Unique Hard Attention: A Tale of Two Sides

Selim Jerad   Anej Svete   Jiaoda Li   Ryan Cotterell

{sjerad, anej.svete, jiaoda.li, ryan.cotterell}@ethz.ch

ETHzürich

"A wonderful fact to reflect upon, that leftmost and rightmost unique hard attention are constituted to be profoundly distinct."

Abstract

Understanding the expressive power of transformers has recently attracted attention, as it offers insights into their abilities and limitations. Many studies analyze unique hard attention transformers, where attention selects a single position that maximizes the attention scores. When multiple positions achieve the maximum score, either the rightmost or the leftmost of those is chosen. In this paper, we highlight the importance of this seeming triviality. Recently, finite-precision transformers with both leftmost- and rightmost-hard attention were shown to be equivalent to Linear Temporal Logic (**LTL**). We show that this no longer holds with only leftmost-hard attention—in that case, they correspond to a *strictly weaker* fragment of **LTL**. Furthermore, we show that models with leftmost-hard attention are equivalent to *soft* attention, suggesting they may better approximate real-world transformers than right-attention models. These findings refine the landscape of transformer expressivity and underscore the role of attention directionality.

1 Introduction

Much work has recently been done on understanding the capabilities and limitations of transformers (Vaswani et al., 2017). Collectively, the body of work on the representational capacity of transformers has provided a nuanced picture of the landscape (Pérez et al., 2021; Hahn, 2020; Chiang and Cholak, 2022; Hao et al., 2022; Merrill et al., 2022a; Merrill and Sabharwal, 2023; Chiang et al., 2023; Yang et al., 2024a; Strobl et al., 2024; Svete and Cotterell, 2024; Nowak et al., 2024; Li and Cotterell, 2025, *inter alia*).[1] Any such investigation begins by choosing an idealization of the architecture. A common modeling decision is to use **unique hard attention** (UHA), which selects a single position maximizing the attention scores (Hahn, 2020; Hao et al., 2022; Barceló et al., 2024). In one of the first exact descriptions of UHA expressivity, Yang et al. (2024a) show that UHA transformers (UHATs) with no positional encodings, strict future masking, and *either leftmost* or *rightmost* tiebreaking are equivalent to linear temporal logic, **LTL**. This connects UHATs to well-understood formalism such as the star-free languages and counter-free automata. An incautious reading of Yang et al.'s result could lead one to generalize the equivalence to all UHATs. However, we zoom in on the overlooked choice of tiebreaking and show that it markedly impacts the model's expressivity.

We show UHATs with *only leftmost* tiebreaking are strictly less expressive by relating them to a fragment of **LTL**, **LTL**[◇]. We do so by adapting Yang et al.'s (2024a) proofs: We describe a variant of the **B-RASP** programming language defined therein, restricted to leftmost tiebreaking, and show it to be equivalent to **LTL**[◇]. Further, leveraging the recent results by Li and Cotterell (2025) that characterize finite-precision future-masked *soft*-attention transformers as equivalent to **LTL**[◇], we establish the equivalence of leftmost UHATs to standard softmax transformers. Moreover, we give explicit **LTL**[◇] formulas and partially ordered finite-state automata (FSAs) describing leftmost UHATs.

2 Transformer Idealization

This section introduces the idealization of the transformer analyzed throughout the paper.

Finite Precision.   Implemented on modern hardware, computations performed by transformers rely on a fixed number of bits. This makes finite-precision a more realistic assumption than unbounded or growing (w.r.t. input length) precision (Barceló et al., 2024; Hahn, 2020; Hao et al., 2022; Merrill et al., 2022b; Merrill and Sabharwal, 2023).

---

[1] App. A holds a more detailed overview of related work.

No positional encodings. We wish to study what we consider to be a *barebone* idealization of the transformer, because this enables us to understand the exact expressive power of UHA. Moreover, finite-precision positional encodings correspond precisely to monadic predicates in **LTL** formulas (Yang et al., 2024a), yielding a predictable and well-understood extension to our analysis.

Unique hard attention. While the original transformer uses soft attention (Vaswani et al., 2017), theoretical work largely analyzes hard attention (Merrill et al., 2022b; Hao et al., 2022; Yang et al., 2024a; Barceló et al., 2024, *inter alia*). The precise implications of this modeling decision are still unclear, but our work, combined with Li and Cotterell's (2025) results, reduces the gap between both models: Soft-attention transformers are equivalent to leftmost UHATs while rightmost UHATs are more expressive. We contextualize our findings with some more related results in Tab. 1.

Strict future masking. Future masking is standard in transformer-based language models. We focus on *strict* masking (where a position cannot attend to itself) as non-strict masking is known to reduce expressive power (Yang et al., 2024a). Moreover, residual connections still allow the model to incorporate information vertically across layers.

3 The Best of UHA, The Worst of UHA

This section provides a high-level overview, intuition, roadmap, and key implications of our results.

3.1 Separation

We begin by building an intuition as to why, with future masking, UHA with leftmost tiebreaking ◄ is strictly less expressive than with rightmost tiebreaking ►. We illustrate this in **B-RASP**, a Boolean-valued programming language (Yang et al., 2024a), as the intermediary between **LTL** and UHATs. To follow the coming examples, we only need familiarity with the following attention operation:

$$P(t) = \blacktriangleleft_{t'}[t' < t, s(t')] \, v(t') : d(t) \qquad (1)$$

$\blacktriangleleft_{t'}[t' < t, s(t')]$ denotes choosing (*attending to*) the *leftmost* (◄) position t' for which $t' < t$ holds (Future masking) and $s(t') = 1$. If such a t' exists, $P(t)$ is assigned the value of the predicate $v(t')$, and otherwise, it is assigned a default value $d(t)$. This emulates leftmost UHA, where $t' < t$

corresponds to strict future masking, $s(t') = 1$ corresponds to maximizing the attention score, and $v(t')$ corresponds to the value vector. We define the rightmost operation ► analogously.

We now note two facts:

(i) Every leftmost operation ◄

$$P(t) = \blacktriangleleft_{t'}[t' < t, s(t')] \, v(t') : d(t) \qquad (2)$$

can be simulated by ► attentions in two steps. First, we gather all positions t that have preceding positions $t' < t$ such that $s(t') = 1$:

$$P_1(t) = \blacktriangleright_{t'}[t' < t, s(t')] \, s(t') : 0. \qquad (3)$$

Then, the leftmost position is just the single position t' with $s(t') = 1$ but $P_1(t') = 0$:

$$P_2(t) = \blacktriangleright_{t'}[t' < t, s(t') \wedge \neg P_1(t')] \, v(t') : d(t). \qquad (4)$$

(ii) There exist operations that ► *can* perform and ◄ *cannot*. For instance, ► attention can read the value immediately to the left of the current position, i.e., $v(t - 1)$, as follows:

$$P(t) = \blacktriangleright_{t'}[t' < t, 1] \, v(t') : 0, \qquad (5)$$

but ◄ attention cannot, as we would need $t' = t-1$ to be the only position with $t' < t$ and $s(t') = 1$ for all $t \in [T]$, which is impossible.

This establishes a separation between **B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$**, which is limited to leftmost tiebreaking ◄ and future masking F, and the full **B-RASP**, leading to the following:

Theorem 3.1 (Informal). *Finite-precision future-masked ◄ UHATs are* weaker *than ► UHATs.*

3.2 Characterizations

In the remainder of the paper, we show that **B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$** is equivalent to the fragment of **LTL** with only the $\diamondsuit$ operator, denoted by **LTL**$[\diamondsuit]$, which in turn is equivalent to partially ordered FSAs (POFA). This provides an exact characterization of leftmost UHATs:

Theorem 3.2 (Informal). *Finite-precision future-masked ◄ UHATs are equivalent to* **LTL**$[\diamondsuit]$.

In §6, we formalize the theorem with proofs intentionally made analogous to Yang et al.'s (2024a), in order to highlight the difference between ◄ and ►. Additionally, in §7, we provide alternative proofs that directly translate ◄ UHATs to **LTL**$[\diamondsuit]$ formulas and POFA.

Transformer	LTL	FO logic	Regex	Monoid	Automata	Note
$\mathcal{T}_{\blacktriangleright}^{\mathsf{F}}, \mathcal{T}_{\blacktriangleleft}^{\mathsf{P}}$	**LTL**$[\diamondsuit, \diamondsuit\!\!\!\!\diamond, \mathbf{S}, \mathbf{U}]$	**FO**$[<]$	star-free	aperiodic	counter-free	Yang et al. (2024a, Thm. 5)
future-masked soft attention	**LTL**$[\diamondsuit]$	**PFO**$^2[<]$	$\mathcal{R}$-expression	$\mathcal{R}$-trivial	POFA	Li and Cotterell (2025)
$\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$	**LTL**$[\diamondsuit]$	**PFO**$^2[<]$	$\mathcal{R}$-expression	$\mathcal{R}$-trivial	POFA	Thms. 6.1 and 6.2
$\mathcal{T}_{\blacktriangleright}^{\mathsf{P}}$	**LTL**$[\diamondsuit\!\!\!\!\diamond]$	**FFO**$^2[<]$	$\mathcal{L}$-expression	$\mathcal{L}$-trivial	RPOFA	Thm. E.1

Table 1: Known equivalences of finite-precision transformers with no positional encodings to different formalism. $\mathcal{T}_{\blacktriangleright}^{\mathsf{F}}$ future-masked rightmost UHATs. $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}, \mathcal{T}_{\blacktriangleleft}^{\mathsf{P}}$, and $\mathcal{T}_{\blacktriangleright}^{\mathsf{P}}$ are defined analogously for past masking and leftmost UHA.

3.3 Implications

Combining with Li and Cotterell's (2025) results that show that soft and average hard attention transformers[2] are equivalent to **LTL**$[\diamondsuit]$ as well, we discover the peculiar fact that with fixed precision, soft attention, average hard attention, and leftmost UHA are all equally expressive.

This insight could shed light on certain observed phenomena in soft-attention transformers. For instance, Liu et al. (2023) find that transformers struggle with the flip-flop language, where the symbol following a "read" instruction must match the symbol following the most recent "write" instruction. Our results suggest that this difficulty arises because leftmost UHATs and thereby soft-attention transformers lack the ability to locate the most recent—rightmost—write instruction.

Furthermore, the fact that rightmost UHA is strictly more expressive than other variants of transformers may partly explain the empirical success of positional encodings that bias attention toward recent tokens, such as ALiBi (Press et al., 2022), as they help approximate rightmost tiebreaking.

4 Linear Temporal Logic

An **alphabet** Σ is a finite, non-empty set of **symbols**. The **Kleene closure** of Σ is $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$, the set of all strings, where $\Sigma^0 \stackrel{\text{def}}{=} \{\varepsilon\}$ contains only the empty string. A **language** $\mathbb{L}$ is a subset of Σ^*. We treat a **language recognizer** as a function $\mathtt{R}: \Sigma^* \to \{0, 1\}$ whose language is $\mathbb{L}(\mathtt{R}) \stackrel{\text{def}}{=} \{\boldsymbol{w} \in \Sigma^* \mid \mathtt{R}(\boldsymbol{w}) = 1\}$. Two recognizers $\mathtt{R}_1$ and $\mathtt{R}_2$ are **equivalent** if and only if $\mathbb{L}(\mathtt{R}_1) = \mathbb{L}(\mathtt{R}_2)$.

Linear temporal logic **LTL**$[\diamondsuit, \diamondsuit\!\!\!\!\diamond, \mathbf{S}, \mathbf{U}]$ is an extension of Boolean logic that considers events over time and can express time-dependent properties (Pnueli, 1977). We define **LTL** over (finite) strings. Formulas in **LTL**$[\diamondsuit, \diamondsuit\!\!\!\!\diamond, \mathbf{S}, \mathbf{U}]$ are composed of atomic formulas π_a for $a \in \Sigma$, Boolean connectives $\wedge, \neg,$[3] and four **temporal operators** $\diamondsuit$ (past), $\diamondsuit\!\!\!\!\diamond$ (future), $\mathbf{S}$ (since) and $\mathbf{U}$ (until). We denote by **LTL**$[\diamondsuit]$ the fragment with only the $\diamondsuit$ operator and Boolean connectives and by **LTL**$[\diamondsuit\!\!\!\!\diamond]$ the fragment with only the $\diamondsuit\!\!\!\!\diamond$ operator and Boolean connectives.

Given a string $\boldsymbol{w} = w_1 \cdots w_T$, **LTL** formulas are interpreted at some position $t \in [T]$. We write $\boldsymbol{w}, t \models \psi$ to denote that ψ is TRUE on $\boldsymbol{w}$ at position t. The semantics for **LTL**$[\diamondsuit]$ are:[4]

- $\boldsymbol{w}, t \models \pi_a \iff w_t = a$;
- $\boldsymbol{w}, t \models \psi_1 \wedge \psi_2 \iff \boldsymbol{w}, t \models \psi_1 \wedge \boldsymbol{w}, t \models \psi_2$;
- $\boldsymbol{w}, t \models \neg\psi \iff \boldsymbol{w}, t \not\models \psi$;
- $\boldsymbol{w}, t \models \diamondsuit\psi \iff \exists t' < t : \boldsymbol{w}, t' \models \psi$.

To define string acceptance, we denote by $T + 1$ a position outside of the string and define

$$\boldsymbol{w} \models \psi \Leftrightarrow \boldsymbol{w}, T + 1 \models \psi. \tag{6}$$

5 B-RASP and B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$

We now introduce **B-RASP** in more detail. The input to a **B-RASP** program is a string $\boldsymbol{w} \in \Sigma^*$ with $|\boldsymbol{w}| = T$. On such an input, a **B-RASP program** computes a sequence of Boolean vectors of size T, with entries indexed by $t \in [T]$ in parentheses, denoted as $P(t)$. Each $w \in \Sigma$ gives rise to an atomic Boolean vector Q_w, defined as follows. For each $t \in [T]$:

$$Q_w(t) = 1 \iff \boldsymbol{w}_t = w \tag{7}$$

To streamline notation, we denote the first $|\Sigma|$ vectors of the program Q_w by $P_1, \cdots, P_{|\Sigma|}$. The $(i+1)^{\text{th}}$ vector $P_{i+1} \stackrel{\text{def}}{=} P'$ is computed inductively using one of the following operations:

(1) **Position-wise operation**: $P'(t)$ is a Boolean combination of zero or more of $\{P_{i'}(t)\}_{i'=1}^i$.

---

[2]Average hard attention divides attention mass equally among positions maximizing the attention score.

[3]$\vee$ can be defined using $\wedge$ and $\neg$.

[4]We refer to App. B for more details on **LTL**$[\diamondsuit, \diamondsuit\!\!\!\!\diamond, \mathbf{S}, \mathbf{U}]$ and its equivalent formalisms.

(2) **Attention operation**: $P'(t)$ can be one of:

$$P'(t) = \blacktriangleleft_{t'}[m(t, t'), s(t, t')]\ v(t, t') : d(t) \quad (8a)$$

$$P'(t) = \blacktriangleright_{t'}[m(t, t'), s(t, t')]\ v(t, t') : d(t) \quad (8b)$$

where:

- The **mask predicate** m is defined as either $m(t, t') \overset{\text{def}}{=} \mathbb{1}\{t' < t\}$ for **strict future masking** (F), or $m(t, t') \overset{\text{def}}{=} \mathbb{1}\{t' > t\}$ for **strict past masking** (P). Notice that the inequalities are *strict*, meaning the current position is excluded from attention. This detail has been shown to *increase* expressivity compared to non-strict masking (Yang et al., 2024a; Li and Cotterell, 2025).[5]
- The **score predicate** $s(t, t')$ is a Boolean combination of $\{P_{i'}(t)\}_{i'=1}^i \cup \{P_{i'}(t')\}_{i'=1}^i$.
- the **value predicate** $v(t, t')$ is defined analogously, and
- the **default value predicate** $d(t)$ is a Boolean combination of values in $\{P_1(t), \ldots, P_i(t)\}$.

We use $\blacktriangleleft$ to denote **leftmost** tiebreaking and $\blacktriangleright$ to denote **rightmost** tiebreaking. For $t \in [T]$, define the set of valid positions as:

$$\mathcal{N}(t) \overset{\text{def}}{=} \{t' \in [T] \mid m(t, t') = 1 \text{ and } s(t, t') = 1\}. \quad (9)$$

The unique position to attend to is then selected as:

$$t^* \overset{\text{def}}{=} \begin{cases} \min \mathcal{N}(t) & \text{if } \blacktriangleleft \\ \max \mathcal{N}(t) & \text{if } \blacktriangleright \end{cases}. \quad (10)$$

Finally, the semantics of the attention operation are given by:

$$P'(t) \overset{\text{def}}{=} \begin{cases} v(t, t^*) & \text{if } |\mathcal{N}(t)| > 0 \\ d(t) & \text{otherwise} \end{cases}. \quad (11)$$

Note that, by Lem. 12 and Prop. 11 of Yang et al. (2024a), we can rewrite every $\mathcal{P}$ to an equivalent program where every attention operation only uses unary scores and unary values, i.e., $s(t, t')$ and $v(t, t')$ depend only on t'.

B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$ is the restricted version of **B-RASP** with only leftmost tiebreaking and future masking.

To define $\mathcal{P}$'s language, we designate a final output vector Y and T as the output position such that $Y(T) = 1$ signals acceptance.

B-RASP is equivalent to finite-precision future-masked rightmost UHATs, $\mathcal{T}_{\blacktriangleright}^{\mathsf{F}}$ (Yang et al., 2024a, Thms. 3 and 4). A similar claim, proved in App. C, holds for **B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$** and leftmost UHATs, $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$.

---

[5]Such UHATs can still access information at the current position via the residual connection; similarly, **B-RASP** programs can do so using the default predicate d.

Theorem 5.1. *For any UHAT in $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$, there exists an equivalent B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$ program. For any B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$ program, there exists an equivalent UHAT in $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$.*

6 B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$ Is Equivalent to LTL[$\diamondsuit$]

We now formalize the equivalence of **B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$** and **LTL[$\diamondsuit$]**. It rests on the following two theorems. The proofs are provided in App. C. They are adapted from Yang et al. (2024a), with the differing parts highlighted in red.

Theorem 6.1. *For any formula ψ of LTL[$\diamondsuit$], there is a B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$ program with a Boolean vector P_ψ such that, for any input $\boldsymbol{w}$ of length T and all $t \in [T]$, we have $\boldsymbol{w}, t \models \psi \iff P_\psi(t) = 1$.*

Theorem 6.2. *For any Boolean vector P of a B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$ program $\mathcal{P}$, there is a formula ψ_P of LTL[$\diamondsuit$] such that for any input $\boldsymbol{w}$ of length T and all $t \in [T]$, we have $P(t) = 1 \iff \boldsymbol{w}, t \models \psi_P$.*

Yang et al. (2024a, Thm. 15) establish an alternative proof of the equivalence between **B-RASP** and full **LTL[$\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}$]** via counter-free automata, which recognize the class of star-free languages. Analogously, we demonstrate that **B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$** corresponds to POFAs, a subclass of counter-free automata that characterize **LTL[$\diamondsuit$]**. A translation from POFAs to **B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$** is provided in App. C.1.

7 Direct Descriptions of $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$

We now describe an alternative description of $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ that *directly* translates it to **LTL[$\diamondsuit$]** and POFAs.

7.1 Describing $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ with LTL[$\diamondsuit$]

In a transformer, the contextual representation at layer ℓ, $\boldsymbol{x}_t^{(\ell)}$, determines a function that computes the next representation, $\boldsymbol{x}_t^{(\ell+1)}$, given the unmasked symbols using the attention mechanism. In UHATs, this function is particularly simple: It computes $\boldsymbol{x}_t^{(\ell+1)}$ by *selecting* the symbol with the highest attention score (as per the tiebreaking mechanism), $\boldsymbol{x}_{t^*}^{(\ell)}$, and combines it with $\boldsymbol{x}_t^{(\ell)}$ via the residual connection: $\boldsymbol{x}_t^{(\ell+1)} = \boldsymbol{x}_t^{(\ell)} + \boldsymbol{x}_{t^*}^{(\ell)}$; see Fig. 1.

This invites the interpretation of transformer layers collecting progressively richer representations of individual symbols by selecting a new representation to append at each layer. We translate this idea into a set of **LTL[$\diamondsuit$]** formulas of the form $\phi_{\boldsymbol{x}_t^{(\ell)} \leftarrow \boldsymbol{x}_{t^*}^{(\ell)}}^{(\ell)}$ that keep track of the fact that the representation $\boldsymbol{x}_t^{(\ell)}$ was updated with the representation
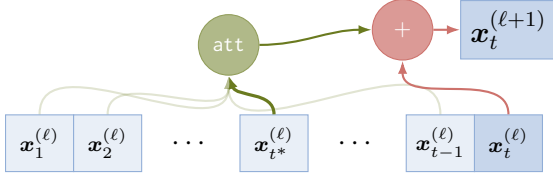
Figure 1: Unique hard attention. $\boldsymbol{x}_{t*}^{(\ell)}$ is combined with $\boldsymbol{x}_t^{(\ell)}$ to compute $\boldsymbol{x}_t^{(\ell+1)} = \boldsymbol{x}_t^{(\ell)} + \boldsymbol{x}_{t*}^{(\ell)}$.

$\boldsymbol{x}_{t*}^{(\ell)}$ at layer ℓ. The full formula is presented in the proof of the following theorem in App. D.2.

Theorem 7.1. *Let* $\mathsf{T} \in \mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ *be a transformer. Then, there exists an equivalent formula* $\psi \in \mathbf{LTL}[\diamondsuit]$.

7.2 Describing $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ with POFAs

To provide an automata-theoretic take on the result, we directly express $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ transformers with POFAs in the proof of the following theorem in App. D.3.

Theorem 7.2. *Let* $\mathsf{T} \in \mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ *be a transformer. Then, there exists an equivalent POFA.*

8 Discussion and Conclusion

We establish the equivalence of future-masked finite-precision leftmost UHATs with no positional encodings, $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$, to a fragment of linear temporal logic, $\mathbf{LTL}[\diamondsuit]$. Together with Yang et al.'s and Li and Cotterell's results, this largely completes the picture of finite-precision transformer expressivity.

Equivalence to soft-attention transformers. §6 not only provides a characterization of $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ in terms of $\mathbf{LTL}[\diamondsuit]$, but also establishes its equivalence to future-masked, finite-precision *soft*-attention transformers, in conjunction with the results of Li and Cotterell's (2025) (summarized in Tab. 1). This equivalence yields a compelling interpretation of $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ as a principled abstraction of soft-attention transformers—one that is more appropriate than $\mathcal{T}_{\blacktriangleright}^{\mathsf{F}}$ due to the expressivity gap between soft attention and rightmost UHA. Consequently, this motivates further investigation of $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ as a simplified yet faithful analog of more complex soft-attention architectures.

Dot-depth hierarchy. The **dot-depth hierarchy** (Cohen and Brzozowski, 1971) classifies star-free languages based on the minimal alternation depth of *concatenation* and *Boolean operations* in the regular expressions defining them. The hierarchy is infinite and reflects increasing expressive power. Brzozowski and Ellen (1980) show that the class of languages definable in $\mathbf{LTL}[\diamondsuit]$ forms

a strict subclass of dot-depth 2, while being incomparable to dot-depth 1. In a related line of work, Bhattamishra et al. (2020) empirically find that transformers struggle to generalize to star-free languages with dot-depth greater than 1.

Until Hierarchy. An alternative (infinite) hierarchy spanning the star-free languages is that of **until hierarchy** (Etessami and Wilke, 1996; Thérien and Wilke, 2001), which stratifies the family according to the required number of $\mathbf{U}$ (or equivalently, $\mathbf{S}$) operations in an $\mathbf{LTL}$ formula required to define a language. Our and Li and Cotterell's (2025) results naturally place leftmost UHA and soft-attention transformers within the 0th layer of this hierarchy.

Abilities and Limitations. Exact characterizations of transformers allow us to derive precise conclusions about models' abilities and limitations. Our results, in particular, mean that $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ transformers, like their soft-attention counterparts, *cannot* model simple languages such as *(i)* strictly local languages and n-gram models, which have been linked to infinite-precision UHATs (Svete and Cotterell, 2024); *(ii)* locally testable languages (which require detecting contiguous substrings); *(iii)* languages of nested parentheses of bounded depth (bounded Dyck languages) which have also been linked to infinite-precision transformers (Yao et al., 2021); and *(iv)* any non-star-free languages such as PARITY. In contrast, the equivalence to $\mathbf{LTL}[\diamondsuit]$ means that $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ *can* model simple languages such as those whose string membership depends on the presence of not necessarily contiguous subsequences or on the string prefix. Li and Cotterell (2025) find strong empirical evidence that this theoretical equivalence faithfully translates into the practical performance of trained transformers on formal languages.

A duality. We finally note that *past* masking and *rightmost* UHA has a natural characterization with $\mathbf{LTL}[\diamondsuit]$, the fragment of $\mathbf{LTL}$ with only the $\diamondsuit$ operator. This duality is summarized in Tab. 1 and elaborated on in App. E.

Limitations

We limit ourselves to a purely theoretical investigation of the expressivity of a particular model of a transformer. In particular, our results hold for finite-precision transformers with leftmost hard

attention and no positional encodings. This is important to consider as the representational capacity of transformers depends on the choices made for all these components. We do not consider learnability and training, which are opening up to be an exciting area of study and promise to bring our understanding of the representational capacity of transformers closer to what we observe in practice (Hahn and Rofin, 2024). In fact, due to the discrete nature of the hard-attention mechanism, training UHATs in practice is infeasible. Nevertheless, recent work shows how temperature scaling and unbounded positional encodings can be used to simulate hard attention with soft attention (Yang et al., 2024b). We leave the empirical investigation of contrasting the performance of $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}, \mathcal{T}_{\blacktriangleright}^{\mathsf{F}}, \mathcal{T}_{\blacktriangleright}^{\mathsf{P}}$ on various language classes to future work.

Ethical considerations

We used AI-based tools (ChatGPT and GitHub Copilot) for writing assistance. We used the tools in compliance with the ACL Policy on the Use of AI Writing Assistance.

Acknowledgements

References

Pablo Barceló, Alexander Kozachinskiy, Anthony Widjaja Lin, and Vladimir Podolskii. 2024. Logical languages accepted by transformer encoders with hard attention. In *The Twelfth International Conference on Learning Representations*.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online. Association for Computational Linguistics.

Janusz Antoni Brzozowski and Faith Ellen. 1980. Languages of R-trivial monoids. *Journal of Computer and System Sciences*, 20(1):32–49.

Ashok K. Chandra, Larry Stockmeyer, and Uzi Vishkin. 1984. Constant depth reducibility. *SIAM Journal on Computing*, 13(2):423–439.

David Chiang and Peter Cholak. 2022. Overcoming a theoretical limitation of self-attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7654–7664, Dublin, Ireland. Association for Computational Linguistics.

David Chiang, Peter Cholak, and Anand Pillay. 2023. Tighter bounds on the expressivity of transformer encoders. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Rina Cohen and Janusz Antoni Brzozowski Brzozowski. 1971. Dot-depth of star-free events. *Journal of Computer and System Sciences*, 5(1):1–16.

K. Etessami and T. Wilke. 1996. An Until hierarchy for temporal logic. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 108–117.

Dov Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. 1980. On the temporal analysis of fairness. In *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '80, page 163–173, New York, NY, USA. Association for Computing Machinery.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Michael Hahn and Mark Rofin. 2024. Why are sensitive functions hard for transformers? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14973–15008, Bangkok, Thailand. Association for Computational Linguistics.

Yiding Hao, Dana Angluin, and Robert Frank. 2022. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810.

William Hesse. 2001. Division is in uniform TC0. In *Automata, Languages and Programming*, pages 104–114, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hans Kamp. 1968. *Tense Logic and the Theory of Linear Order*. Ph.D. thesis, UCLA.

Kenneth Krohn and John Rhodes. 1965. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Transactions of The American Mathematical Society - TRANS AMER MATH SOC*, 116.

Jiaoda Li and Ryan Cotterell. 2025. Characterizing the expressivity of transformer language models. *arXiv preprint arXiv:2505.23623*.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023. Exposing attention glitches with flip-flop language modeling. In

Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Robert McNaughton and Seymour Papert. 1971. *Counter-Free Automata*. M.I.T. Press research monographs. M.I.T. Press.

William Merrill and Ashish Sabharwal. 2023. A logic for expressing log-precision transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

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

Franz Nowak, Anej Svete, Alexandra Butoi, and Ryan Cotterell. 2024. On the representational capacity of neural language models with chain-of-thought reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12510–12548, Bangkok, Thailand. Association for Computational Linguistics.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. 2021. Attention is turing-complete. *Journal of Machine Learning Research*, 22(75):1–35.

Amir Pnueli. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Lena Strobl. 2023. Average-hard attention transformers are constant-depth uniform threshold circuits. *arXiv preprint arXiv:2308.03212*.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. 2024. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561.

Anej Svete and Ryan Cotterell. 2024. Transformers can represent n-gram language models. In *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. Association for Computational Linguistics.

Denis Thérien and Thomas Wilke. 2001. Temporal Logic and Semidirect Products: An Effective Characterization of the Until Hierarchy. *SIAM Journal on Computing*, 31(3):777–798.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Andy Yang, David Chiang, and Dana Angluin. 2024a. Masked hard-attention transformers recognize exactly the star-free languages. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Andy Yang, Lena Strobl, David Chiang, and Dana Angluin. 2024b. Simulating hard attention using soft attention.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. 2021. Self-attention networks can process bounded hierarchical languages. In *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. Association for Computational Linguistics.

A Related Work

Existing work has established a rich landscape of results on the expressivity of transformers.

Lower and upper bounds for UHA. Hahn (2020) show that UHA transformers with unbounded precision, left attention, and no masking can not recognize PARITY (bit strings with an odd number of ones) nor DYCK-1 (language of correctly nested parentheses of one type). Hao et al. (2022) refine this result by showing that UHA transformers with unbounded precision and left attention can recognize at most languages in AC^0, the family of circuits of constant depth, polynomial size, and unbounded fan-in. Maybe surprisingly, this suggests such transformers can not recognize even simple languages outside AC^0 such as PARITY and MAJORITY (all bit strings in which more than half of bits are 1s). Other problems not in AC^0 include sorting, integer multiplication (Chandra et al., 1984), and integer division (Hesse, 2001). Barceló et al. (2024) show that UHA transformers augmented with arbitrary positional encodings are lower bounded by an extension of $\mathbf{FO}[<]$ with all possible monadic numerical predicates (which includes all regular languages in AC^0). Yang et al. (2024a) further refine the understanding of the relationship between $\mathbf{FO}[<]$ and finite-precision transformers by proving the equivalence between the two when the transformers are equipped with strict future masking.

Average-hard attention. Average-hard attention (AHA) differs from UHA in that when confronted with several positions with equal scores, they average their values to compute the next contextual representation. It can be seen as a special case of soft-attention (Li and Cotterell, 2025). Hao et al. (2022) show AHA *unbounded* precision transformers are more expressive than UHA transformers, as they can recognize languages outside AC^0, such as PARITY and DYCK-1. Merrill et al. (2022b) show that AHA transformers with floating-point activations can be simulated in TC^0 (the extension of AC^0 with majority gates, which output 1 iff at least half of the inputs are 1), while Strobl (2023) extend this result by tightening the upper bound to L-UNIFORM TC^0 (which consists of TC^0 circuits with the additional constraint that there exists a deterministic Turing Machine that runs in logarithmic space that can describe the circuits).

Transformers and logic. Previous results relating transformers to logic include Chiang et al. (2023), who show that finite-precision softmax transformers are upper-bounded by a generalization of first-order logic with counting quantifiers and modular arithmetic over input position indices. On the other hand, they show this logic to be a lower bound on the expressivity of unbounded-precision transformers. Merrill and Sabharwal (2023) contribute by characterizing a more expressive variant of transformers—they allow precision logarithmic in the input length and show an upper bound of first-order logic extended with majority-vote quantifiers.

Equivalence of right-attention transformers to $\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$. Yang et al. (2024a) show the equivalence of $\mathcal{T}_{\blacktriangleright}^{\mathsf{F}}$ and $\mathcal{T}_{\blacktriangleleft}^{\mathsf{P}}$ to $\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$. In their constructions, the operators $\mathbf{S}$ and $\mathbf{U}$ in $\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$ can only be expressed by transformers in $\mathcal{T}_{\blacktriangleright}^{\mathsf{F}}$, $\mathcal{T}_{\blacktriangleleft}^{\mathsf{P}}$ respectively (and vice-verse, $\mathcal{T}_{\blacktriangleright}^{\mathsf{F}}$ and $\mathcal{T}_{\blacktriangleleft}^{\mathsf{P}}$ require the operators $\mathbf{S},\mathbf{U}$ respectively in their $\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$ formulation). Moreover, by Gabbay et al. (1980), the fragment of $\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$ consisting of only $\mathbf{U}$ and the Boolean connectives, denoted by $\mathbf{LTL}[\mathbf{U}]$ (or, in the analogous symmetric case, $\mathbf{LTL}[\mathbf{S}]$), is sufficient for equivalence $\mathbf{FO}[<]$ (which is equivalent to $\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$ by Kamp (1968)).

Equivalence of soft-attention transformers to $\mathbf{LTL}[\diamondsuit]$. Li and Cotterell (2025) relates $\mathbf{LTL}[\diamondsuit]$ and $\mathbf{PFO}^2[<]$ (the fragment of $\mathbf{FO}[<]$ which considers at most two distinct variables at the same time where any bound variable can only peek into the past of a free variable) to languages with $\mathcal{R}$-trivial monoids (Brzozowski and Ellen, 1980). These are described by a set of equivalent formalisms such as $\mathcal{R}$-expressions, $\mathcal{R}$-trivial monoids, and partially ordered automata. Li and Cotterell (2025) use this equivalence to show strict future-masked softmax transformers are equivalent to $\mathbf{LTL}[\diamondsuit]$. The upper bound is proven through the equivalent $\mathbf{PFO}^2[<]$: They show $\mathbf{PFO}^2[<]$ can simulate a sum of a finite number of fixed-precision floating-point numbers (and thus the weighted sum in softmax), while other transformations (computations of keys and queries, dot-products, etc.) can be computed with the standard Boolean connectives. They show the lower bound of $\mathbf{LTL}[\diamondsuit]$ by translating the standard Boolean

operations into feedforward networks, and the $\diamond$ operation with the future-masked attention mechanism. Together with Hao et al.'s (2022) results, this illuminates the complexity of transformer expressivity: While in the unbounded-precision regime, AHA transformers strictly subsume UHA transformers, in the finite-precision regime, the relationship depends on the direction of attention mechanism. The models are equivalent if leftmost tiebreaking is used and in the case of rightmost tiebreaking, UHA is strictly more expressive than AHA, the reverse of the unbounded-precision case.

B   Background

This section provides the necessary background on the formalisms used in the paper and details some concepts introduced in the main part of the paper.

B.1   Finite-State Automata

Definition B.1. *A **semiautomaton** $\mathcal{A}$ is a 3-tuple (Σ, Q, δ) where Σ is an alphabet, Q is a finite set of **states** and $\delta \colon Q \times \Sigma \to Q$ is a **transition function**.*
    *We further define an **initialized semiautomaton** as a semiautomaton with an initial state.*

Definition B.2. *A **deterministic finite automaton (DFA)** $\mathcal{A}$ is a 5-tuple $(\Sigma, Q, q_\iota, F, \delta)$ where (Σ, Q, δ) is a semiautomaton, $q_\iota \in Q$ is an initial state, and $F \subseteq Q$ is a set of final states.*

Definition B.3. *Let $\delta^* \colon Q \times \Sigma^* \to Q$ be the transitive closure of δ, defined as*

$$\delta^*(q, w) = \delta(q, w), \text{ for } w \in \Sigma \tag{12a}$$
$$\delta^*(q, w_1 \cdots w_T) = \delta(\delta^*(q, w_1 \cdots w_{T-1}), w_T) \tag{12b}$$

*with $\delta^*(q, \varepsilon) = q$ for any $q \in Q$. A **partially ordered DFA (POFA)** is a DFA $\mathcal{A} = (\Sigma, Q, q_\iota, F, \delta)$ where there is a partial order relation $\preceq$ on Q defined as $q \preceq p$ if and only if $\delta^*(q, \boldsymbol{w}) = p$ for some string $\boldsymbol{w} \in \Sigma^*$.*

Intuitively, POFAs are *acyclic* DFAs with possible self-loops, resulting in partially ordered states.

Definition B.4. *Let a DFA $\mathcal{A} = (\Sigma, Q, q_\iota, F, \delta)$ and $\mathbb{L}$ be the language it accepts. We define the **reverse automaton** $\mathcal{A}^R$ as the automaton that recognizes the reverse language $\mathbb{L}^R$ consisting of all strings in $\mathbb{L}$ but reversed.*

Definition B.5. *A DFA $\mathcal{A}$ is a **partially ordered reverse automaton (RPOFA)** if $\mathcal{A}^R$ is a POFA.*

Definition B.6. *Let $\mathcal{B}_1 = (\Sigma, Q_1, \delta_1)$ be a semiautomaton. Let $\mathcal{B}_2 = (Q_1 \times \Sigma, Q_2, \delta_2)$ be another, possibly partial,[6] semiautomaton such that for every $q_1 \in Q_1, q_2 \in Q_2$, either $\delta(q_2, \langle q_1, w \rangle)$ is defined for every $w \in \Sigma$ or undefined for every $w \in \Sigma$. The **cascade product** $\mathcal{B}_1 \circ \mathcal{B}_2$ is the semiautomaton $\mathcal{C} = (\Sigma, Q, \delta)$ such that:*

- $Q = \{(q_1, q_2) : \delta_2(q_2, \langle q_1, w \rangle) \text{ is defined}\}$

- $\delta(\langle q_1, q_2 \rangle, w) = (\delta_1(q_1, w), \delta_2(q_2, \langle q_1, w \rangle))$

Definition B.7. *For $n \geq 0$, an **n-way fork** is an initialized semiautomaton $(\Sigma, \{q_0, q_1, \cdots, q_n\}, \delta)$ where $\Sigma = \Sigma_0 \cup \Sigma_1 \cdots \cup \Sigma_n$, the Σ_i's are non-empty and pairwise disjoint, $\delta(q_0, a) = q_i$ for all $a \in \Sigma_i$, and $\delta(q_i, a) = q_i$ for all $a \in \Sigma$. A **half-reset** (Fig. 2) is a 1-way fork.*

Definition B.8. *A surjection $\phi \colon Q \to Q'$ is an **automaton homomorphism** from the semiautomata $\mathcal{A} = (\Sigma, Q, \delta)$ to $\mathcal{A}' = (\Sigma', Q', \delta')$ if for every $q \in Q, w \in \Sigma$:*

$$\phi(\delta(q, w)) = \delta'(\phi(q), w) \tag{13}$$

In this case, we say $\mathcal{A}'$ is homomorphic to $\mathcal{A}$, or that $\mathcal{A}'$ is the homomorphic image of $\mathcal{A}$.

---

[6]A partial automaton is one in which some symbol–state combinations could lead to an undefined transition.
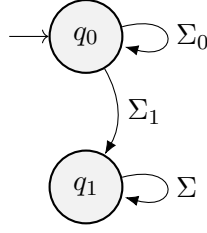
Figure 2: A 1-way fork

B.2 Syntactic Monoids

Definition B.9. *A **monoid** $\mathbb{M}$ is a set equipped with a binary operation and an identity element.*

For instance, the **free monoid** is the set Σ^* equipped with the concatenation operation and the empty string ε as identity.

Definition B.10. *A monoid $\mathbb{M}$ is $\mathcal{R}$-trivial if for all $w_1, w_2, w_3 \in \mathbb{M}$, $w_1 w_2 w_3 = w_1$ implies $w_1 w_2 = w_1$. A monoid $\mathbb{M}$ is $\mathcal{L}$-trivial if for all $w_1, w_2, w_3 \in \mathbb{M}$, $w_3 w_2 w_1 = w_1$ implies $w_2 w_1 = w_1$.*

More details about $\mathcal{R}$-trivial monoids can be found in Brzozowski and Ellen (1980).

Definition B.11. *The **syntactic congruence** $\preceq_{\mathbb{L}}$ is the equivalence relation on Σ^* given the language $\mathbb{L}$ such that for all $\boldsymbol{x}, \boldsymbol{y} \in \Sigma^*$, we have $\boldsymbol{x} \preceq_{\mathbb{L}} \boldsymbol{y}$ if and only if:*

$$\boldsymbol{s}\boldsymbol{x}\boldsymbol{z} \in \mathbb{L} \iff \boldsymbol{s}\boldsymbol{y}\boldsymbol{z} \in \mathbb{L} \; \forall \boldsymbol{s}, \boldsymbol{z} \in \Sigma^* \tag{14}$$

Definition B.12. *The quotient monoid $\Sigma^*/\preceq_{\mathbb{L}}$ is the **syntactic monoid** of $\mathbb{L}$.*

B.3 Regular Expressions

A **regular language** can be described by **regular expressions**, which are elements of the closure of $\emptyset, \varepsilon$, and all $w \in \Sigma$ under concatenation, concatenation, and Kleene star. A regular language is **star-free** if the regular expression that describes the language does not require the Kleene star operator.

Definition B.13. *An $\mathcal{R}$-expression is a finite union of regular expressions of the form $\Sigma_0^* w_1 \Sigma_1^* \cdots w_n \Sigma_n^*$ where $w_t \in \Sigma$, $\Sigma_t \subseteq \Sigma$ and $w_t \notin \Sigma_{t-1}^*$ for $1 \le t \le n$. An $\mathcal{L}$-expression is defined analogously with the dual constraint of $w_t \notin \Sigma_t^*$.*

For instance, $a\Sigma^*$ is an $\mathcal{R}$-expression, while $\Sigma^* a$ is an $\mathcal{L}$-expression.

B.4 Linear Temporal Logic

We now present all possible semantics in $\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$. The semantics are defined inductively:
- $\boldsymbol{w}, t \models \pi_a \iff w_t = a$;
- $\boldsymbol{w}, t \models \psi_1 \vee \psi_2 \iff \boldsymbol{w}, t \models \psi_1 \vee \boldsymbol{w}, t \models \psi_2$;
- $\boldsymbol{w}, t \models \psi_1 \wedge \psi_2 \iff \boldsymbol{w}, t \models \psi_1 \wedge \boldsymbol{w}, t \models \psi_2$;
- $\boldsymbol{w}, t \models \neg \psi \iff \boldsymbol{w}, t \not\models \psi$;
- $\boldsymbol{w}, t \models \diamondsuit\psi \iff \exists t' < t : \boldsymbol{w}, t' \models \psi$;
- $\boldsymbol{w}, t \models \diamondsuit\psi \iff \exists t' > t : \boldsymbol{w}, t' \models \psi$;
- $\boldsymbol{w}, t \models \psi_1 \mathbf{S} \psi_2 \iff \exists t' < t : \boldsymbol{w}, t' \models \psi_2$ and $\boldsymbol{w}, k \models \psi_1$ for all k with $t' < k < t$;
- $\boldsymbol{w}, t \models \psi_1 \mathbf{U} \psi_2 \iff \exists t' > t : \boldsymbol{w}, t' \models \psi_2$ and $\boldsymbol{w}, k \models \psi_1$ for all k with $t < k < t'$.

$\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$ defines exactly the class of star-free languages (Kamp, 1968; McNaughton and Papert, 1971). $\mathbf{LTL}[\diamondsuit]$ and $\mathbf{LTL}[\diamondsuit]$, the fragments of $\mathbf{LTL}$ with only the $\diamondsuit$ and $\diamondsuit$ operators, respectively, are strictly less expressive than $\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$ (neither can recognize $a\Sigma^* b$, which can be recognized by $\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$), and have characterizations in terms of monoids, automata, and first-order logic (App. B.6).

986

B.5 First-Order Logic

First-order logic with the $<$ relation, denoted by $\mathbf{FO}[<]$, extends the usual propositional logic with predicates and quantifiers. We consider free variables $x, y, z, \cdots$ that represent positions over a string $\boldsymbol{w}$ of size T. $\mathbf{FO}[<]$ includes unary predicates π_w for $w \in \Sigma$, where $\pi_w(x) = \text{TRUE} \iff$ there is a w at position x. As with $\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$, we can inductively define formulas from π_w using the standard Boolean operators, the binary predicate $<$ between positions, and the existential quantifier $\exists$. By Kamp (1968), $\mathbf{FO}[<]$ is equivalent to $\mathbf{LTL}[\diamondsuit, \diamondsuit, \mathbf{S}, \mathbf{U}]$ and by McNaughton and Papert (1971), it is equivalent to the star-free languages.

$\mathbf{FO}^2[<]$ is the fragment of $\mathbf{FO}[<]$ that can only consider two distinct variables at the same time. Li and Cotterell (2025) further define $\mathbf{PFO}^2[<]$ as the past fragment of $\mathbf{FO}^2[<]$ that can only "peek into the past." Namely, any single-variable formula $\phi(x)$ can only have bounded existential quantifiers of the form "$\exists y < x$". We analogously have the future fragment of $\mathbf{FO}^2[<]$, $\mathbf{FFO}^2[<]$, where we only allow peeking into the future with quantifiers of the form "$\exists y > x$".

B.6 Equivalence Between Formalisms

Due to Brzozowski and Ellen (1980) and Li and Cotterell (2025), we have the two dual theorems:

Theorem B.1. *Let $\mathbb{L} \subseteq \Sigma^*$ be a regular language, $\mathbb{M}$ be its syntactic monoid, and $\mathcal{A}$ be the minimal DFA accepting it. The following conditions are equivalent:*

(i) $\mathbb{M}$ is $\mathcal{R}$-trivial,
(ii) $\mathbb{L}$ can be denoted by an $\mathcal{R}$-expression,
(iii) $\mathcal{A}$ is a POFA,
(iv) $\mathcal{A}$ is the homomorphic image of a cascade product of half-resets,
(v) $\mathbb{L}$ can be recognized by a formula in $\mathbf{PFO}^2[<]$,
(vi) $\mathbb{L}$ can be recognized by a formula in $\mathbf{LTL}[\diamondsuit]$.

Theorem B.2. *Let $\mathbb{L} \subseteq \Sigma^*$ be a regular language, $\mathbb{M}$ be its syntactic monoid, and $\mathcal{A}$ be the minimal DFA accepting it. The following conditions are equivalent:*

(i) $\mathbb{M}$ is $\mathcal{L}$-trivial,
(ii) $\mathbb{L}$ can be denoted by an $\mathcal{L}$-expression,
(iii) $\mathcal{A}$ is a RPOFA (equivalently, $\mathcal{A}^R$ is a POFA),
(iv) $\mathcal{A}^R$ is the homomorphic image of a cascade product of half-resets,
(v) $\mathbb{L}$ can be recognized by a formula in $\mathbf{FFO}^2[<]$,
(vi) $\mathbb{L}$ can be recognized by a formula in $\mathbf{LTL}[\diamondsuit]$.

C Proofs of $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ and $\mathbf{LTL}[\diamondsuit]$ Equivalence

The equivalence between $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ and $\mathbf{B\text{-}RASP}_{\blacktriangleleft}^{\mathsf{F}}$ follows directly from Theorems 3 and 4 of Yang et al. (2024a).

Theorem 5.1. *For any UHAT in $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$, there exists an equivalent $\mathbf{B\text{-}RASP}_{\blacktriangleleft}^{\mathsf{F}}$ program. For any $\mathbf{B\text{-}RASP}_{\blacktriangleleft}^{\mathsf{F}}$ program, there exists an equivalent UHAT in $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$.*

Proof. Yang et al. (2024a, Thms. 3 and 4) and the supporting lemmata (Lemmata 21 and 24) treat $\blacktriangleleft$ and $\blacktriangleright$ separately. Translations constrained to $\blacktriangleleft$ are thus special cases of their proofs. ∎

The following proofs largely follow the proofs of Lemmata 13 and 14 of Yang et al. (2024a). We highlight the part that distinguishes $\blacktriangleleft$ and $\blacktriangleright$ in red.

Theorem 6.1. *For any formula ψ of $\mathbf{LTL}[\diamondsuit]$, there is a $\mathbf{B\text{-}RASP}_{\blacktriangleleft}^{\mathsf{F}}$ program with a Boolean vector P_ψ such that, for any input $\boldsymbol{w}$ of length T and all $t \in [T]$, we have $\boldsymbol{w}, t \models \psi \iff P_\psi(t) = 1$.*

Proof. We proceed by induction.

Base case. The atomic formulas π_a can be represented by initial Boolean vectors Q_a for $a \in \Sigma$.

Inductive step. Assume ψ_1 and ψ_2 can be converted to **B-RASP$_\blacktriangleleft^\mathsf{F}$** vectors P_{ψ_1} and P_{ψ_2}, respectively. We distinguish three cases of building a new formula from ψ_1 and ψ_2:

• Case *(1)*: $\psi = \neg\psi_1$. Add a position-wise operation:

$$P_\psi(t) = \neg P_{\psi_1}(t). \tag{15}$$

• Case *(2)*: $\psi = \psi_1 \wedge \psi_2$. Add a position-wise operation:

$$P_\psi(t) = P_{\psi_1}(t) \wedge P_{\psi_2}(t). \tag{16}$$

• Case *(3)*: $\psi = \Diamond\psi_1$. Add an attention operation with future masking and $\blacktriangleleft$ tiebreaking:

$$P_\psi(t) = \blacktriangleleft_{t'}[t' < t, P_{\psi_1}(t')] \; P_{\psi_1}(t') : 0. \tag{17}$$

∎

Theorem 6.2. *For any Boolean vector P of a **B-RASP$_\blacktriangleleft^\mathsf{F}$** program $\mathcal{P}$, there is a formula ψ_P of **LTL**$[\Diamond]$ such that for any input $\boldsymbol{w}$ of length T and all $t \in [T]$, we have $P(t) = 1 \iff \boldsymbol{w}, t \models \psi_P$.*

Proof. We proceed by induction.

Base case. Each atomic vector $Q_a(t)$ can be translated to the atomic formula π_a.

Induction step. Assume vectors $P_1, \ldots, P_{i-1}$ can be translated to $\psi_{P_1}, \ldots, \psi_{P_{i-1}}$. We distinguish different options of building a new vector out of $P_1, \ldots, P_{i-1}$:

• Case *(1)*: $P_i(t)$ is a position-wise operation:

$$P_i(t) = f(P_1(t), \ldots, P_{i-1}(t)), \tag{18}$$

where f is a Boolean function. We can translate $P_i(t)$ into $\psi_i = f(\psi_1, \ldots, \psi_{i-1})$.

• Case *(2)*: $P_i(t)$ is an attention operation that uses leftmost tiebreaking and future masking, that is,

$$P_i(t) = \blacktriangleleft_{t'}[t' < t, s(t')] \; v(t') : d(t). \tag{19}$$

By the inductive hypothesis, there are **LTL**$[\Diamond]$ formulas ψ_S, ψ_V, and ψ_D corresponding to s, v, and d. We can thus write $P_i(i)$ as:

$$\psi_i = (\Diamond(\psi_S \wedge \neg\Diamond\psi_S \wedge \psi_V)) \vee (\neg(\Diamond\psi_S) \wedge \psi_D). \tag{20}$$

where $\psi_S \wedge \neg\Diamond\psi_S$ identifies the leftmost position that satisfies ψ_S in the string.

• Case *(3)*: We now treat the attention operation with rightmost tiebreaking and future masking, i.e.,

$$P_i(t) = \blacktriangleright_{t'}[t' < t, s(t')] \; v(t') : d(t). \tag{21}$$

In this case, we need to find the rightmost position satisfying ψ_S not in the *entire string* $\boldsymbol{w}$, but before the *current position*, which can only be realized using the **S** operator:

$$\psi_i = (\neg\psi_S \mathbf{S}(\psi_S \wedge \psi_V)) \vee (\neg(\Diamond\psi_S) \wedge \psi_D). \tag{22}$$

∎

C.1 POFAs as $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$

The Krohn–Rhodes theorem (Krohn and Rhodes, 1965) states that any deterministic automaton is the homomorphic image of a cascade of simple automata whose transitions induce resets or permutations of the states. Thm. B.1 by Brzozowski and Ellen (1980) provides an analogous decomposition for POFAs, namely that they are homomorphic to a cascade of *half-resets*. In a very similar manner to Yang et al. (2024a)'s Thm. 15, we can express this decomposition in a **B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$** program, which can further be expressed by a transformer in $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$.

We first formalize how **B-RASP** can compute sequence-to-sequence functions $\Sigma^* \to \Gamma^*$ if for instance, we want a **B-RASP** program to simulate an automaton (i.e., output the same states traversed by the automaton when ran on the same input). We designate a set of output vectors Y_γ, for all $\gamma \in \Gamma$, and the sequence-to-sequence function then maps t to γ if $Y_\gamma(t)$ is true.

We then say a **B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$** program $\mathcal{P}$ simulates an initialized semiautomaton $\mathcal{A} = (\Sigma, Q, \delta)$ with initial state q_s iff for every input string $\boldsymbol{w}$, the output vectors of $\mathcal{P}$ encode the sequence of states traversed by $\mathcal{A}$ when ran on $\boldsymbol{w}$ with initial state q_s.

Lemma C.1. *Let $\mathcal{H} = (\Sigma_0 \cup \Sigma_1, \{q_0, q_1\}, q_0, \delta)$ be a half-reset. Then there exists a **B-RASP** program $\mathcal{P}_\mathcal{H}$ that simulates $\mathcal{H}$ started in start state q_0.*

Proof. We first define two **B-RASP** programs $\mathcal{B}_0$, $\mathcal{B}_1$, which return 1 if and only if the half-reset is in state q_0 or q_1, respectively. Namely:

$$\mathcal{B}_0(t) = \blacktriangleleft_{t'}[t' < t, \bigvee_{w \in \Sigma_1} Q_w(t')] \ 0 : 1$$
$$\mathcal{B}_1(t) = 1 - \mathcal{B}_0(t) \tag{23}$$

In half-resets, the presence or absence at any instance of symbols in Σ_1 determines whether we are in q_0 or q_1. It suffices to check if at the current index t, there has been a past $w \in \Sigma_1$, which can be done by computing $\bigvee_{w \in \Sigma_1} Q_w(t')$ for all indices t' satisfying future masking. ∎

Lemma C.2. *Let $\mathcal{A} = (\Sigma, Q_1, \delta_1)$ be a semiautomaton that can be simulated from state $\boldsymbol{s}_1$ by a **B-RASP** program $\mathcal{P}_\mathcal{A}$. Let $\mathcal{H} = (Q_1 \times \Sigma, Q_2, \delta_2)$ be a half-reset and let $\mathcal{C} = \mathcal{A} \circ \mathcal{H}$. Then, there exists a program $\mathcal{P}_\mathcal{C}$ that simulates $\mathcal{C}$ started in state $(\boldsymbol{s}_1, \boldsymbol{s}_2)$ for an arbitrary $\boldsymbol{s}_2 \in Q_2$.*

Proof. We now use for all $q \in Q_1$, the predicates $\mathcal{B}_{\mathcal{A},q}$ that denote whether $\mathcal{A}$ is in some state $q \in Q_1$ at time t when started at $\boldsymbol{s}_1$. By the assumption that $\mathcal{A}$ can be simulated by a **B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$** program, we have access to such predicates.

We now define predicates for $q \in Q_1, w \in \Sigma$:

$$Q'_{(q,w)}(t) = \mathcal{B}_{\mathcal{A},q}(t) \wedge Q_w(t) \tag{24}$$

These formulas encode the presence of an element in $Q_1 \times \Sigma$ (a state in the first semiautomaton $\mathcal{A}$ and an alphabet symbol) for the half-reset $\mathcal{H}$.

As $\mathcal{H}$ is a half-reset, we can classify every tuple $(q, w) \in Q_1 \times \Sigma$ into one of two sets Σ_0 or Σ_1 (depending on if they reset to q_0 or q_1 in $\mathcal{H}$). As in Lem. C.1, we can define predicates $\mathcal{B}_{\mathcal{H},q}(t)$ using the predicates $Q'_{(q,w)}$ and the known classification of elements in $Q_1 \times \Sigma$ into some state q_0 or q_1.

To finally simulate the cascade product $\mathcal{C} = \mathcal{A} \circ \mathcal{H}$, we define predicates that compute for every state (q, p), $q \in Q_1, p \in Q_2$, whether $\mathcal{C}$ is in it:

$$\mathcal{C}_{(q,p)}(t) = \mathcal{B}_{\mathcal{A},q}(t) \wedge \mathcal{B}_{\mathcal{H},p}(t) \tag{25}$$

∎

Theorem C.1. *Let $\mathcal{A}$ be a POFA. Then, there exists an equivalent transformer $\mathsf{T} \in \mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$.*

Proof. Let $\mathcal{C} = \mathcal{B}_0 \circ \cdots \circ \mathcal{B}_k$ the semiautomaton $\mathcal{A}$ is homomorphic to. Let $\phi \colon Q' \to Q$ be the homomorphism from $\mathcal{C}$ to $\mathcal{A}$ (where Q are the states of $\mathcal{A}$, Q' are the states of $\mathcal{C}$). By Lem. C.2, we can iteratively define formulas $\mathcal{C}_{q'}$ for all $q' \in Q'$ that simulate the semiautomaton $\mathcal{C}$. If we write instructions describing the homomorphism ϕ, we can then write formulas that yield the states traversed by $\mathcal{A}$ as:

$$\mathcal{A}_q(t) = \bigvee_{p \in Q', \phi(p)=q} \mathcal{C}_p(t) \tag{26}$$

$\mathcal{A}_q(t)$, however, describes the state *before reading the symbol at t* (by strict masking). We want predicates that yield the state *after reading at the symbol position t*. We thus write:

$$Y_q(t) = \bigvee_{\substack{p \in Q, w \in \Sigma \\ \delta(p,w)=q}} \mathcal{A}_p(t) \wedge Q_w(t) \tag{27}$$

Finally, we denote by F the set of final states in $\mathcal{A}$. We thus define the output vector Y by:

$$Y(t) = \bigvee_{q \in F} Y_q(t) \tag{28}$$

$Y(T) = 1$ if and only if $\mathcal{A}$ is in one of the final states when reading the entire string. This concludes the translation of $\mathcal{A}$ to **B-RASP$_{\blacktriangleleft}^{\mathsf{F}}$**, showing the existence of a $\mathsf{T} \in \mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ equivalent to $\mathcal{A}$. ∎

D Direct Translations of $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$

D.1 A Normal Form for $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ Transformers

The representational capacity of transformers depends tightly on the modeling assumptions. §6 studies the same architecture as Yang et al. (2024a). In this section, we outline a formalization of $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ transformers in a form that allows us to describe direct translations to **LTL**$[\diamondsuit]$ and POFAs in §7 more easily. The idea of this is similar to the normal form of Hao et al. (2022).

D.1.1 Finite Precision and Simplifications

As Yang et al. (2024a), we work with finite-precision transformers with no positional encodings. Further, we focus on strict future masking and leftmost hard attention, defined formally below.

In our formalization, we omit element-wise transformations such as the query, key, and value transformations and element-wise MLPs. Instead, we directly aggregate the original representations into (constant-size) vectors that can take increasingly many values as the number of layers increases.[7] The intuition and motivation for this stems from the use of hard attention: At any layer, hard attention augments the current contextual representation with the representation of one (preceding) symbol—the hardmax. Thus, all information that the transformer can capture is already captured if the contextual representations keep around the identities of the elements that were returned by hardmax. The element-wise representations, in this case, do not provide any additional representational power. We elaborate on this in App. D.1.3.

Note that we do not use layer normalization in our transformers and focus on a single-head attention mechanism. Both can be incorporated into our proofs analogously to the solutions presented in Yang et al. (2024a) (see Yang et al. (2024a, §4.1) for a discussion of multi-head attention and Yang et al. (2024a, §4.3) for layer normalization).

D.1.2 The Attention Mechanism

The central component of a transformer is the **transformer layer**.

Definition D.1. *A **transformer layer** $\mathcal{L} \colon \left(\mathbb{R}^D\right)^+ \to \left(\mathbb{R}^D\right)^+$ is a length-preserving function defined as*

$$\mathcal{L}((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)) \overset{\text{def}}{=} (\boldsymbol{y}_t)_{t=1}^T \tag{29a}$$

$$(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T) \overset{\text{def}}{=} \text{att}((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)) + (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$$

---

[7]This is formalized in Lem. D.1.

We will use $\mathcal{L}_t \overset{\text{def}}{=} \mathrm{proj}_t \circ \mathcal{L}$ for the function that computes $\mathcal{L}((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T))$ and extracts the contextual representation of the tth symbol by projecting out that dimension.

The **attention mechanism** att is specified with the following components:
- A **scoring function** score: $\mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$.
- A **masking function** $m\colon \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ that determines the positions attended to. We use future and past masking as in **B-RASP**. We write $\mathcal{M}(t) \overset{\text{def}}{=} \{t' \mid m(t, t') = 1\}$.
- A **normalization function** norm: $\mathbb{R}^T \to \boldsymbol{\Delta}^{T-1}$ that normalizes the attention scores.

We then define the attention mechanism as:

$$\mathrm{att}((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)) \overset{\text{def}}{=} (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T) \tag{30a}$$

$$\boldsymbol{y}_t \overset{\text{def}}{=} \sum_{t' \in \mathcal{M}(t)} s_{t'} \boldsymbol{x}_{t'} \tag{30b}$$

$$\boldsymbol{s} \overset{\text{def}}{=} \mathrm{norm}((\mathrm{score}(\boldsymbol{x}_{t'}, \boldsymbol{x}_t))_{t' \in \mathcal{M}(t)}) \tag{30c}$$

The $\sum$-notation in Eq. (30b) can naturally be thought of as *collecting* information from the unmasked positions. Intuitively, if the space of contextual representations is large enough, this can be interpreted as *concatenating* the representations together. This will be particularly useful in our UHA formulation, where only a single $\boldsymbol{x}_{t'}$ will be selected and the finiteness of the representation space will mean that *all* relevant information about the string $\boldsymbol{w}_{\leq t}$ will be stored in $\boldsymbol{x}_t$. See also App. D.1.3.

Let $L \in \mathbb{N}_{\geq 1}$ and $\mathcal{L}_1, \ldots, \mathcal{L}_L$ be transformer layers. A **transformer** $\mathsf{T}\colon \Sigma^* \to (\mathbb{R}^D)^+$ is a composition of transformer layers:

$$\mathsf{T} \overset{\text{def}}{=} \mathcal{L}_L \circ \cdots \circ \mathcal{L}_1 \circ \mathrm{embed} \tag{31}$$

where $\mathrm{embed}\colon \Sigma^* \to (\mathbb{R}^D)^+$ is a position-wise embedding function that maps symbols to their static representations.

We also write

$$(\boldsymbol{x}_1^{(\ell)}, \ldots, \boldsymbol{x}_T^{(\ell)}) = (\mathcal{L}_\ell \circ \cdots \circ \mathcal{L}_1 \circ \mathrm{embed})(\boldsymbol{w}) \tag{32}$$

for some layer index $\ell \in [L]$ and string $\boldsymbol{w} = w_1 \cdots w_T \in \Sigma^*$. We call $\boldsymbol{x}_t^{(\ell)}$ the **contextual representation** of symbol w_t at layer ℓ.

A transformer computes the contextual representations of the string $\boldsymbol{w} = w_1 \cdots w_T \mathrm{EOS}$ as

$$(\boldsymbol{x}_1^{(L)}, \ldots, \boldsymbol{x}_T^{(L)}, \boldsymbol{x}_{\mathrm{EOS}}^{(L)}) \overset{\text{def}}{=} \mathsf{T}(\boldsymbol{w}). \tag{33}$$

We take $\boldsymbol{x}_{\mathrm{EOS}}^{(\ell)}$ to be the representation of the entire string. This motivates the definition of the **transformer encoding function** h:

$$\boldsymbol{h}(\boldsymbol{w}) \overset{\text{def}}{=} \boldsymbol{x}_{\mathrm{EOS}}^{(L)}. \tag{34}$$

This allows us to define a transformer's language. This is usually defined based on a linear classifier based on $\boldsymbol{h}(\boldsymbol{w})$;

$$\mathbb{L}(\mathsf{T}) \overset{\text{def}}{=} \{\boldsymbol{w} \in \Sigma^* \mid \boldsymbol{\theta}^\top \boldsymbol{h}(\boldsymbol{w}) > 0\} \tag{35}$$

for some $\boldsymbol{\theta} \in \mathbb{R}^D$. Since we are working with finite-precision transformers, the set of possible $\boldsymbol{h}(\boldsymbol{w})$ is finite (in our normal form, it is a subset of Σ^{2^L}). We can thus equate the condition $\boldsymbol{\theta}^\top \boldsymbol{h}(\boldsymbol{w}) > 0$ with $\boldsymbol{h}(\boldsymbol{w})$'s inclusion in a subset of Σ^{2^L} and define the language of a transformer as follows.

Definition D.2. *Let* $\mathsf{T} \in \mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$. *We define its language* $\mathbb{L}(\mathsf{T})$ *as*

$$\mathbb{L}(\mathsf{T}) \overset{\text{def}}{=} \{\boldsymbol{w} \in \Sigma^* \mid \boldsymbol{h}(\boldsymbol{w}) \in \mathcal{F}_{\mathsf{T}}\} \tag{36}$$

where $\mathcal{F}_{\mathsf{T}}$ *is a set of accepting final representations.*

991

D.1.3 Unique Hard Attention

Let $\mathcal{C} \in \{\min, \max\}$ and let $\boldsymbol{s} \in \mathbb{R}^N$. We define

$$\mathrm{hardmax}(\boldsymbol{s})_n \stackrel{\text{def}}{=} \begin{cases} 1 & \textbf{if } n = \mathcal{C}(\mathrm{argmax}(\boldsymbol{s})) \\ 0 & \textbf{otherwise} \end{cases} \tag{37}$$

Here, $\mathrm{argmax}\,\boldsymbol{s}$ denotes the set of indices attaining the maximum value in the vector $\boldsymbol{s}$. The function $\mathcal{C}$ selects a unique index from this set: $\mathcal{C} = \max$ corresponds to **rightmost tiebreaking** ▶ and $\mathcal{C} = \min$ corresponds to **leftmost tiebreaking** ◀.

Definition D.3. *Unique hard attention is computed with the* $\mathrm{hardmax}$ *projection function:*

$$\mathrm{hardmax}(\boldsymbol{s}) = \mathrm{hardmax}(\boldsymbol{s}) \tag{38}$$

With some abuse of notation, we sometimes write $\mathrm{hardmax}(\boldsymbol{s})$ for the *position* $\mathcal{C}(\mathrm{argmax}(\boldsymbol{s}))$.

We denote future or past masking with F or P, respectively. $\mathcal{T}_{◀}^{\mathsf{F}}$, for example, denotes the class of transformers with future masking and leftmost attention.

The following lemma is a restatement of Yang et al. (2024a, Lem. 22).

Lemma D.1. *Let* T *be a UHA transformer over* Σ*. Denote with* $\boldsymbol{x}_t^{(\ell)}$ *the contextual representation of the symbol* w_t *at layer* ℓ*. The following holds:*

$$|\{\boldsymbol{x}_t^{(\ell)} \mid \boldsymbol{w} \in \Sigma^*, t \in [|\boldsymbol{w}|]\}| \leq |\Sigma|^{2^\ell}. \tag{39}$$

Proof. We prove the statement by induction on the number of layers.

Base case: $\ell = 1$. In the first layer, as we have static representations for symbols, the embedding at some position t is uniquely determined by the symbol w_t at that position. We thus have exactly $|\Sigma|$ possible representations for a given position, regardless of the length of the string.

Inductive step: $\ell > 1$. Suppose that the invariance holds for $\ell - 1$: $|\{\boldsymbol{x}_t^{(\ell-1)} \mid \boldsymbol{w} \in \Sigma^*, t \in [|\boldsymbol{w}|]\}| \leq |\Sigma|^{2^{\ell-1}}$. For any position t in the string, at layer ℓ, we will compute $\boldsymbol{x}_t^{(\ell)} = \mathtt{att}((\boldsymbol{x}_1, \dots, \boldsymbol{x}_T)) + (\boldsymbol{x}_1, \dots, \boldsymbol{x}_T)$ where $\boldsymbol{x}_t^{(\ell-1)}$ is the representation of the symbol at the previous layer $\ell - 1$. By the induction hypothesis, the element $\boldsymbol{x}_t^{(\ell-1)}$ takes one out of at most $|\Sigma|^{2^{\ell-1}}$ possible values. Moreover, the attention mechanism selects one element from the previous layer $\ell - 1$, which holds one out of at most $|\Sigma|^{2^{\ell-1}}$ possibles values by the induction hypothesis. The element $\boldsymbol{a}_t$ holds thus one out of $|\Sigma|^{2^{\ell-1}} \times |\Sigma|^{2^{\ell-1}} = |\Sigma|^{2^\ell}$ representations, concluding the induction step and the proof. ∎

Contextual representations as elements of a finite set. Lem. D.1 allows us to simplify notation: Any representation of a symbol at layer ℓ is uniquely identified by 2^ℓ symbols, i.e., $\boldsymbol{x}^{(\ell)} \in \Sigma^{2^\ell}$. We think of this as the collection of representations attended to at each of layer $\ell' < \ell$, since each selects a single position to be added to the current representation. We will thus refer to $\boldsymbol{x}^{(\ell)}$ as elements of Σ^{2^ℓ}.

D.1.4 An Invariance

In this section and in App. D.2, we use Σ for the alphabet of input symbols and Ξ for a general alphabet (finite set) of relevance. Later, Ξ will correspond to sets of the form Σ^{2^ℓ} for some $\ell \in \mathbb{N}$.

Definition D.4. *Let* Ξ *be an alphabet and* $\boldsymbol{w} \in \Xi^*$*. We define the* **symbol order** $\omega(\boldsymbol{w})$ *of* $\boldsymbol{w}$ *as the string obtained from* $\boldsymbol{w}$ *by keeping only the first occurrence of each symbol.*

We define the following relation on Ξ^*:

$$\boldsymbol{w} \simeq_\omega \boldsymbol{w}' \iff \omega(\boldsymbol{w}) = \omega(\boldsymbol{w}'). \tag{40}$$

It is not hard to verify that $\simeq_\omega$ is an equivalence relation on Ξ^* and to verify that $|\Xi^*/\simeq_\omega| = |\mathrm{OS}(\Xi)|$, where $\mathrm{OS}(\Xi)$ is the (finite) set of all ordered subsets of Ξ. We denote the equivalence class of $\boldsymbol{w} \in \Xi^*$ by $[\boldsymbol{w}] \in \Xi^*/\simeq_\omega$. We have the following important invariance.

Lemma D.2 (Attention invariance). *Let $\mathcal{L}$ be an $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ transformer layer over Ξ. For any $\boldsymbol{w}, \boldsymbol{w}' \in \Xi^*$, if $\boldsymbol{w} \simeq_{\omega} \boldsymbol{w}'$, then $\mathcal{L}_{|\boldsymbol{w}|}(\boldsymbol{w}) = \mathcal{L}_{|\boldsymbol{w}'|}(\boldsymbol{w}').$*

Proof. This follows directly from the definition of leftmost hard attention: Additional occurrences of symbols to the right of the first occurrence do not change the position attended to, meaning that the output at the final symbol is the same. ∎

D.2 $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ as **LTL**$[\diamondsuit]$

We view a transformer layer as a function that takes a contextual representation $\boldsymbol{x}_t^{(\ell)}$ and returns a function that takes in an ordered set of contextual representations $\omega(\boldsymbol{w})$ with $\boldsymbol{w} \in \Xi^*$ and returns the representation chosen by the unique hard attention mechanism.

$$\mathcal{L}^{(\ell)} \colon \Xi \to \mathrm{Map}(\mathrm{OS}(\Xi), \Xi), \tag{41a}$$

$$\mathcal{L}^{(\ell)} \colon \boldsymbol{x} \mapsto \boldsymbol{\lambda_x}. \tag{41b}$$

$\boldsymbol{\lambda_x}$ is the function that takes an ordered set of contextual representations $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ and returns the representation chosen by UHA:[8]

$$\boldsymbol{\lambda_x} \colon \mathrm{OS}(\Xi) \to \Xi, \tag{42a}$$

$$\boldsymbol{\lambda_x} \colon \mathcal{X} \mapsto \boldsymbol{x}_{\mathrm{hardmax}((\mathrm{score}(\boldsymbol{x}_t, \boldsymbol{x}_{t'}))_{\boldsymbol{x}_{t'} \in \mathcal{X}})}. \tag{42b}$$

We also define

$$\boldsymbol{x}'' \preceq_{\boldsymbol{x}} \boldsymbol{x}' \iff \mathrm{score}(\boldsymbol{x}, \boldsymbol{x}'') \leq \mathrm{score}(\boldsymbol{x}, \boldsymbol{x}') \tag{43a}$$

$$\boldsymbol{x}'' \simeq_{\boldsymbol{x}} \boldsymbol{x}' \iff \mathrm{score}(\boldsymbol{x}, \boldsymbol{x}'') = \mathrm{score}(\boldsymbol{x}, \boldsymbol{x}') \tag{43b}$$

$$\boldsymbol{x}'' \prec_{\boldsymbol{x}} \boldsymbol{x}' \iff \boldsymbol{x}'' \preceq_{\boldsymbol{x}} \boldsymbol{x}' \text{ and } \neg (\boldsymbol{x}'' \simeq_{\boldsymbol{x}} \boldsymbol{x}'). \tag{43c}$$

Theorem 7.1. *Let $\mathsf{T} \in \mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ be a transformer. Then, there exists an equivalent formula $\psi \in \mathbf{LTL}[\diamondsuit]$.*

Proof. We define an **LTL**$[\diamondsuit]$ formula representing $\mathsf{T} \in \mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ with the attention mechanism implemented by the function $\boldsymbol{\lambda_x}$ (cf. Eq. (42b)) by specifying a set of formulas representing each layer, starting with the initial one, and continuing inductively. At a high level, we define the formulas $\phi_{\boldsymbol{x} \leftarrow \boldsymbol{y}}^{(\ell)}$ to mean that the contextual representation $\boldsymbol{y}$ is added to the contextual representation $\boldsymbol{x}$ at layer ℓ, i.e., that $\boldsymbol{y}$ is the maximizer of the query-key score when $\boldsymbol{x}$ is the query: $\boldsymbol{\lambda_x}(\mathcal{X}) = \boldsymbol{y}$ for $\mathcal{X}$ representing the current string. We construct the formulas layer by layer.

Base case: $\ell = 1$. We begin by specifying first-layer formulas, which work over Σ. We define for $a, b \in \Sigma$:

$$\phi_{a \leftarrow b}^{(1)} \stackrel{\mathrm{def}}{=} \pi_a \wedge \underbrace{\bigwedge_{\substack{w \in \Sigma: \\ b \prec_a w}} \neg \diamondsuit \pi_w}_{\substack{\text{No previous symbols with} \\ \text{higher scores than b}}} \wedge \diamondsuit \Big(\pi_b \wedge \underbrace{\bigwedge_{w \in \Sigma \setminus \{b\}: b \simeq_a w} \neg \diamondsuit \pi_w}_{\substack{\text{There is } b \text{ in the past with no} \\ \text{equally-scored symbols in its past}}} \Big). \tag{44}$$

In words, b's value is added to a's static embedding if *(i)* there are no symbols in the past of a that have a higher score than b and *(ii)* there exists a position with b in the past such that there are no symbols with equal scores to its left (leftmost tiebreaking).

---

[8]We omit the dependence of $\boldsymbol{\lambda}$ on the layer index for readability.

Inductive step: $\ell > 1$. Let now $\mathcal{L}$ be the transformer layer at layer $\ell > 1$ and assume we have correctly constructed $\phi_{\boldsymbol{x} \leftarrow \boldsymbol{y}}^{(\ell')}$ for $\ell' < \ell$.

Firstly, we define the formulas $\pi_{\boldsymbol{x}}^{(\ell)}$ that, analogously to π, specify the presence of contextual representations for elements in Σ^{2^ℓ}. Writing $\boldsymbol{x} \in \Sigma^{2^\ell}$ as $\boldsymbol{x} = (\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_{\ell-1})$ with $\boldsymbol{z}_0 \in \Sigma$ and $\boldsymbol{z}_{\ell'} \in \Sigma^{2^{\ell'}}, \ell' \in \{0, \ldots, \ell-1\}$, we define:

$$\pi_{\boldsymbol{x}}^{(\ell)} = \pi_{\boldsymbol{z}_0} \wedge \underbrace{\bigwedge_{\ell'=1}^{\ell-1} \phi_{\boldsymbol{x}_{\leq \ell'-1} \leftarrow \boldsymbol{z}_{\ell'}}^{(\ell')}}_{\substack{\text{Verify correct representations} \\ \text{throughout layers are added to } \boldsymbol{z}_0 \in \Sigma}} \tag{45}$$

where $\boldsymbol{x}_{\leq \ell'} = (\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{\ell'})$. This checks the presence of $\boldsymbol{x} \in \Sigma^{2^\ell}$ as the contextual representation by asserting that the individual representations in $\boldsymbol{x}$ at lower levels were indeed added by checking the formulas $\phi_{\boldsymbol{x}_{\leq \ell'-1} \leftarrow \boldsymbol{z}_{\ell'}}^{(\ell')}$.

We now define, for $\boldsymbol{x}, \boldsymbol{y} \in \Xi = \Sigma^{2^\ell}$:

$$\phi_{\boldsymbol{x} \leftarrow \boldsymbol{y}}^{(\ell)} = \bigvee_{\boldsymbol{\mathcal{X}} \in \mathrm{OS}(\Xi)} \left[\underbrace{\mathsf{order}^{(\ell-1)}(\boldsymbol{\mathcal{X}})}_{\substack{\text{Identify correct ordered subset of} \\ \text{representations in the previous layer}}} \wedge \underbrace{\mathsf{best}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\mathcal{X}})}_{\substack{\text{Check if } \boldsymbol{y} \text{ is best representation for } \boldsymbol{x} \\ \text{given the correct ordered subset } \boldsymbol{\mathcal{X}}}} \right] \tag{46}$$

Intuitively, the formula iterates over all possible ordered subsets of Ξ, checks which one describes the string in the past, and then asserts whether $\boldsymbol{y}$ is the best symbol to add to $\boldsymbol{x}$ given the set of contextual representations $\boldsymbol{\mathcal{X}}$. Here, $\mathsf{order}^{(\ell)}$ is a formula that checks whether $\omega(\boldsymbol{w}) = \boldsymbol{\mathcal{X}}$ by making sure that the string in the past follows the same order as $\boldsymbol{\mathcal{X}}$:

$$\mathsf{order}^{(\ell)}(\boldsymbol{\mathcal{X}}) \overset{\text{def}}{=} \underbrace{\left[\Diamond(\pi_{\boldsymbol{z}_1}^{(\ell)} \wedge \Diamond(\pi_{\boldsymbol{z}_2}^{(\ell)} \wedge \cdots \wedge (\Diamond \pi_{\boldsymbol{z}_{|\boldsymbol{\mathcal{X}}|}}^{(\ell)}))) \right]}_{\text{Elements of } \boldsymbol{\mathcal{X}} \text{ are present in correct order}} \wedge \underbrace{\bigwedge_{\boldsymbol{z} \in \Xi \setminus \boldsymbol{\mathcal{X}}} \neg \Diamond \pi_{\boldsymbol{z}}^{(\ell)}}_{\substack{\text{Representations not in } \boldsymbol{\mathcal{X}} \\ \text{are not present}}} \tag{47}$$

Analogously to Eq. (44), $\mathsf{best}^{(\ell)}$ checks whether $\boldsymbol{y}$ is the best symbol to add to $\boldsymbol{x}$ given the set of contextual representations $\boldsymbol{\mathcal{X}}$ by *(i)* asserting $\boldsymbol{x}$ is at the current position, *(ii)* there are no representations in the past of $\boldsymbol{x}$ given the current ordered subset $\boldsymbol{\mathcal{X}}$ that have a higher score than $\boldsymbol{y}$ and *(iii)* there exists a position in $\boldsymbol{\mathcal{X}}$ with $\boldsymbol{y}$ in the past such that there are no representations with equal scores to its left (leftmost tiebreaking).

$$\mathsf{best}^{(\ell)}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\mathcal{X}}) \overset{\text{def}}{=} \pi_{\boldsymbol{x}}^{(\ell)} \wedge \underbrace{\bigwedge_{\substack{\boldsymbol{z} \in \boldsymbol{\mathcal{X}}: \\ \boldsymbol{y} \prec_{\boldsymbol{x}} \boldsymbol{z}}} \neg \Diamond \pi_{\boldsymbol{z}}^{(\ell)}}_{\substack{\text{No previous representations} \\ \text{with higher scores than } \boldsymbol{y}}} \wedge \Diamond \left(\pi_{\boldsymbol{y}}^{(\ell)} \wedge \underbrace{\bigwedge_{\substack{\boldsymbol{z} \in \boldsymbol{\mathcal{X}} \setminus \{\boldsymbol{y}\}: \\ \boldsymbol{y} \simeq_{\boldsymbol{x}} \boldsymbol{z}}} \neg \Diamond \pi_{\boldsymbol{z}}^{(\ell)}}_{\substack{\text{There is } \boldsymbol{y} \text{ in the past with no} \\ \text{equally-scored representations in its past}}} \right). \tag{48}$$

Finally, let $\mathcal{L}$ be the final transformer layer and let $F \subseteq \Sigma^{2^L}$ be the set of representations for EOS that lead to string acceptance by T. The **LTL**$[\Diamond]$ formula ψ representing T simply has to check whether the representation for EOS is in F:

$$\psi = \bigvee_{\boldsymbol{x} \in F} \pi_{\boldsymbol{x}}^{(L)}. \tag{49}$$

$\blacksquare$

D.3 $\mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ as POFAs

Theorem D.1. *Let $\mathsf{T} \in \mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ be a transformer. Then, there exists an equivalent POFA.*

Proof. We will construct a semiautomaton $\mathcal{A}$ that will, after reading $\boldsymbol{w}_{\leq t}$, store in its state the ordered subsets $\mathcal{X}^{(\ell)} \in \mathrm{OS}(\Sigma^{2^\ell})$ of contextual representations for all the transformer layers. In other words, it will hold L *equivalence classes* of the string $\boldsymbol{w}_{\leq t}$, one for each layer. This state will be updated sequentially according to the self-attention mechanism implemented by the transformer.

Formally, given a transformer $\mathsf{T} \in \mathcal{T}_{\blacktriangleleft}^{\mathsf{F}}$ with the attention mechanism implemented by the function $\boldsymbol{\lambda_x}$ (cf. Eq. (42b)) over the alphabet Σ, we define the semiautomaton $\mathcal{A} = (\Sigma, Q, \delta)$. We take the set of states Q to be

$$Q \overset{\text{def}}{=} \underbrace{\mathrm{OS}(\Sigma) \times \cdots \times \mathrm{OS}(\Sigma^{2^L})}_{\text{Ordered sets of representations of all layers.}} . \tag{50}$$

For clarity, we will explicitly write out the states $q \in Q$ in their components:

$$q = \begin{pmatrix} \mathcal{X}^{(0)} \\ \mathcal{X}^{(1)} \\ \vdots \\ \mathcal{X}^{(L)} \end{pmatrix} \tag{51}$$

with $\mathcal{X}^{(\ell)} \in \mathrm{OS}(\Sigma^{2^\ell})$ for $\ell \in \{0, \ldots, L\}$.

$\mathcal{A}$ will update $\mathcal{X}^{(\ell)}$ with new occurrences of contextual representations by "appending" new contextual representations. Let us describe how the transition function δ updates the state q upon reading the symbol w. We write q for the source state and $\mathcal{X}^{(\ell)}$ for its components, and q' for the target state and $\mathcal{X}'^{(\ell)}$ for its components.

We then define $\boldsymbol{x}'^{(0)} = w$ to mean that the static representation of this symbol is the symbol itself. For $\ell \geq 1$, we define

$$\boldsymbol{x}'^{(\ell)} \overset{\text{def}}{=} \begin{pmatrix} \boldsymbol{x}'^{(\ell-1)} \\ \boldsymbol{\lambda_{\boldsymbol{x}'^{(\ell-1)}}}\big(\mathcal{X}^{(\ell-1)}\big) \end{pmatrix} \tag{52}$$

which simulates the ℓth layer of T by *(1)* copying the symbol's representation $\boldsymbol{x}'^{(\ell-1)}$ from the previous layer into the first component (residual connection) and *(2)* computing the attended-to representation based on all the contextual representations seen so far at the previous layer ($\mathcal{X}^{(\ell-1)}$) and the symbol's representation at the previous layer ($\boldsymbol{x}'^{(\ell-1)}$). Crucially, Eq. (52) can be computed *in advance* (knowing the scoring function score) for any $\mathcal{X}^{(\ell)} \in \mathrm{OS}(\Sigma^{2^\ell})$ and $\boldsymbol{x} \in \Sigma^{2^\ell}$ for all $\ell \in \{0, \ldots, L\}$ due to the finiteness of all the considered states.

We then update the set of observed contextual representations as

$$\mathcal{X}'^{(\ell)} \overset{\text{def}}{=} \mathcal{X}^{(\ell)} \cup \{\boldsymbol{x}'^{(\ell)}\} \tag{53}$$

to incorporate the information about the new contextual representations into the ordered sets of seen contextual representations at each layer $\ell \in \{0, \ldots, L\}$. The union in Eq. (53) is to be interpreted as adding an element to an ordered set.

Defining

$$q' = \begin{pmatrix} \mathcal{X}'^{(0)} \\ \mathcal{X}'^{(1)} \\ \vdots \\ \mathcal{X}'^{(L)} \end{pmatrix} \tag{54}$$

and setting $\delta(q, w) = q'$, for all choices of q and w, we have defined the transition function δ that updates each state with the new contextual representations at each layer.

The set of observed contextual representations $\mathcal{X}$ satisfies a partial order as the concatenation of a new representation at a new position at every step implies we transition into novel states. This further implies that the semiautomaton is partially ordered.

We construct an *automaton* from $\mathcal{A}$ by setting the initial and final states. We set the initial state to be the one with empty ordered subsets of observed representations: $\mathcal{X}^{(\ell)} \overset{\text{def}}{=} \emptyset$ for all $\ell \in \{0, \ldots, L\}$. A subset of $\mathrm{OS}(\Sigma^{2^L})$ will lead to T accepting a string. We set the states with $\mathcal{X}^L$ in that set to be final, leading to an equivalent automaton. ∎

E Duality with $\mathcal{T}_\blacktriangleright^P$, LTL[⟁]

The class of transformers $\mathcal{T}_\blacktriangleleft^F$, the main focus of the paper, has a natural symmetric characterization in $\mathcal{T}_\blacktriangleright^P$: While $\mathcal{T}_\blacktriangleleft^F$ can only peek strictly into the past, $\mathcal{T}_\blacktriangleright^P$ can symmetrically only peek strictly into the future using strict past masking and rightmost UHA. $\mathcal{T}_\blacktriangleright^P$ can be informally seen as transformers in $\mathcal{T}_\blacktriangleleft^F$ that instead read symbols *right-to-left*, only considering the future when updating a symbol representation, analogously to the duality between $\mathcal{R}$-trivial and $\mathcal{L}$-trivial languages (Brzozowski and Ellen, 1980).

Thus, all results for $\mathcal{T}_\blacktriangleleft^F$ and **LTL**[◇] apply analogously to the symmetric case of $\mathcal{T}_\blacktriangleright^P$ and **LTL**[⟁]. **LTL**[⟁] is the dual of **LTL**[◇]—it only permits peeking into the future rather than the past. Similarly, partially ordered reverse automata (RPOFA) are semiautomata whose reversal (automaton constructed by inverting the directionality of the transitions) are POFAs. The reverse of a RPOFA is then homomorphic to a cascade product of half-resets. We thus can write the dual statement of Thm. 3.2.

Theorem E.1. *Let $T \in \mathcal{T}_\blacktriangleright^P$ be a transformer. Then, there exists an equivalent formula $\psi \in$ **LTL**[⟁].*
*Let $\psi \in$ **LTL**[⟁] be a formula. Then, there exists an equivalent transformer $T \in \mathcal{T}_\blacktriangleright^P$.*