{ReLU}\left(\boldsymbol{\Theta}^{\text{F1}} \mathbf{H}(\overline{\mathbf{w}})_{:,n} + \mathbf{b}^{\text{F1}}\right) + \mathbf{b}^{\text{F2}}, \tag{27}$$

where $\boldsymbol{\Theta}^{\text{F1}} \in \mathbb{F}^{D'\times D}, \mathbf{b}^{\text{F1}} \in \mathbb{F}^{D'}, \boldsymbol{\Theta}^{\text{F2}} \in \mathbb{F}^{D'\times D}, \mathbf{b}^{\text{F2}} \in \mathbb{F}^{D}$ are trainable parameters, D' is the hidden size of the intermediate layer.

B.1.3 Output layer

The final output layer is a function of type $\mathbb{F}^{D\times(N+1)} \to \mathbb{F}$ that computes a scalar score based on the representation at the final position (corresponding to EOS):

$$\mathrm{C}\left(\mathbf{H}^{(L)}(\overline{\mathbf{w}})\right) = \left(\boldsymbol{\theta}^{\text{C}}\right)^{\top} \mathbf{H}^{(L)}(\overline{\mathbf{w}})_{:,N+1} + b^{\text{C}} \tag{28}$$

where $\boldsymbol{\theta}^{\text{C}} \in \mathbb{F}^{D}$ and $b^{\text{C}} \in \mathbb{F}$ are learnable parameters.

B.2 From transformers to $\mathbf{PFO}^2[<]$

In this sub-section, we show that any transformer can be simulated by $\mathbf{PFO}^2[<]$. We begin by formalizing what it means for a function to be simulated by $\mathbf{PFO}^2[<]$. With a slight abuse of notation, we write D to refer either to the model's hidden dimensionality or to 1, in the case of a scalar classification output.

Definition B.1. *A function $\mathbf{H}\colon \overline{\Sigma}^{N+1} \to \mathbb{F}^{D\times(N+1)}$ is said to be simulated by $\mathbf{PFO}^2[<]$ if for every dimension $d \in \{1,\ldots,D\}$ and every floating-point value $f \in \mathbb{F}$, there exists a single-variable formula $\phi(x)$ in $\mathbf{PFO}^2[<]$ such that for every position $n \in \{1,\ldots,N+1\}$,*

$$\mathbf{H}(\overline{\mathbf{w}})_{d,n} = f \quad \text{if and only if} \quad \overline{\mathbf{w}}, n \models \phi(x). \tag{29}$$

Similarly, a function $\mathbf{H}$ of type $\overline{\Sigma}^{N+1} \to \mathbb{F}^{(N+1)\times(N+1)}$ is said to be simulated by $\mathbf{PFO}^2[<]$ if for every floating-point value $f \in \mathbb{F}$, there exists a two-variable formula $\phi(x,y)$ in $\mathbf{PFO}^2[<]$ such that for every pair of positions $n, m \in \{1,\ldots,N+1\}$,

$$\mathbf{H}(\overline{\mathbf{w}})_{n,m} = f \quad \text{if and only if} \quad \overline{\mathbf{w}}, n, m \models \phi(x,y). \tag{30}$$

We will prove the following proposition, which we will use repeatedly.

Proposition B.2. *Let $\mathbf{H}_1, \mathbf{H}_2$ be functions of type $\overline{\Sigma}^{N+1} \to \mathbb{F}^{D}$, and let $\mathbf{H}$ be either a function $\mathbb{F}^{D} \to \mathbb{F}^{D'}$ (where D' is not necessarily equal to D) or $\mathbb{F}^{D} \times \mathbb{F}^{D} \to \mathbb{F}$. If both $\mathbf{H}_1$ and $\mathbf{H}_2$ can be simulated by $\mathbf{PFO}^2[<]$, then so can $\mathbf{H}(\mathbf{H}_1)$ and $\mathbf{H}(\mathbf{H}_1, \mathbf{H}_2)$.*

Proof. Since the inputs are fixed-dimensional vectors over $\mathbb{F}$, the number of possible value combinations is finite. For any given output value $f \in \mathbb{F}$, one can enumerate all input configurations for which the function yields f, and construct a $\mathbf{PFO}^2[<]$ formula expressing their disjunction. ■

This proposition accounts for all components of the transformer architecture, such as the feedforward sublayers, projection operations, dot products, elementwise operations such as addition, scalar operations such as division, the $\exp$ function, and the classification head—except for the embedding layer, the summations involved in the attention computation, and the operation of identifying the maximum score introduced by the stabilized softmax.

We handle the embedding layer as follows:

Lemma B.3. *The embedding layer can be simulated by* $\mathbf{PFO}^2[<]$.

Proof. For any dimension d and position n, the embedding layer outputs a fixed value f if the symbol at position n is mapped to f by the embedding function $\mathbf{e}$ in the d-th coordinate. Since $\overline{\Sigma}$ is finite and $\mathbf{e}$ is fixed, we can express this using a disjunction over all symbols that are mapped to f in coordinate d:

$$\mathbf{E}(\overline{\mathbf{w}})_{d,n} = f \quad \text{if and only if} \quad \overline{\mathbf{w}}, n \models \bigvee_{\mathtt{a} \in \{\mathtt{a} \mid \mathbf{e}(\mathtt{a})_d = f\}} \pi_{\mathtt{a}}(x). \tag{31}$$

■

We next address the summations in the attention mechanism. We begin by establishing the following result:

Proposition B.4. $\mathbf{PFO}^2[<]$ *can count up to a threshold.*

Proof. The threshold counting quantifiers can be defined as follows.

- The quantifier for "there exists at least one" coincides with the standard bounded existential quantifier:
$$\exists^{\geq 1} y < x : \phi(y) \stackrel{\text{def}}{=} \exists y < x : \phi(y). \tag{32}$$
- "There exist at least two" can be defined as:
$$\exists^{\geq 2} y < x : \phi(y) \stackrel{\text{def}}{=} \exists y < x : (\phi(y) \wedge \exists x < y : \phi(x)). \tag{33}$$
- We can define "exactly one" by combining the above:
$$\exists^{=1} y < x : \phi(y) \stackrel{\text{def}}{=} \exists^{\geq 1} y < x : \phi(y) \wedge \neg \exists^{\geq 2} y < x : \phi(y). \tag{34}$$
- Additional counting quantifiers up to a threshold can be defined in a similar fashion.

■

Soft attention involves two types of summations, which we address in turn. The first appears in the denominator of the $\mathrm{softmax}$ function, used to compute the attention weights $\boldsymbol{\alpha}$ (Eq. (24)). Since $\exp$ outputs non-negative values, this summation consists entirely of non-negative numbers and can be simulated by $\mathbf{PFO}^2[<]$.

Lemma B.5. $\mathbf{PFO}^2[<]$ *can simulate a sum of non-negative fixed-precision floating-point numbers.*

Proof. Since $\mathbb{F}_{\geq 0}$ is finite, there exists a finite set of possible sums. Moreover, because all values are non-negative, for each such sum there are only finitely many combinations of input values that yield it. Each combination can be characterized using the threshold-counting quantifiers introduced in Proposition B.4: Too many positive entries lead to overflow, while 0's do not affect the sum. Taking a finite disjunction over all such combinations defines a $\mathbf{PFO}^2[<]$ formula that holds exactly when the prefix sum equals a given value. ■

The second summation occurs in the computation of attention outputs (Eq. (26)). The value matrix $\mathbf{V} \in \mathbb{F}^{D \times (N+1)}$ may include both positive and negative values, so we cannot apply the previous approach directly. However, as pointed out by Merrill and Sabharwal [31]:

Proposition B.6. *Under fixed precision, there exists an upper limit on the number of non-zero entries in the output of a* $\mathrm{softmax}$ *function. We refer to the upper bound as the* maximum attention span $N_{\max}$.

Proof. This upper bound is given by:

$$N_{\max} = \left\lfloor \frac{\min(1, \max(\mathbb{F} \setminus \{\infty\}))}{\min(\mathbb{F}_{>0})} \right\rfloor, \tag{35}$$

where $\lfloor \cdot \rfloor$ is the floor function. ■

Consequently, the number of nonzero attention weights is bounded. It follows that Eq. (26) computes the sum over a bounded number of terms, which can be simulated by $\mathbf{PFO}^2[<]$:

Lemma B.7. $\mathbf{PFO}^2[<]$ *can simulate a sum of a bounded number of fixed-precision floating-point numbers.*

Proof. Since the number of summands is bounded, we can enumerate all possible combinations of inputs. This behavior can be simulated using the threshold counting quantifiers introduced in Proposition B.4. ■

Finally, we account for the numerical-stability adjustment in Eq. (25). The only additional operation introduced by this stabilization is the identification of the maximum score among a finite set of floating-point values, which can be expressed in $\mathbf{PFO}^2[<]$ as follows:

Proposition B.8. $\mathbf{PFO}^2[<]$ *can identify the maximum score* $\max_{i<n} \mathbf{S}_{n,i}$.

Proof. Let the ordered set of floating-point values be $\mathbb{F} = \{-\infty, f_1, f_2, \ldots, f_K, \infty\}$. For each $f \in \mathbb{F}$, let $\phi_f(x)$ be a $\mathbf{PFO}^2[<]$-formula that holds iff the value at position x equals f. Since the set of floating-point numbers is finite, we can enumerate all possible maximum values. For each candidate maximum value f, we can construct a $\mathbf{PFO}^2[<]$ formula $\phi_f^{\max}(x)$ that holds if and only if the scores among all positions $y < x$ are less than or equal to f, and at least one value equals f. They can be defined inductively over the total order of $\mathbb{F}$ as follows:

$$\phi_{\infty}^{\max}(x) \stackrel{\text{def}}{=} \exists y < x : \phi_{\infty}(y), \tag{36a}$$

$$\phi_{f_K}^{\max}(x) \stackrel{\text{def}}{=} (\neg \exists y < x : \phi_{\infty}^{\max}(y)) \wedge \exists y < x : \phi_{f_K}(y), \tag{36b}$$

$$\vdots$$

$$\phi_{f_1}^{\max}(x) \stackrel{\text{def}}{=} \left(\neg \exists y < x : \bigvee_{f \in \{\infty, f_K, \ldots, f_2\}} \phi_f^{\max}(y) \right) \wedge \exists y < x : \phi_{f_1}(y), \tag{36c}$$

$$\phi_{-\infty}^{\max}(x) \stackrel{\text{def}}{=} \left(\neg \exists y < x : \bigvee_{f \in \{\infty, f_K, \ldots, f_1\}} \phi_f^{\max}(y) \right) \wedge \exists y < x : \phi_{-\infty}(y). \tag{36d}$$

■

We have now completed the proof of the following theorem:

Theorem 3.2. *Every transformer can be simulated by* $\mathbf{PFO}^2[<]$.

Proof. By Lemma B.3, the embedding layer can be simulated by $\mathbf{PFO}^2[<]$. By Proposition B.2, simulation by $\mathbf{PFO}^2[<]$ is closed under all subsequent operations in the transformer, except for summations. By Lemma B.5, Proposition B.6, and Lemma B.7, the summation operations involved in attention computation can also be simulated by $\mathbf{PFO}^2[<]$. Therefore, the entire transformer can be simulated by $\mathbf{PFO}^2[<]$. ■

B.3 From LTL[P] to transformers

In this subsection, we show that every $\mathbf{LTL}[\mathbf{P}]$ formula can be simulated by a transformer. The central idea is to encode the truth value of the formula at each position as a dedicated coordinate within the transformer's hidden state: the value 1 represents $\top$, and 0 represents $\bot$.

The formal definition is as follows.

Definition B.9. *A $\mathbf{LTL}[\mathbf{P}]$ formula ψ is said to be simulated by a function $\mathbf{H}$ of type $\overline{\Sigma}^{N+1} \to \mathbb{F}^{D\times(N+1)}$ if there exists a coordinate $d \in \{1,\dots,D\}$ such that for every position $n \in \{1,\dots,N{+}1\}$ and every input string $\overline{\boldsymbol{w}} \in \overline{\Sigma}^*$,*

$$\mathbf{H}(\overline{\boldsymbol{w}})_{d,n} = \begin{cases} 1, & \text{if } \overline{\boldsymbol{w}}, n \models \psi, \\ 0, & \text{otherwise.} \end{cases} \tag{37}$$

We introduce a one-hot encoding function $[\![]\!]$, which maps an index to a column vector of zeros except for a single entry equal to 1 at that index. Let $\mathbf{0}$ denote the all-zero column vector.

We first show that the atomic formulas can be translated into an embedding layer.

Lemma B.10. *The atomic formulas π_a for $a \in \overline{\Sigma}$ can be simulated by an embedding layer.*

Proof. We assign a distinct coordinate d to every distinct symbol $a \in \overline{\Sigma}$ with a map $g\colon \overline{\Sigma} \to \{1,\dots,|\overline{\Sigma}|\}$. Since the biases in attention mechanisms are sometimes omitted, we include a dedicated bias coordinate d_b that always holds a constant value 1. The embedding layer is then defined as follows:

$$\mathbf{H}^{(0)}(\overline{\mathtt{w}})_{d,n} = \begin{cases} 1, & \text{if } d = d_b, \\ 1, & \text{if } d = g(\mathtt{w}_n), \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

Coordinates that remain inactive at this stage are initialized to 0; they will later be used to store logical formulas introduced in subsequent layers. ∎

Next, we show that the Boolean connectives can be simulated.

Lemma B.11. *If ψ_1 and ψ_2 can be simulated by a transformer, then their Boolean combinations can also be simulated by a transformer.*

Proof. Assume ψ_1 and ψ_2 are simulated by $\mathbf{H}^{(\ell)}_{d_1,:}$ and $\mathbf{H}^{(\ell)}_{d_2,:}$ respectively. For brevity, we write $\mathbf{x} = \mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_1,:}$ and $\mathbf{y} = \mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_2,:}$, so $\mathbf{x}, \mathbf{y} \in \mathbb{F}^{1\times(N+1)}$ are row vectors indexed by positions. All operations below are applied elementwise.

- $\neg\psi_1$: Construct a layer that computes, in a fresh coordinate $d_\neg$,

$$-\mathbf{x} + 1. \tag{39}$$

  This can be implemented with an FFN as follows. Select a hidden unit d' in the intermediate hidden state $\{1,\dots,D'\}$ and set the first linear transformation to:

$$\boldsymbol{\Theta}^{\mathrm{F1}}{}_{d,:} = \begin{cases} -[\![d_1]\!]^\top & \text{if } d = d', \\ \mathbf{0}^\top & \text{otherwise,} \end{cases} \qquad \mathbf{b}^{\mathrm{F1}} = [\![d']\!]. \tag{40}$$

  The second linear transformation maps d' to the output coordinate $d_\neg$:

$$\boldsymbol{\Theta}^{\mathrm{F2}}{}_{d,:} = \begin{cases} [\![d']\!]^\top & \text{if } d = d_\neg, \\ \mathbf{0}^\top & \text{otherwise,} \end{cases} \qquad \mathbf{b}^{\mathrm{F2}} = \mathbf{0}. \tag{41}$$

  The attention sublayer is bypassed by setting $\boldsymbol{\Theta}^{\mathrm{Q}}, \boldsymbol{\Theta}^{\mathrm{K}}, \boldsymbol{\Theta}^{\mathrm{V}}$ to zero matrices.

- $\psi = \psi_1 \wedge \psi_2$: Compute, in a fresh coordinate $d_\wedge$,

$$\mathrm{ReLU}\left(\mathbf{x} + \mathbf{y} - 1\right). \tag{42}$$

Implement this with a single FFN as follows. Select a hidden unit d' and set the first linear transformation to:

$$\mathbf{\Theta}^{\mathrm{F1}}{}_{d,:} = \begin{cases} [\![d_1]\!]^\top + [\![d_2]\!]^\top & \text{if } d = d', \\ \mathbf{0}^\top & \text{otherwise,} \end{cases} \qquad \mathbf{b}^{\mathrm{F1}} = -[\![d']\!]. \tag{43}$$

The second linear transformation connects d' to the output coordinate $d_\wedge$:

$$\mathbf{\Theta}^{\mathrm{F2}}{}_{d,:} = \begin{cases} [\![d']\!]^\top & \text{if } d = d_\wedge, \\ \mathbf{0}^\top & \text{otherwise,} \end{cases} \qquad \mathbf{b}^{\mathrm{F2}} = \mathbf{0}. \tag{44}$$

Again, the attention sublayer is bypassed.

■

Finally, we address the temporal operator $\mathbf{P}$.

Lemma B.12. *If ψ can be simulated by a transformer, then $\mathbf{P}\psi$ can also be simulated by a transformer.*

Proof. We start by introducing the base construction and then discuss two issues arising from fixed-precision arithmetic, together with their corresponding fixes. Assume ψ is simulated by $\mathbf{H}^{(\ell)}_{d_1,:}$.

Base construction. We construct an attention mechanism such that each position n attends uniformly to all previous positions $m < n$ where $\mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_1,m} = 1$. Specifically, we need the attention scores to be:

$$\mathbf{S}_{n,m} = \begin{cases} 0 & \text{if } \mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_1,m} = 1, \\ -f_{\text{large}} & \text{if } \mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_1,m} = 0, \end{cases} \tag{45}$$

where $f_{\text{large}} \in \mathbb{F}$ is a large enough positive floating-point number such that $\exp(-f_{\text{large}}) = 0$. This ensures that positions where ψ is $\bot$ receive zero attention weight.

This behavior can be implemented by configuring the query and key vectors as follows:

$$\forall n, \quad \mathbf{Q}_{d,n} = \begin{cases} f_{\text{large}} & \text{if } d = d_1, \\ 0 & \text{otherwise,} \end{cases} \tag{46}$$

and

$$\mathbf{K} = \mathbf{H}^{(\ell)}(\overline{\mathbf{w}}) - 1. \tag{47}$$

where the subtraction by 1 is applied elementwise.

The corresponding attention projection weights are:

$$\mathbf{\Theta}^{\mathrm{Q}}{}_{d,:} = \begin{cases} f_{\text{large}}[\![d_b]\!]^\top & \text{if } d = d_1, \\ \mathbf{0}^\top & \text{otherwise,} \end{cases} \tag{48}$$

and

$$\mathbf{\Theta}^{\mathrm{K}}{}_{d,:} = [\![d]\!]^\top - [\![d_b]\!]^\top, \tag{49}$$

where d_b denotes the bias coordinate.

Next, we copy the values from $\mathbf{H}^{(\ell)}_{d_1,:}$ into an unused coordinate $d_{\mathbf{P}}$ via the value projection:

$$\mathbf{\Theta}^{\mathrm{V}}{}_{d,:} = \begin{cases} [\![d_1]\!]^\top & \text{if } d = d_{\mathbf{P}}, \\ \mathbf{0}^\top & \text{otherwise.} \end{cases} \tag{50}$$

Thus, $\mathbf{V}$ becomes:

$$\mathbf{V}_{d,:} = \begin{cases} \mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_1,:} & \text{if } d = d_{\mathbf{P}}, \\ \mathbf{0}^\top & \text{otherwise.} \end{cases} \tag{51}$$

Let $N_\top$ denote the number of positions $m < n$ such that $\mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_1,m} = 1$. If $N_\top > 0$, then $\boldsymbol{\alpha}_{n,m} = 1/N_\top$, giving $\mathbf{H}^{(\ell+0.5)}(\overline{\mathbf{w}})_{d_{\mathbf{P}},n} = 1$. Otherwise, if $N_\top = 0$, the attention weights are distributed uniformly over previous positions, all of which satisfy $\mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_1,m} = 0$, yielding $\mathbf{H}^{(\ell+0.5)}(\overline{\mathbf{w}})_{d_{\mathbf{P}},n} = 0$. Hence, $\mathbf{H}^{(\ell+0.5)}_{d_{\mathbf{P}},:}$ simulates $\mathbf{P}\psi$ in the ideal case.

Patch 1: Vanishing Attention Weights When too many previous positions with $\mathbf{H}^{(\ell)}_{d_1,m} = 1$, i.e., $N_\top > N_{\max}$, the resulting attention weights $\boldsymbol{\alpha}_{n,:}$ will underflow to $\mathbf{0}^\top$. This causes $\mathbf{H}^{(\ell+0.5)}(\overline{\mathbf{w}})_{d_\mathbf{P},n} = 0$ even when $\mathbf{P}\psi$ is true, leading to incorrect behavior.

To fix this, we first compute the logical conjunction of d_1 and $d_\mathbf{P}$ and store the result in $d_\wedge$ via the feedforward sublayer. When $\mathbf{H}^{(\ell+1)}_{d_\wedge,n} = 1$, it indicates that the position n satisfies ψ and there exists at least one prior position that also satisfies ψ. The number of such positions is bounded by $N_{\max}$, as starting from the $(N_{\max}+1)^{\text{th}}$ position, the first attention sublayer will suffer from vanishing attention weights, causing $\mathbf{H}^{(\ell+0.5)}(\overline{\mathbf{w}})_{d_\mathbf{P},n} = 0$.

Next, we construct a second layer in which the attention sublayer uniformly attends to positions $m < n$ where $\mathbf{H}^{(\ell+1)}(\overline{\mathbf{w}})_{d_\wedge,m} = 1$. This constructs a coordinate $d_{\mathbf{PP}}$ simulating $\mathbf{P}(\psi \wedge \mathbf{P}\psi)$. Since the number of positions where $\mathbf{H}^{(\ell+1)}_{d_\wedge,m} = 1$ is bounded, this second attention sublayer is not subject to vanishing attention weights and reliably computes its output.

Note that $\mathbf{P}(\psi \wedge \mathbf{P}\psi)$ evaluates to $\top$ when at least two earlier positions satisfy ψ. If exactly one earlier position satisfies ψ, then $\mathbf{P}\psi$ holds but $\mathbf{P}(\psi \wedge \mathbf{P}\psi)$ does not. In this case, however, the first attention sublayer (producing $d_\mathbf{P}$) already correctly simulates $\mathbf{P}\psi$, since the attention only needs to cover one position. Thus, we take the logical disjunction of $d_\mathbf{P}$ and $d_{\mathbf{PP}}$, implemented via the feedforward sublayer. The resulting coordinate $d_\vee$ correctly simulates the formula $\mathbf{P}\psi$.

Patch 2: Rounding Errors Fixed-precision arithmetic introduces rounding errors when computing the attention weights $\boldsymbol{\alpha}_{n,m}$. The fraction $1/N_\top$ may not be exactly representable by a finite-precision floating-point number. Moreover, the iterative summation in Eq. (26) can further accumulate rounding errors. Consequently, the output $\mathbf{H}^{(\ell+0.5)}_{d_\mathbf{P},n}$ may equal a value very close to 1, when it should be exactly 1. This could lead to incorrect results in subsequent computations.

We address this with a thresholding mechanism implemented via a feedforward sublayer. Define $\mathsf{round}(1)$ as the set of floating-point numbers that deviates from 1 by at most δ:

$$\mathsf{round}(1) = \{f \in \mathbb{F} \mid |f - 1| \leq \delta\}\,. \tag{52}$$

In standard floating-point systems, such as IEEE 754 [23], δ is on the order of the machine precision, e.g., approximately 2^{-24} for single precision and 2^{-11} for half precision. Let f_{small} be a small positive floating-point number such that $1/f_{\text{small}}$ is also representable, $f_{\text{small}} \cdot \frac{1}{f_{\text{small}}} = 1$, and $\min(\mathsf{round}(1)) - f_{\text{small}} > 0$. We rectify the value at $d_\mathbf{P}$ using: Let f_{small} be a small positive floating-point number such that $1/f_{\text{small}}$ is also representable, $f_{\text{small}} \cdot \frac{1}{f_{\text{small}}} = 1$, and $\min(\mathsf{round}(1)) - f_{\text{small}} > 0$. We rectify the value at $d_\mathbf{P}$ using:

$$-\frac{1}{f_{\text{small}}}\left(\mathrm{ReLU}\left(\mathbf{H}^{(\ell+0.5)}_{d_\mathbf{P},:} - f_{\text{small}}\right) - \mathrm{ReLU}\left(\mathbf{H}^{(\ell+0.5)}_{d_\mathbf{P},:}\right)\right). \tag{53}$$

This function outputs 1 if $\mathbf{H}^{(\ell+0.5)}_{d_\mathbf{P},n} \in \mathsf{round}(1)$, and 0 if $\mathbf{H}^{(\ell+0.5)}_{d_\mathbf{P},n} = 0$. To implement this with an FFN, select two hidden units d'_1, d'_2 and define:

$$\boldsymbol{\Theta}^{\text{F1}}{}_{d,:} = \begin{cases} [\![d_\mathbf{P}]\!]^\top & \text{if } d = d'_1 \text{ or } d = d'_2, \\ \mathbf{0}^\top & \text{otherwise,} \end{cases} \qquad \mathbf{b}^{\text{F1}} = -f_{\text{small}}[\![d'_1]\!]. \tag{54}$$

and

$$\boldsymbol{\Theta}^{\text{F2}}{}_{d,:} = \begin{cases} -\frac{1}{f_{\text{small}}}([\![d'_1]\!]^\top + [\![d'_2]\!]^\top) & \text{if } d = d_\mathbf{P}, \\ \mathbf{0}^\top & \text{otherwise,} \end{cases} \qquad \mathbf{b}^{\text{F2}} = \mathbf{0}. \tag{55}$$

■

So far, we have ignored layer normalization. We now show that its presence does not affect the representational power of transformers.

Proposition B.13. *If a* $\mathbf{LTL}[\mathbf{P}]$ *formula* ψ *can be simulated by a transformer without layer normalization, then it can also be simulated by a transformer with layer normalization.*

Proof. Let $\mathbf{x}$ be a vector. Layer normalization is defined as follows:

$$\text{LayerNorm}(\mathbf{x}) = \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta, \tag{56}$$

where μ and σ are the mean and standard deviation of the elements in $\mathbf{x}$, γ and β are parameters, and ϵ is a small constant for numerical stability.

We introduce, for every logical coordinate d, a mirror coordinate $\widehat{d}$ and enforce

$$\mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{\widehat{d},n} = 1 - \mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d,n} \tag{57}$$

for all sublayers ℓ and positions n. This guarantees that for every position n,

$$\mu_n = \frac{1}{2} \quad \text{and} \quad \sigma_n^2 = \frac{1}{4}. \tag{58}$$

independently of the particular truth assignment.

With this structure, we can choose the parameters of layer normalization such that the operation becomes the identity function (i.e., has no effect):

$$\gamma = \sqrt{\sigma_n^2 + \epsilon} = \sqrt{\frac{1}{4} + \epsilon}, \quad \beta = \mu_n = \frac{1}{2}. \tag{59}$$

However, the attention sublayer may produce values close to 1 but not exactly equal to it. Consequently, layer normalization can slightly perturb these values away from the intended Boolean values. We denote by $\mathsf{round}(0)$ and $\mathsf{round}(1)$ the sets of floating-point numbers that deviate from 0 and 1, respectively, by at most a small constant δ, which is again on the order of the machine precision. This deviation is harmless, since the subsequent feedforward sublayer can rectify the outputs using the following thresholding function: let d_1 be the coordinate we hope to rectify and choose $f_1, f_2 \in \mathbb{F}$ with $\max(\mathsf{round}(0)) < f_1 < f_2 < \min(\mathsf{round}(1))$. Apply:

$$\frac{1}{f_2 - f_1} \left(\left(\mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_1,:} - f_1 \right) - \text{ReLU}\left(\mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_1,:} - f_2 \right) \right). \tag{60}$$

To implement this with an FFN, select two hidden units d_1', d_2' and define:

$$\mathbf{\Theta}^{\text{F1}}{}_{d,:} = \begin{cases} [\![d_1]\!]^\top & \text{if } d = d_1' \text{ or } d = d_2', \\ \mathbf{0}^\top & \text{otherwise,} \end{cases} \qquad \mathbf{b}^{\text{F1}} = -f_1 [\![d_1']\!] - f_2 [\![d_2']\!]. \tag{61}$$

and

$$\mathbf{\Theta}^{\text{F2}}{}_{d,:} = \begin{cases} \frac{1}{f_2 - f_1} ([\![d_1']\!]^\top - [\![d_2']\!]^\top) & \text{if } d = d_1, \\ \mathbf{0}^\top & \text{otherwise,} \end{cases} \qquad \mathbf{b}^{\text{F2}} = \mathbf{0}. \tag{62}$$

This yields 1 if $\mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_1,n} \in \mathsf{round}(1)$, and 0 if $\mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{d_1,n} \in \mathsf{round}(0)$. ■

Now, we obtain the following theorem:

Theorem 3.3. *Every* **LTL**[**P**] *formula can be simulated by a transformer.*

Proof. By applying structural induction to the formula, using the lemmas Lemma B.10, Lemma B.11, and Lemma B.12, we demonstrate that each formula ψ can be simulated by dimension d_ψ in the transformer's hidden state. Furthermore, by Proposition B.13, we can nullify the effect of layer normalization.

Finally, the classification head only needs to copy the relevant dimension d_ψ as the output. ■

C Characterizations of LTL[P]

In this section, we introduce three alternative formalisms and show that they are equivalent in expressive power to **LTL**[**P**].

C.1 Left-deterministic polynomials

A **monomial** over an alphabet Σ is a language of the form:

$$\Sigma_0^* \mathtt{a}_1 \Sigma_1^* \cdots \mathtt{a}_n \Sigma_n^*, \tag{63}$$

where $\mathtt{a}_1, \ldots, \mathtt{a}_k \in \Sigma$ and $\Sigma_0, \Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$.

A monomial is called

- **left-deterministic** if for every $k \in \{1, \ldots, n\}$, $\mathtt{a}_k \notin \Sigma_{k-1}$,
- **right-deterministic** if for every $k \in \{1, \ldots, n\}$, $\mathtt{a}_k \notin \Sigma_k$,
- **unambiguous** if it is either left-deterministic or right-deterministic.

A **polynomial** is a finite union of monomials, and it is said to be left-deterministic (resp. right-deterministic or unambiguous) if it is a finite disjoint union of left-deterministic (resp. right-deterministic or unambiguous) monomials.

Example:
The language FIRST $\mathtt{b}\Sigma^*$ is left-deterministic and therefore definable in **LTL**[**P**]. In contrast, the language LAST $\Sigma^*\mathtt{b}$ is right-deterministic but not left-deterministic and thus not recognizable by **LTL**[**P**].

C.2 $\mathcal{R}$-trivial monoids

A **semigroup** is a set endowed with an associative operator, and a **monoid** is a semigroup additionally endowed with an identity element. A canonical example is the set Σ^*, with string concatenation as the associative operation and the empty string ε as the identity element. A monoid $\mathbb{M}$ is said to be $\mathcal{R}$-**trivial** (resp. $\mathcal{L}$-**trivial**) if for all elements $s, t \in \mathbb{M}$, the condition $s\mathbb{M} = t\mathbb{M}$ (respectively, $\mathbb{M}s = \mathbb{M}t$) implies $s = t$.

We now define an equivalence relation on Σ^* known as the **syntactic congruence**. Given a language $\mathbb{L} \subseteq \Sigma^*$, two strings $\mathbf{s}, \mathbf{t} \in \Sigma^*$ are **syntactically equivalent**, written $\mathbf{s} \equiv_{\mathbb{L}} \mathbf{t}$, if and only if:

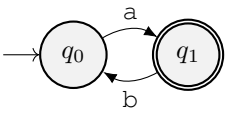
$$\forall \mathbf{u}, \mathbf{v} \in \Sigma^* : \mathbf{usv} \in \mathbb{L} \Leftrightarrow \mathbf{utv} \in \mathbb{L}. \tag{64}$$

The equivalence class of a string $\mathbf{w}$ is denoted as $[\mathbf{w}]$. The quotient monoid $\Sigma^*/\equiv_{\mathbb{L}}$, i.e., the set of equivalence classes under $\equiv_{\mathbb{L}}$, is called the **syntactic monoid** of the language $\mathbb{L}$.

C.3 Partially ordered DFAs

PODFA is introduced in §4. Constructing the DFA for a given language is a useful method for assessing whether the language is definable in **LTL**[**P**].

Example:
It is non-trivial to determine whether certain languages can be defined in **LTL**[**P**]. However, it becomes clearer when we consider its corresponding automaton. For instance, consider the language DYCK-$(1, 1)$, written as $(\mathtt{ab})^*$ in regular expression, the Dyck language over one type of parentheses with nesting depth limited to 1. The minimal DFA accepting this language is shown below:



This DFA is not partially ordered, as it can revisit both q_1 and q_0. Therefore, DYCK-$(1, 1)$ is not definable in **LTL**[**P**].

Table 2: Relations between logics, formal languages, monoids, and finite automata.

Temporal logic	First-order logic	Formal language	Monoid	Automata
$\mathbf{LTL}[\mathbf{P}, \mathbf{F}, \mathbf{S}, \mathbf{U}]$	$\mathbf{FO}[<]$	Star-free	Aperiodic	Counter-free
$\mathbf{LTL}[\mathbf{P}, \mathbf{F}]$	$\mathbf{FO}^2[<]$	Unambiguous polynomial	DA	2-PODFA
$\mathbf{LTL}[\mathbf{P}]$	$\mathbf{PFO}^2[<]$	Left-deterministic polynomial	$\mathcal{R}$-trivial	PODFA

C.4 Equivalence

We first show that every PODFA can be defined in $\mathbf{LTL}[\mathbf{P}]$.

Lemma C.1. *Let $\mathcal{A} = (\Sigma, Q, q_I, F, \delta)$ be a PODFA and $\boldsymbol{w} \in \Sigma$ a string of length N. For every $q \in Q$, there exists a formula $\psi_{q>}$ in $\mathbf{LTL}[\mathbf{P}]$ such that $\mathcal{A}$ is in q upon reading $\boldsymbol{w}_{<n}$ if and only if $\boldsymbol{w}, n-1 \models \psi_{q>}$.*

Proof. We first construct a formula $\psi_{=q}$ for each state $q \in Q$, such that $\mathbf{w}, n-1 \models \psi_{=q}$ if and only if $\mathcal{A}$ is in q after reading $\mathbf{w}_{<n-1}$, i.e., *right before* reading $\mathtt{w}_{n-1}$. We proceed by induction on the partial order $\preceq$ defined over Q.

Base case. The initial state q_I can be defined by ensuring that no prior symbol caused a transition out of q_I:

$$\psi_{=q_I} = \bigwedge_{\mathtt{a} \in \{\mathtt{a} \mid \delta(q_I, \mathtt{a}) \neq q_I\}} \neg \mathbf{P} \pi_{\mathtt{a}}. \tag{65}$$

Induction step. Assume that for all $q' \preceq q$ with $q' \neq q$, the formulas $\psi_{=q'}$ have already been constructed. Then, $\mathcal{A}$ has entered q at some prior point if:

$$\psi_{<q} = \bigvee_{q' \in Q} \bigvee_{\mathtt{a} \in \{\mathtt{a} \mid \delta(q', \mathtt{a}) = q\}} \mathbf{P}(\psi_{=q'} \wedge \pi_{\mathtt{a}}). \tag{66}$$

To define $\psi_{=q}$, we must ensure that q has been entered and it is not exited yet:

$$\psi_{=q} = \psi_{<q} \wedge \bigwedge_{\mathtt{a} \in \{\mathtt{a} \mid \delta(q, \mathtt{a}) \neq q\}} \neg \mathbf{P}(\psi_{<q} \wedge \pi_{\mathtt{a}}). \tag{67}$$

Once we obtain $\psi_{=q}$, we can define $\psi_{q>}$ as follows:

$$\psi_{q>} = \bigvee_{q' \in Q} \bigvee_{\mathtt{a} \in \{\mathtt{a} \mid \delta(q', \mathtt{a}) = q\}} \psi_{=q'} \wedge \pi_{\mathtt{a}} \tag{68}$$

Finally, a string $\mathbf{w}$ is accepted by $\mathcal{A}$ if it reaches a final state:

$$\bigvee_{q \in F} \psi_q. \tag{69}$$

■

Now we move on to prove the direction from $\mathbf{LTL}[\mathbf{P}]$ to $\mathcal{R}$-trivial monoid. Our proof is inspired by the proof of Proposition 4 in Diekert et al. [11].

A key characterization of $\mathcal{R}$-trivial monoids is given by Brzozowski and Fich [4]:

Proposition C.2 (Brzozowski and Fich [4])**.** *$\mathbb{M}$ is $\mathcal{R}$-trivial if and only if there exists an integer $K > 0$ such that for all $s, t \in \mathbb{M}$, we have $(st)^K s = (st)^K$.*

Thus, we prove the following lemma as preparation:

Lemma C.3. *Let ψ be a formula in $\mathbf{LTL}[\mathbf{P}]$ with operator depth at most K and $K > 0$. For every $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{s}, \boldsymbol{t} \in \Sigma^*$, we have*

$$\boldsymbol{u}(\boldsymbol{st})^K \boldsymbol{sv} \models \psi \quad \text{if and only if} \quad \boldsymbol{u}(\boldsymbol{st})^K \boldsymbol{v} \models \psi. \tag{70}$$

Proof. The case $\mathbf{s} = \varepsilon$ is trivial, so we assume $\mathbf{s} \neq \varepsilon$.

Let $\mathbf{w} = \mathbf{u}(\mathbf{st})^K\mathbf{sv}$ and $\mathbf{w}' = \mathbf{u}(\mathbf{st})^K\mathbf{v}$. As $\mathbf{w}'$ is a subsequence of $\mathbf{w}$, we align positions in $\mathbf{w}'$ with a subset of positions in $\mathbf{w}$. Define $N = |\mathbf{w}|$. We consider position $N+1$ to point just beyond the end of $\mathbf{w}$.

We define a pair $(n, n') \in \{1, \ldots, N+1\}$ as **legal** if

$$(n = n' = N+1) \quad \vee \quad \left(\mathtt{w}_n = \mathtt{w}_{n'} \wedge (n' < \left|\mathbf{u}(\mathbf{st})^K\right| \vee n' > \left|\mathbf{u}(\mathbf{st})^K\mathbf{s}\right|)\right). \tag{71}$$

Next, define the middle block of $\mathbf{w}$ as:

$$\mathbb{B}_k = \left\{n \;\middle|\; \left|\mathbf{u}(\mathbf{st})^k\right| < n < \left|\mathbf{u}(\mathbf{st})^K\mathbf{s}\right|\right\}, \tag{72}$$

A legal pair (n, n') is said to be k**-close** if:

$$n = n' \quad \vee \quad n, n' \in \mathbb{B}_k. \tag{73}$$

We now prove the following inductive claim: Let ψ be a formula in $\mathbf{LTL}[\mathbf{P}]$ with operator depth at most k, for some $0 \leq k \leq K$. For every k-close pair (n, n'), we have

$$\mathbf{w}, n \models \psi \quad \text{ if and only if } \quad \mathbf{w}', n' \models \psi. \tag{74}$$

We proceed by induction on k.

Base case. $k = 0$: In this case, ψ is a Boolean combination of atomic formulas $\pi_{\mathtt{a}}$. (n, n') being a legal pair implies either $n = n' = N+1$ or $\mathtt{w}_n = \mathtt{w}_{n'}$, so the claim holds trivially.

Induction step. Assume the claim holds for depth $k-1$. Let $\psi_1 = \mathbf{P}\psi$ where ψ has depth $k-1$. Suppose $\mathbf{w}, n \models \psi_1$. Then there exists a position $m < n$ such that $\mathbf{w}, m \models \psi$. For $\mathbf{w}', n' \models \psi_1$ to hold, we need to identify a position $m' < n'$ such that $\mathbf{w}', m' \models \psi$. By the inductive hypothesis, it suffices for $m' < n'$ and for (m, m') to be $(k-1)$-close. There are two cases:

• Case 1: $m \notin \mathbb{B}_k$. Then we can set $m' = m$. (m, m') is $(k-1)$-close by definition. We now need to show that $m' < n'$.
– Subcase 1.1: $n, n' \in \mathbb{B}_k$. Since $m \notin \mathbb{B}_k$ and $m < n$, we can conclude $m \leq \left|\mathbf{u}(\mathbf{st})^k\right|$. Thus, $m' = m < n'$ because $n' \in \mathbb{B}_k$.
– Subcase 1.2: $n = n'$. Then we have $m' = m < n = n'$.

• Case 2: $m \in \mathbb{B}_k$. Then, as $\mathbb{B}_{k-1} \setminus \mathbb{B}_k$ covers all symbols in $\mathbb{B}_k$, there exists $m' \in \mathbb{B}_{k-1} \setminus \mathbb{B}_k$ such that $\mathtt{w}_{m'} = \mathtt{w}_m$. Again, (m, m') is $(k-1)$-close since $m, m' \in \mathbb{B}_{k-1}$. We need to show that $m' < n'$:
– Subcase 2.1: $n, n' \in \mathbb{B}_k$. Since $m' \in \mathbb{B}_{k-1} \setminus \mathbb{B}_k$ and $n' \in \mathbb{B}_k$, we have $m' < n'$.
– Subcase 2.2: $n = n'$. Since $m' \in \mathbb{B}_{k-1} \setminus \mathbb{B}_k$ and $m \in \mathbb{B}_k$, we have $m' < m < n = n'$.

The converse direction, from $\mathbf{w}', n' \models \psi$ to $\mathbf{w}, n \models \psi$ follows analogously.

By applying the claim with $k = K$, $n = N+1$, and $n' = N+1$, we conclude that $\mathbf{w} \models \psi$ if and only if $\mathbf{w}' \models \psi$. ■

This result then follows straightforwardly:

Lemma C.4. *Every* $\mathbf{LTL}[\mathbf{P}]$*-definable language has a* $\mathcal{R}$*-trivial syntactic monoid.*

Proof. Lemma C.3 suffices to show that every $\mathbf{LTL}[\mathbf{P}]$-definable language satisfies the characterization in Proposition C.2. ■

We now have all the pieces in place to prove the main theorem.

Theorem 4.1. *Let* $\mathbb{L} \subseteq \Sigma^*$ *be a regular language,* $\mathbb{M}$ *be its syntactic monoid, and* $\mathcal{A}$ *be the DFA accepting it. The following assertions are equivalent: (1)* $\mathbb{L}$ *is a left-deterministic polynomial, (2)* $\mathbb{M}$ *is* $\mathcal{R}$*-trivial, (3)* $\mathcal{A}$ *is partially ordered, and (4)* $\mathbb{L}$ *is definable by* $\mathbf{LTL}[\mathbf{P}]$.

Proof. Brzozowski and Fich [4] showed the equivalence of (1), (2), and (3). To complete the picture, it remains to incorporate (4), which follows from Lemma C.1 and Lemma C.4. ■

The relationships between various temporal logics, first-order logics, formal languages, monoids, and finite automata are summarized in Tab. 2. We treat the last row (for $\mathbf{LTL}[\mathbf{P}]$) in this paper. For further details on the other classes ($\mathbf{LTL}[\mathbf{P}, \mathbf{F}]$ and $\mathbf{LTL}[\mathbf{P}, \mathbf{F}, \mathbf{S}, \mathbf{U}]$), interested readers can refer to McNaughton and Papert [28] and Tesson and Thérien [45].

D Transformer Language Models

The transformer architecture follows the specification in §B.1, with one modification: the final output layer is replaced by a language modeling head $\mathbf{L} \colon \mathbb{F}^D \to \mathbb{F}^{|\overline{\Sigma} \cup \{\textsc{unk}\}|}$. $\mathbf{L}$ is defined as follows:

$$\mathbf{L}(\mathbf{H}^{(L)}(\mathbf{w}_{<n})) = \operatorname{softmax}\left(\mathbf{\Theta}^{\mathrm{L}} \mathbf{H}^{(L)}(\mathbf{w})_{:,n-1} + \mathbf{b}^{\mathrm{L}}\right) \tag{75}$$

where $\mathbf{\Theta}^{\mathrm{L}} \in \mathbb{F}^{(|\overline{\Sigma}|+1) \times D}$ and $\mathbf{b}^{\mathrm{L}} \in \mathbb{F}^{|\overline{\Sigma}|+1}$ are learnable parameters.

We have laid the groundwork that makes the proofs of the following two theorems straightforward.

Theorem 5.1. *Every transformer LM can be simulated by* $\mathbf{PFO}^2[<]$.

Proof. The mapping $\mathbf{L}$ consists of a linear layer followed by a $\mathrm{softmax}$. Since both the linear layer and $\mathrm{softmax}$ are definable in $\mathbf{PFO}^2[<]$ as shown in §B.2, the result follows directly. ■

Theorem 5.2. *Every PODFA LM can be simulated by a transformer LM.*

Proof. Let $\mathcal{A} = (\Sigma, Q, q_I, F, \delta)$ be a PODFA, and let $p_{\mathcal{A}}$ be its induced LM. By Lemma C.1, for each state $q \in Q$, there exists a $\mathbf{LTL}[\mathbf{P}]$ formula ψ_q that characterizes the automaton being in state q. Then, by Theorem 3.3, each such formula can be simulated by a dedicated dimension in the hidden state of a transformer.

Since $\mathcal{A}$ is deterministic, at each position, there is exactly one coordinate that takes the value 1, while the rest are 0. Therefore, the final prediction head effectively acts as a lookup table that maps each dimension to the target distribution $\overrightarrow{p}_{\mathcal{A}}$ induced by $\mathcal{A}$, which depends solely on the state $\mathcal{A}$ is in. ■

E Experiments

In this section, we present additional empirical results that align with our theoretical findings.

E.1 Language recognition

We supplement §6.1 with detailed descriptions of the experimental setup, tasks, and results.

E.1.1 Experimental setup

Our experimental setup follows Deletang et al. [10], Butoi et al. [6]. We use a transformer with soft attention, strict future masking, $L = 5$ layers, model size $D = 64$, and NoPE. Training strings are of length up to 40, while test strings range from length 41 to 500. The model is trained for 1,000,000 steps with a batch size of 128. For evaluation, we generate 512 samples per test length. For comparison, we also train a long short-term memory (LSTM) [20] with a hidden size of 256. Each experiment is run with 5 different random seeds and 3 learning rates $(1 \times 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4})$.

All experiments were conducted on a single GPU with 24 GB of memory, each taking approximately one hour to complete. Our code is adapted from `https://github.com/google-deepmind/neural_networks_chomsky_hierarchy`, licensed under the Apache License, Version 2.0.

E.1.2 Languages

We consider five language classes, arranged in a strict inclusion hierarchy—each class is a proper subset of the one preceding it. For each class, we select one or more representative languages.

Counter languages. Counter languages are languages that can be recognized by counter machines—finite-state automata augmented by a number of counters [15, 30]. In our experiments, we choose one of the simplest counter languages: CNT $\mathtt{a}^n\mathtt{b}^n$, the set of strings consisting of an arbitrary number of $\mathtt{a}$s followed by an equal number of $\mathtt{b}$s.

Regular languages. A defining characteristic of non-star-free regular languages is modular counting. For instance, PARITY $\mathtt{a}^*(\mathtt{ba}^*\mathtt{ba}^*)^*$, the language of binary strings with an even number of $\mathtt{b}$s, is one of the simplest instances of this class.

Star-free languages. Languages that are definable in $\mathbf{LTL}[\mathbf{P}, \mathbf{F}, \mathbf{S}, \mathbf{U}]$. We focus on three common examples that are not definable in $\mathbf{LTL}[\mathbf{P}, \mathbf{F}]$:

- DYCK-$(1, 2)$: $(\mathtt{a}(\mathtt{ab})^*\mathtt{b})^*$, the Dyck language (well-balanced parentheses) with 1 pair of parentheses, limited to depth 2.
- DYCK-$(1, 1)$: $(\mathtt{ab})^*$, the Dyck language with 1 pair of parentheses, limited to depth 1. This is also a canonical example of strictly locally testable languages, where string membership depends only on adjacent symbol blocks [28].
- LT-2: $\Sigma^*\mathtt{ab}\Sigma^*$ with $|\Sigma| > 2$, the set of strings containing $\mathtt{ab}$ as a substring (symbols appearing contiguously). This is an example of a locally testable language [28].

Unambiguous polynomials. Languages that are definable in $\mathbf{LTL}[\mathbf{P}, \mathbf{F}]$. We are interested in those not definable in $\mathbf{LTL}[\mathbf{P}]$, i.e., right-deterministic but not left-deterministic polynomials. We select two such languages:

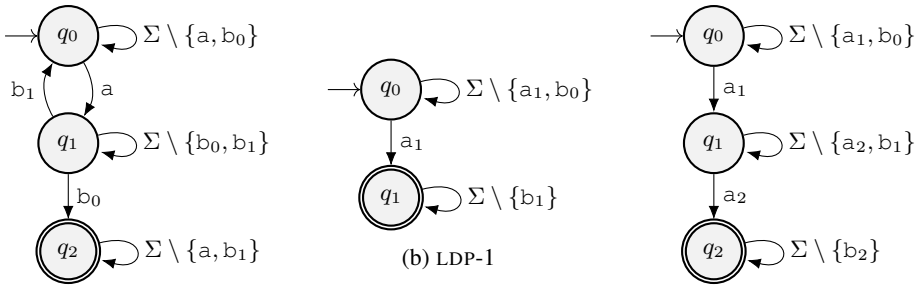
- RDP-1: $(\Sigma \setminus \{\mathtt{b}_0\})^*\mathtt{a}(\Sigma \setminus \{\mathtt{a}, \mathtt{b}_1\})^*$, a simple right-deterministic monomial.
- LAST: $\Sigma^*\mathtt{b}$, the language of strings ending with $\mathtt{b}$. It can be seen as the simplest representative of the class.

Left-deterministic polynomials. We now consider languages that are definable in $\mathbf{LTL}[\mathbf{P}]$, selecting five examples that serve as contrasts to the previously discussed ones.:

- PT-2: $\Sigma^*\mathtt{a}\Sigma^*\mathtt{b}\Sigma^*$ with $|\Sigma| > 2$, the language of strings that contain $\mathtt{ab}$ as a subsequence. Languages of this type are known as piecewise testable languages [40].
- LT-1: $\Sigma^*\mathtt{a}\Sigma^*$, the set of strings that contain $\mathtt{a}$ as a substring. This is the simplest case in both locally testable and piecewise testable languages.
- LDP-1: $(\Sigma \setminus \{\mathtt{a}, \mathtt{b}_0\})^*\mathtt{a}(\Sigma \setminus \{\mathtt{b}_1\})^*$, a left-deterministic monomial symmetrical to RDP-1.
- Although LDP-1 and RDP-1 are symmetrical, the former can be recognized by a DFA with only 2 states, whereas the latter requires 3 (see Fig. 3). For a fair comparison, we also include LDP-2 $(\Sigma \setminus \{\mathtt{a}_1, \mathtt{b}_0\})^*\mathtt{a}_1(\Sigma \setminus \{\mathtt{a}_2, \mathtt{b}_1\})^*\mathtt{a}_2(\Sigma \setminus \{\mathtt{b}_2\})^*$, which also requires 3 states.
- FIRST: $\mathtt{b}\Sigma^*$, the set of strings beginning with $\mathtt{b}$.

Sample generation. For each language, we construct both positive and negative samples. Some examples are constructed adversarially to increase the difficulty of the classification task.:

- CNT: Negative samples contain one fewer $\mathtt{a}$ or $\mathtt{b}$.
- PARITY: Negative samples contain an odd number of $\mathtt{b}$s.
- DYCK-$(1, 2)$, DYCK-$(1, 1)$: One symbol in a positive sample is flipped. Since these languages require even-length strings, we only use even-length inputs.
- LT-2 and LT-1: Negative samples omit $\mathtt{ab}$ (resp. $\mathtt{a}$) as a substring. Positive samples are constrained to include exactly one such occurrence.
- RDP-1, LDP-1, and LDP-2: Negative samples contain a single incorrect symbol.
- LAST and FIRST: Negative samples do not end or begin with $\mathtt{b}$, respectively.
- PT-2: Negative samples lack the $\mathtt{ab}$ subsequence.



(a) RDP-1 (b) LDP-1 (c) LDP-2

Figure 3: DFAs for the given languages. Nodes represent states, and arrows represent transitions. The initial state is indicated by an incoming arrow with no source node, and accepting (final) states are shown with double circles.

E.1.3 Results

We compute classification accuracy and report both the maximum and mean values across all runs in Tab. 1. The LSTM achieves perfect accuracy on all tasks, consistent with previous work showing that LSTMs can recognize regular languages [29] and implement counting mechanisms [48]. This confirms that the tasks are learnable given the available training data.

Counter languages. While several papers have shown that counting quantifiers can be simulated by arbitrary-precision [9, 49] or log-precision transformers [31], our results indicate that a fixed-precision transformer cannot recognize CNT. The model achieves a maximum accuracy of only 83.3%. This finding contrasts with Bhattamishra et al. [3], who report that transformers can recognize $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$. The discrepancy may stem from our evaluation on longer input lengths.

Regular languages. In line with Hahn [17], we find that the transformer completely fails to learn the non-star-free regular language PARITY, reaching at most 52.1% accuracy. Although Chiang and Cholak [8] design a transformer with customized positional encodings capable of recognizing PARITY, these encodings are not always representable under fixed precision, and the resulting architecture is not learnable under standard training conditions.

Star-free languages. Yang et al. [50] show that fixed-precision transformers with UHA recognize exactly the star-free languages. Similarly, Yao et al. [51] demonstrate that transformers with positional encodings of the form n/N—which are not always representable under fixed precision—can recognize bounded Dyck languages, a family of star-free languages. In contrast, we prove that transformers with soft attention and NoPE cannot recognize even the simplest bounded Dyck languages (DYCK-$(1, 2)$ and DYCK-$(1, 1)$), nor the locally testable language LT-2. These results are corroborated by our experiments: the transformer fails to learn any of these three languages. The best performance is on DYCK-$(1, 1)$, with a maximum accuracy of 87.7%, which still indicates poor generalization, especially considering the simplicity of the task.

Unambiguous polynomials. As predicted by our theory, the transformer fails to learn unambiguous polynomials that are not left-deterministic. The model achieves a maximum accuracy of 90.0% on RDP-1 and 64.8% on LAST.

Left-deterministic polynomials. Although the transformer cannot learn LT-2, it achieves perfect accuracy on both PT-2 and LT-1. In contrast to the poor performance on RDP-1 and LAST, the model learns their symmetrical counterparts (LDP-1, LDP-2, and FIRST) with 100% accuracy. Notably, Chiang and Cholak [8] construct unmasked transformer encoders capable of recognizing FIRST, but report difficulty in training such models in practice. Our results show that masked transformer decoders can learn FIRST easily and consistently, suggesting that masking may offer a more robust source of positional information than positional encodings.

Summary The empirical results align fully with our theoretical predictions. Importantly, we use single-precision (32-bit) floating-point numbers, and the string lengths never exceed the maximum

Table 3: Language modeling experiments. Maximum and mean per-token accuracies ($\pm$ standard deviation) are reported. Exact 100.0% are highlighted in bold.

Language	Max (%)	Mean (%)
RDP-1	98.3	73.6 ± 12.5
LDP-1	**100.0**	98.6 ± 2.5
LDP-2	**100.0**	98.9 ± 3.7

attention span of the transformer. That is, attention can uniformly cover all prior positions without numerical underflow or overflow. Yet, despite these favorable conditions, the transformer exhibits no expressive power beyond what is predicted by our formal characterization.

E.2 Language modeling

The DFAs corresponding to the languages used in our experiments are shown in Fig. 3, and their maximum and mean per-token accuracies are reported in Tab. 3. As predicted, the transformer language model learns left-deterministic polynomials perfectly but fails on the right-deterministic polynomial.

F Additional results

In this section, we present further theoretical results of potential interest.

F.1 Hard attention

Average hard attention (AHA) uniformly attends to positions $m < n$ with the maximum score $\mathbf{S}_{n,m}$. Formally, Eq. (24) is modified as follows:

$$\boldsymbol{\alpha}_{n,m} \stackrel{\text{def}}{=} \frac{\mathbb{1}\left\{\mathbf{S}_{n,m} = \max_{i<n} \mathbf{S}_{n,i}\right\}}{\sum_{j<n} \mathbb{1}\left\{\mathbf{S}_{n,j} = \max_{i<n} \mathbf{S}_{n,i}\right\}}, \tag{76}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function.

We now show that AHA is also definable in $\mathbf{PFO}^2[<]$.

Theorem F.1. *Every transformer with AHA can be simulated by* $\mathbf{PFO}^2[<]$.

Proof. By Proposition B.8, $\mathbf{PFO}^2[<]$ can identify the maximum score $\max_{i<n} \mathbf{S}_{n,i}$. The denominator in Eq. (76) is a sum of non-negative terms and can therefore be simulated by $\mathbf{PFO}^2[<]$ via Lemma B.5. As with the soft-attention case in Eq. (24), the resulting attention weights may vanish when the number of positions exceeds the model's bounded attention span; hence, Lemma B.7 applies here as well. The remaining steps of the construction follow identically to those in §B.2. ■

The other direction is straightforward.

Theorem F.2. *Every* $\mathbf{LTL}[\mathbf{P}]$ *formula can be simulated by a transformer with AHA.*

Proof. Recall that the attention mechanism we constructed in the proof of Lemma B.12 uniformly attends to all positions with the highest score $\mathbf{S}_{n,m} = 0$, effectively making it equivalent to AHA. Therefore, $\mathbf{LTL}[\mathbf{P}]$ formulas can also be simulated by transformers with AHA. ■

Now, we turn to UHA, another widely studied attention mechanism where each position n attends to a single position $m < n$ with the highest $\mathbf{S}_{n,m}$. In the case of ties, the *rightmost* position is selected. Yang et al. [50] prove that transformers with UHA are as expressive as $\mathbf{LTL}[\mathbf{P}, \mathbf{F}, \mathbf{S}, \mathbf{U}]$.

Combining the results from above, we have:

Corollary F.3. *Transformers with AHA are as expressive as those with soft attention, which are strictly less expressive than those with UHA.*

Proof. The equality AHA $=$ soft attention follows directly from Theorems F.1 and F.2.

The strict inequality soft attention $<$ UHA follows directly from the fact that $\mathbf{LTL}[\mathbf{P}]$ is strictly less expressive than $\mathbf{LTL}[\mathbf{P}, \mathbf{F}, \mathbf{S}, \mathbf{U}]$. ■

F.2 Non-strict future masking

Most commonly used transformers adopt non-strict future-masked soft attention, where each position is allowed to attend to itself. In this case, Eq. (24) and Eq. (26) become:

$$\boldsymbol{\alpha}_{n,m} \stackrel{\text{def}}{=} \frac{\exp(\mathbf{S}_{n,m})}{\sum_{i \leq n} \exp(\mathbf{S}_{n,i})}, \tag{77}$$

and

$$\mathbf{A}(\mathbf{H}(\overline{\mathbf{w}}))_{:,n} \stackrel{\text{def}}{=} \sum_{m \leq n} \boldsymbol{\alpha}_{n,m} \mathbf{V}_{:,m}. \tag{78}$$

respectively.

This modification, however, limits the expressiveness of the model.

Theorem F.4. *Transformers with non-strict future masking are strictly less expressive than those with strict future masking.*

Proof. Since we have already established that transformers with strict masking are as expressive as $\mathbf{PFO}^2[<]$, it suffices to show that any transformer with non-strict masking is strictly less expressive than $\mathbf{PFO}^2[<]$.

We begin by observing that transformers with non-strict masking can be simulated by $\mathbf{PFO}^2[<]$. The required modification to Proposition B.4 involves including the current position in the threshold counting quantifiers:

$$\exists^{\geq 1} y \leq x : \phi(y) \stackrel{\text{def}}{=} \left(\exists^{\geq 1} y < x : \phi(y)\right) \vee \phi(x), \tag{79a}$$

$$\exists^{\geq 2} y \leq x : \phi(y) \stackrel{\text{def}}{=} \left(\exists^{\geq 2} y < x : \phi(y)\right) \vee \left(\left(\exists^{\geq 1} y < x : \phi(y)\right) \wedge \phi(x)\right), \tag{79b}$$

$$\vdots$$

Next, we construct a language that is definable in $\mathbf{PFO}^2[<]$ but cannot be recognized by a transformer with non-strict masking. Consider the language $\mathtt{bb}\Sigma^*$, which can be defined in $\mathbf{PFO}^2[<]$ by the following sentence:

$$\exists x : (\pi_{\mathtt{b}}(x) \wedge \left(\exists^{=1} y < x : \pi_{\mathtt{b}}(y)\right) \wedge \neg \exists y < x : \neg \pi_{\mathtt{b}}(y)). \tag{80}$$

On the other hand, given any transformer $\mathbf{H}$ with non-strict masking, and for any string $\mathtt{w} \in \mathtt{bb}\Sigma^*$, we have:

$$\mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{:,2} = \mathbf{H}^{(\ell)}(\overline{\mathbf{w}})_{:,1} \quad \text{for all layers } \ell \in \{1, \ldots, L\} \tag{81}$$

This is because, under non-strict masking, position n attends to all positions $\leq n$, including itself. When the prefix consists entirely of identical symbols, each attention pattern and subsequent computation depend solely on the identical sequence of embeddings, resulting in identical representations at each position. However, differentiating the two leading $\mathtt{b}$s is essential for recognizing $\mathtt{bb}\Sigma^*$, as the corresponding DFA will enter a different state upon reading the first symbol $\mathtt{b}$. ■

Empirically, we confirm this limitation by training transformers—one with strict masking and one with non-strict masking—on the language $\mathtt{bb}\Sigma^*$. The results are consistent with our theoretical prediction:

- The strictly masked transformer achieves 100.0% maximum accuracy and $96.3\% \pm 5.8\%$ average accuracy.
- The non-strictly masked variant reaches only 95.8% maximum accuracy and $74.4\% \pm 8.2\%$ average accuracy.

F.3 Positional encodings

Positional encodings are introduced in Vaswani et al. [46] to inject information about the positions into the transformer. In general, there are two categories of positional encodings:

• **Absolute positional encoding:** This is a function $\mathbf{p}\colon \{1, \ldots, N+1\} \to \mathbb{F}^D$. It is typically injected into the input layer by modifying Eq. (21) as follows:

$$\mathbf{E}(\overline{\mathbf{w}})_{:,n} \stackrel{\text{def}}{=} \mathbf{e}(\mathtt{w}_n) + \mathbf{p}(n), \quad n \in \{1, \ldots, N+1\}. \tag{82}$$

• **Relative positional encoding:** This is a function $\mathbf{p}\colon \{1, \ldots, N+1\} \times \{1, \ldots, N+1\} \to \mathbb{F}^D$. It is typically injected into the attention sublayer by modifying Eq. (23) as follows [39, 21]:

$$\mathbf{S}_{n,m} \stackrel{\text{def}}{=} \frac{\mathbf{Q}_{:,n} \cdot \mathbf{K}_{:,m} + \mathbf{Q}_{:,n} \cdot \mathbf{p}(n,m)}{\sqrt{D}}. \tag{83}$$

In formal logic, a numerical predicate is a predicate that depends solely on the positions in a string, not the symbols in those positions. Numerical predicates that depend on one (resp. two) position(s) are referred to as unary (resp. binary) numerical predicates. For instance, the relation $<$ is a binary numerical predicate.

We now show that all absolute (resp. relative) positional encodings can be simulated by unary (resp. binary) numerical predicates.

Theorem F.5. *Let* $\mathbf{p}$ *be an absolute (resp. relative) positional encoding and* $\mathbb{L} \subseteq \Sigma^*$ *be a regular language. There exists a collection of unary (resp. binary) numerical predicates* $\mathcal{P}$ *such that the following assertions are equivalent:*

1. $\mathbb{L}$ *can be recognized by a transformer with positional encoding* $\mathbf{p}$.
2. $\mathbb{L}$ *is definable by* $\mathbf{FO}^2[<, \mathcal{P}]$*, i.e.,* $\mathbf{FO}^2[<]$ *extended with the numerical predicates in* $\mathcal{P}$.

Proof. Positional encodings, under fixed precision, have a finite image. Therefore, for every dimension $d \in 1, \ldots, D$ and every floating-point number $f \in \mathbb{F}$, we can define a numerical predicate r such that:

$$r(n) = \top \quad \text{if} \quad \mathbf{p}(n)_d = f, \tag{84}$$

or

$$r(n,m) = \top \quad \text{if} \quad \mathbf{p}(n,m)_d = f. \tag{85}$$

■

The reverse direction and a precise logical characterization of commonly used positional encodings, e.g., sinusoidal and rotary [43], are left for future work.

G Related work

The expressivity of transformers in the context of formal methods has been extensively studied in recent literature [42]. Various transformer variants have been explored, and different assumptions have been made to enhance the expressivity of transformers.

Chain of thought (CoT). CoT reasoning, which involves generating intermediate steps before arriving at a final answer, has become a popular approach. Pérez et al. [38] demonstrated that a transformer with both an encoder and a decoder, when allowed a polynomial number of intermediate computation steps, is Turing complete. Similarly, Merrill and Sabharwal [33], Nowak et al. [35], Li et al. [27] show that the expressivity of transformers can be improved with various forms of CoT reasoning. In this work, along with many others, we do not allow intermediate steps, which restricts expressivity. In such a case, Hao et al. [18], Merrill et al. [34], Merrill and Sabharwal [32], Chiang et al. [9], Yang and Chiang [49] have established various upper bounds on expressivity.

Non-fixed precision. If a transformer is allowed arbitrary precision, or if the precision increases with input length, Bhattamishra et al. [3] proved that it is more expressive than simplified and stateless counter machines. Additionally, Chiang et al. [9] and Yang and Chiang [49] showed that a transformer with such precision can recognize any language definable by temporal counting logic. In contrast, our work focuses on the scenario where precision is fixed, which imposes more limitations on expressivity.

Bespoke positional encodings. Some prior studies have employed bespoke positional encodings, many of which cannot be represented under fixed precision, to overcome certain limitations of transformers. For example, Chiang and Cholak [8] used the encoding n/N to enable transformers to recognize parity, and Barcelo et al. [2] demonstrated that transformers with a specially designed positional encoding can achieve a lower bound of $\mathbf{FO}[<]$ with unary numerical predicates. In contrast, our constructions and proofs do not rely on any form of positional encoding.

Hard attention. Two forms of hard attention have been explored in the literature. Yang et al. [50] showed that masked fixed-precision transformers with UHA and NoPE recognize exactly the star-free languages. Barcelo et al. [2] established a similar lower bound for transformers with AHA and certain positional encodings. Prior to our work, it was unclear, under fixed precision, whether these two hard attention mechanisms were more or less expressive than, or comparable to, the standard soft attention. Surprisingly, we find that UHA is strictly more expressive than soft attention, while AHA is as expressive as soft attention.

H Limitations

The primary limitation of this work lies in the omission of positional encodings. While we briefly discuss their role as numerical predicates, the exact numerical predicates simulated by commonly used positional encodings remain unknown. We hope to explore this in future work. Nevertheless, we believe it is important to first understand the expressivity of a barebones transformer architecture, as this forms the foundation for systematically incorporating various forms of positional encoding later on.

Another limitation—particularly from an empirical perspective—is our stringent evaluation criteria for transformer performance. While certain levels of accuracy might be considered successful in empirical studies, we require perfect accuracy. This is motivated by a formal perspective: a model (e.g., an automaton or logical formula) either recognizes a language or it does not; anything short of perfection is regarded as failure, suggesting that some form of approximation or shortcut has been employed. We interpret our results as identifying a class of tasks that transformers have the full capacity to solve.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Claims made in abstract and §1 accurately reflect the paper's contributions and scope.

   Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: §H.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Assumptions: §§ B.1, 3.1 and 5, proofs: §§ B.2, B.3, C, D and F.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: A description of the experimental setup and data generation is provided in §§ E.1.1, E.1.2 and 6.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.

• If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
• Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
• While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code included in the supplementary material.

Guidelines:

• The answer NA means that paper does not include experiments requiring code.
• Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
• While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
• The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
• The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
• The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
• At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
• Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: §§ E and 6

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Standard deviations are reported in Tabs. 1 and 3 and §F.2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Computer resources are reported in §E.1.1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: This research conform with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

• The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: We foresee no societal impact of this work.

Guidelines:

• The answer NA means that there is no societal impact of the work performed.
• If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
• Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
• The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
• The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
• If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks

Guidelines:

• The answer NA means that the paper poses no such risks.
• Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
• Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
• We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Our code is adapted from Deletang et al. [10]. The authors are properly credited and the license and terms are explicitly mentioned and properly respected in §E.1.1.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: This paper does not introduce new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This research does not involve crowdsourcing.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This research does not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

• We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
• For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: This research does not involve LLMs.

Guidelines:

• The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
• Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.