EXTRACTING LINEAR TEMPORAL LOGIC FORMULAS FROM HARD ATTENTION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

There has been increasing interest into finding exact encodings of hard attention transformers (HAT) for binary classification into Linear Temporal Logic (LTL), to verify and reason about such models using the mature ecosystem of LTL tools. Although several theoretical encoding approaches have been proposed, to the best of our knowledge, none of them have been implemented. In this work, we propose an exact encoding of a HAT into LTL with past amenable to practical implementation. We also extract a *sufficient reason* for an individual input w: an LTL formula guaranteeing that any other input satisfying it receives the same classification as w. We implement both the encoding of a HAT and of a sufficient reason for an input and perform experiments using synthetic data. We also include experiments for the balanced parenthesis problem.

1 INTRODUCTION

Transformers are neural architectures enriched with *attention* mechanisms, which allow them to focus on those parts of the input and activation space most relevant for the output of the model Vaswani et al. (2017). However, like many other neural architectures, transformers are notoriously opaque. Explaining and predicting their behaviour, characterising which tasks they can learn, and proving formal guarantees all remain challenging. To address this, recent work has characterised the expressive power of transformer variants with different architectural components, notably (unique) hard attention Lin & Barcelo (2026); Zhang et al. (2024). A particularly promising direction is the exact translation of hard attention transformers (HATs) for binary classification into Linear Temporal Logic (LTL) Yang et al. (2024); Jerad et al. (2025); Li & Cotterell (2025). LTL is a natural target for two reasons: its mature tool ecosystem can be leveraged to verify and reason about such models, and transformers have been shown to learn to solve LTL problems in practice Hahn et al. (2021).

Existing encodings of HATs into equivalent LTL formulas, however, serve mainly as proofs of expressivity results and, to the best of our knowledge, they have not been implemented. To address this gap, we propose an exact *global* encoding of leftmost HATs without positional encoding or masking into LTL with past operators, designed for practical implementation. We prove that our encoding is correct (Theorem 3.1).

The global encoding of a transformer can be very long, as it must capture the model's behaviour on every input over a given alphabet. As a more practical alternative, we also show how to encode a *sufficient reason* for a given input w: an LTL formula that describes only a relevant part of the model, with the property that every input satisfying the formula receives exactly the same classification as w. This notion has been studied for Bayesian network classifiers Shih et al. (2018) and decision trees Arenas et al. (2022) but, to the best of our knowledge, not for transformers. We prove that our construction yields a sufficient reason for any input (Theorem 3.2).

We implement both the global encoding and the sufficient reason extraction, and measure the length of the resulting formulas on the balanced parentheses task and on synthetic datasets. While global encodings are feasible only over small alphabets, sufficient reasons also scale to larger ones. We also observe that their length varies considerably with the input word and the transformer model. Proofs not essential to our main results are provided in the Appendix.

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

2 PRELIMINARIES

We use uppercase (resp. lowercase) bold letters for matrices (resp. vectors). We write $\mathbf{X}[i, j]$ for the element in row i, column j of a matrix $\mathbf{X}$, and $\mathbf{X}[i]$ for the vector corresponding to row i. For a vector $\mathbf{x}$, $\mathbf{x}[i]$ denotes its i-th component. Vector-to-vector functions are applied to matrices row-wise. We use $\mathbb{N}$ (resp. $\mathbb{N}_0$) for the set of natural numbers excluding (resp. including) 0.

A (finite) *alphabet* is a finite non-empty set of *symbols*. A *word* over an alphabet Σ is a finite non-empty sequence of symbols of Σ. We write $|w|$ and $|\Sigma|$ to represent the length of a word w and the number of elements in an alphabet Σ, respectively. A *position* in a word of length n is a number from 1 to n; for a word of the form $a_1, \ldots, a_n$, the *symbol in position* i is a_i, for each $1 \leq i \leq n$.

2.1 HARD ATTENTION TRANSFORMERS

We consider a standard transformer model with *leftmost hard-max attention*, *single-head attention*, no *positional encoding*, and no *masking*. For this definition, we assume an arbitrary alphabet Σ.

A *transformer* with $L \in \mathbb{N}$ layers and dimensions d and $d' \in \mathbb{N}$ is a tuple

$$\left\langle \mathbf{W}_{enc}, \{\mathbf{Q}^\ell\}_{1\leq\ell\leq L}, \{\mathbf{K}^\ell\}_{1\leq\ell\leq L}, \{\mathbf{V}^\ell\}_{1\leq\ell\leq L}, \{\mathbf{O}^\ell\}_{1\leq\ell\leq L}, \sigma, \theta \right\rangle \tag{1}$$

where $\mathbf{W}_{enc}$ is a (learnable) matrix of dimension $|\Sigma| \times d$ called the *encoding matrix*, where each row is unique and *corresponds* to a symbol in Σ; $\mathbf{Q}^\ell$, $\mathbf{K}^\ell$, $\mathbf{V}^\ell$ are (learnable) matrices of dimension $d \times d'$, called the *query*, *key*, and *value* matrices; $\mathbf{O}^\ell$ is a (learnable) matrix of dimension $d' \times d$ called the *output* matrix; σ is a continuous (typically non-linear) real function called the *activation* function, and θ is a real value called the *classification threshold*.

A transformer T induces a *classification function* that maps words over Σ into a Boolean value `True` or `False`; by a slight abuse of notation, we denote this function also by T. We next define this function for an arbitrary transformer T of the form in Equation 1. Consider an arbitrary word $w = a_1, \ldots, a_n$ over Σ. The Boolean value $T(w)$ is computed from the *residual matrices* $\mathbf{X}_0, \ldots, \mathbf{X}_L$, defined inductively as follows. Matrix $\mathbf{X}_0$ is a matrix of dimension $n \times d$ where row $\mathbf{X}_0[i]$ is identical to the row of $\mathbf{W}_{enc}$ corresponding to symbol a_i. Next, for each $1 \leq \ell \leq L$, we let

$$\mathbf{X}_\ell = \sigma(\mathbf{A}^\ell \mathbf{X}_{\ell-1} \mathbf{V}^\ell \mathbf{O}^\ell) + \mathbf{X}_{\ell-1}, \tag{2}$$

where $\mathbf{A}^\ell$ is the *attention matrix* at layer ℓ defined as

$$\mathbf{A}^\ell = \texttt{LArgMax}(\mathbf{X}_{\ell-1}\mathbf{Q}^\ell(\mathbf{X}_{\ell-1}\mathbf{K}^\ell)^T). \tag{3}$$

The first row of $\mathbf{X}_L$ determines the output; in particular, $T(w) = \texttt{True}$ if $\mathbf{X}_L[1, j] > \theta$ for all $1 \leq j \leq d$, and otherwise $T(w) = \texttt{False}$.

For each $0 \leq \ell \leq L$ and position $1 \leq i \leq n$, we refer to $\mathbf{X}_\ell[i]$ as the *feature vector* for i in layer ℓ. Using `LArgMax` in Equation 3 implements "hard" attention: each row of $\mathbf{A}^\ell$ has exactly one position equal to 1 and all others equal to 0. We say that position $1 \leq i \leq n$ *attends* to position $1 \leq j \leq n$ in layer ℓ if $\mathbf{A}^\ell[i, j] = 1$. Then, by Equation 2, layer ℓ updates the feature vector of position i using only the feature vector of position j from layer $\ell - 1$. In contrast, soft attention replaces `LArgMax` with functions that can assign positive values to all positions, such as `SoftMax`, so that the update uses a weighted combination of all feature vectors from layer $\ell - 1$.

2.2 LINEAR TEMPORAL LOGIC

An LTL formula with past (LTL$^-$) (Lichtenstein et al., 1985) is defined inductively as follows. For an alphabet Σ, each symbol in Σ is an LTL$^-$ formula. Special symbol $\top$, not in Σ, is an LTL$^-$ formula. If ϕ and ψ are LTL$^-$ formulas then $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $\neg\phi$, $(\phi \,\mathsf{U}\, \psi)$, $(\phi \,\mathsf{S}\, \psi)$, and $\mathsf{Y}\psi$ are LTL$^-$ formulas. We write $(\phi \rightarrow \psi)$ as a shorthand for $(\neg\phi \vee \psi)$. A word w either *satisfies* an LTL$^-$ formula φ in a position $1 \leq i \leq |w|$, written $w, i \models \varphi$, or does not satisfy it, written $w, i \not\models \varphi$, according to:

- $w, i \models p$ iff p is in the i-th position in w; and $w, i \models \top$ for each $1 \leq i \leq |w|$;
- $w, i \models \neg\,\phi$ iff $w, i \not\models \phi$;
- $w, i \models (\phi \wedge \psi)$ iff $w, i \models \phi$ and $w, i \models \psi$; and $w, i \models (\phi \vee \psi)$ iff $w, i \models \phi$ or $w, i \models \psi$;

- $w, i \models (\phi \,\mathsf{U}\, \psi)$ iff there is $j \geq i$ such that $w, j \models \psi$ and, for all k such that $i \leq k < j$, we have $w, k \models \phi$; and $w, i \models (\phi \,\mathsf{S}\, \psi)$ iff there is $j \leq i$ such that $w, j \models \psi$ and, for all k such that $j < k \leq i$, we have $w, k \models \phi$; finally, $w, i \models (\mathsf{Y}\phi)$ iff $i > 1$ and $w, i-1 \models \phi$.

We define $\Diamond^\star \varphi$ as $(\top \,\mathsf{S}(\varphi \wedge \neg(\mathsf{Y}\top)))$. The formula holds at any position iff φ holds at the first position [1]. Finally, we say that a word w *satisfies* an LTL$^-$ formula φ, written $w \models \varphi$, iff $w, 1 \models \varphi$.

3 HAT ENCODING INTO LTL$^-$

In this section, we fix an arbitrary alphabet Σ and a transformer T of the form of Equation 1, L layers, and dimensions d and d', and we show how to encode T into a formula Φ_T provably equivalent to it, in the sense that both the formula and the transformer classify any word over Σ in the same way.

The central idea is representing each layer ℓ of transformer T as a set of *update rules* that transform words over a special auxiliary alphabet $\Sigma_{\ell-1}$ into words over another auxiliary alphabet Σ_ℓ. For example, the update rules of layer 1 transform an input word w over Σ into a word of the same length over auxiliary alphabet Σ_1. Then, the update rules of layer 2 transform this word into another over alphabet Σ_2, and so on. Each symbol of an auxiliary alphabet represents a specific attention pattern and corresponds to a vector that might appear as a residual matrix row. The alphabets are computed from T and Σ alone, independently of any inputs. Although there are infinite valid transformer inputs, the alphabets and the sets of update rules are finite, because the finiteness of Σ and the hard attention of T together ensure there are only finitely many possibilities for the values of residual matrices.

We then consider the symbols of the output alphabet Σ_L that represent a positive classification by T, and then we "unfold" them applying the update rules backwards, until only symbols of the original alphabet Σ remain. The result is a finite set of LTL$^-$ formulas over Σ that characterise exactly those words classified positive by T. The disjunction of these formulas is a single LTL$^-$ formula Φ_T over Σ equivalent to T, in the sense that for each word w over Σ, $T(w) = \texttt{true}$ if and only if $w \models \Phi_T$.

3.1 REPRESENTING LAYERS AS UPDATE RULES

We start by defining inductively our auxiliary alphabets Σ_ℓ, which describe potentially relevant attention patterns. For example, the alphabet Σ_1 consists of symbols $s_{a,b}$, with $a, b \in \Sigma$, each expressing the claim that "this position started as symbol a and attended to a position with symbol b".

Definition 1. *Let $\Sigma_0 = \Sigma$. For each $1 \leq \ell \leq L$, Σ_ℓ is the alphabet consisting of a symbol $s_{a,b}$ for every ordered pair (a, b) of (not necessarily distinct) symbols of $\Sigma_{\ell-1}$.*

Recursive symbol nesting ensures that alphabets from distinct layers are always disjoint. Moreover, each symbol s in an alphabet Σ_ℓ *corresponds* to a vector $\mathbf{x}_s$, defined inductively as follows.

Definition 2. *For each $a \in \Sigma_0$, its* corresponding *vector $\mathbf{x}_a$ is the row of $\mathbf{W}_{enc}$ corresponding to a. Furthermore, for each $1 \leq \ell \leq L$ and symbol $s_{a,b} \in \Sigma_\ell$, its* corresponding *vector $\mathbf{x}_{s_{a,b}}$ is*

$$\mathbf{x}_{s_{a,b}} \;=\; \sigma\big(\mathbf{x}_b\, \mathbf{V}^\ell\, \mathbf{O}^\ell\big) + \mathbf{x}_a,$$

where $\mathbf{x}_a$ and $\mathbf{x}_b$ are the symbols corresponding to $a, b \in \Sigma_{\ell-1}$.

The corresponding vectors can be used to recover the residual matrices from the words generated by the update rules. We state and prove this formally in Lemma 2.

We are now ready to define the update rules. For each layer $1 \leq \ell \leq L$, we define a set $\mathcal{R}_\ell$ of update rules, where each rule is an expression of the form $\varphi \to s$ where the *body* φ is an LTL$^-$ formula that mentions only symbols from $\Sigma_{\ell-1}$ and the *head* s is a symbol in Σ_ℓ.

Definition 3. *For each (not necessarily distinct) pair $a, b \in \Sigma_{\ell-1}$, let $\alpha_{a,b} = \mathbf{x}_a \mathbf{Q}^\ell (\mathbf{x}_b \mathbf{K}^\ell)^T$. For each $a \in \Sigma_{\ell-1}$, let $\beta_{a,1}, \ldots, \beta_{a,m_a}$ be the partition of $\Sigma_{\ell-1}$ into classes of equal $\alpha_{a,b}$ ordered by increasing value, so that $\alpha_{a,b} < \alpha_{a,b'}$ whenever $b \in \beta_{a,j}$, $b' \in \beta_{a,j'}$, and $j < j'$. Finally, let j_b be the index such that $b \in \beta_{a,j_b}$. The set $\mathcal{R}_\ell$ contains the following rule for each $a, b \in \Sigma_{\ell-1}$:*

$$a \wedge \Diamond^\star \left(\bigwedge_{b' \in \beta_{a,j_b}} (\neg b' \,\mathsf{U}\, b) \;\wedge \bigwedge_{j' > j_b} \bigwedge_{b' \in \beta_{a,j'}} \neg(\top \,\mathsf{U}\, b') \right) \to s_{a,b}. \tag{4}$$

[1] Note that $\neg(\mathsf{Y}\top)$ can only hold at the first position of a word.

As described at the start of the section, update rules apply to a word over $\Sigma_{\ell-1}$, producing a new word over Σ_ℓ of the same length. Intuitively, rule (4) states that each position in the word with symbol a attending to a position with symbol b in layer ℓ becomes symbol $s_{a,b}$ in the new word. This happens when no b' from any $\beta_{a,j'}$ with $j' > j_b$ appears in the word, and no $b' \in \beta_{a,j_b}$ distinct from b appears before b in the word–as otherwise a's position would attend to them instead of b. Note that the first conjunction in Equation 4 includes the conjunct $(\neg b \,\mathsf{U}\, b)$, which states that b occurs in the word.

We observe that each symbol in Σ_ℓ appears as the head of exactly one rule in $\mathcal{R}_\ell$.

Lemma 1. *For all $1 \leq \ell \leq L$ and all $s \in \Sigma_\ell$, there is exactly one rule in $\mathcal{R}_\ell$ with head s.*

3.2 Global Encoding

Next we show how we can combine the update rules of $\mathcal{R}_1, \ldots, \mathcal{R}_L$ defined in the previous section into an LTL$^-$ formula that simulates the application of the transformer. Our procedure is based on *rule unfolding*: we consider the bodies of rules of $\mathcal{R}_L$ producing symbols corresponding to positive classifications; then, we "unfold" these formulas using the rules of $\mathcal{R}_{L-1}$, then $\mathcal{R}_{L-2}$, ..., and finally $\mathcal{R}_1$. This results in a set of formulas over Σ, which completely characterise T. We then take a disjunction of these formulas, which results in an LTL$^-$ formula equivalent to T.

We next define inductively the unfolding operators $\phi^{\circ,1}, \ldots, \phi^{\circ,L}$.

Definition 4. *For each LTL$^-$ formula ϕ, we let $\phi^{\circ,0} = \phi$ and $\phi^{\circ,\ell} = \psi^{\circ,\ell-1}$ for each $1 \leq \ell \in \leq L$, where ψ replaces each occurrence of an $s \in \Sigma_\ell$ in ϕ by $\mathsf{body}(r_s)$, for r_s the rule in $\mathcal{R}_\ell$ with head s.*

Intuitively, for an LTL$^-$ formula ϕ and $0 \leq \ell \leq L$, $\phi^{\circ,\ell}$ is the result of ℓ rounds of unfolding ϕ using the rules of $\mathcal{R}_\ell, \ldots, \mathcal{R}_1$. In each round, Lemma 1 ensures that all symbols can be unfolded, and Equation 4 ensures that only symbols from the preceding alphabet are produced; thus, $\phi^{\circ,\ell}$ is always a formula over Σ. We use the unfolding operator to define the transformer's global encoding.

Definition 5. *Let Σ_L^+ be the subset of Σ_L containing each $a \in \Sigma_L$ such that $\mathbf{x}_a[i] > \theta$ for all $1 \leq i \leq d$. The formula Φ_T is*

$$(\bigvee_{a \in \Sigma_L^+} a^{\circ,L}).$$

We next state our main result: the formula in Definition 5 is equivalent to the transformer.

Theorem 3.1. *For all words w over Σ, it holds that $T(w) = \texttt{True}$ iff $w \models \Phi_T$.*

We emphasise that *everything presented so far is independent of any particular input*. The update rules are computed directly from the transformer T and the alphabet Σ alone; no input word plays any role in their construction. They describe the transformer's action on *every* possible input over Σ.

The rest of this section is devoted to proving Theorem 3.1. To this end, we consider an *arbitrary* word $w = a_1, \ldots, a_n$ over Σ and show that $T(w) = \texttt{True}$ if and only if $w \models \Phi_T$. Throughout this section, we let $\mathbf{X}_0, \ldots, \mathbf{X}_L$ and $\mathbf{A}^1, \ldots, \mathbf{A}^L$ be the residual and attention matrices, respectively, generated when applying T to w, defined as in Section 2.1.

Our first step is to show that the update rules have their intended meaning. To this end, we define the words produced by the update rules, formalizing the process outlined at the start of Section 3. For a word w', w'_i is the symbol in position i of w'. The definition is inductive on the L layers of T.

Definition 6. *Let $w^0 = w$. For each $1 \leq \ell \leq L$, w^ℓ is the word where w_i^ℓ is the symbol $s_{a,b} \in \Sigma_\ell$ with $a = w_i^{\ell-1}$, $b = w_j^{\ell-1}$, and $1 \leq j \leq n$ the position attended to by i in layer ℓ, for $1 \leq i \leq n$.*

Each symbol in w^ℓ is well-defined since $a, b \in \Sigma_{\ell-1}$ guarantees $s_{a,b} \in \Sigma_\ell$. Next, we show that the words $w^0, \ldots, w^L$ correspond to the vectors $\mathbf{X}_0, \ldots, \mathbf{X}_L$; more specifically, we show that the vector corresponding to each symbol w_i^ℓ is precisely the i-th row of $\mathbf{X}_\ell$.

Lemma 2. *For each $0 \leq \ell \leq L$ and $1 \leq i \leq n$, the vector $\mathbf{x}_{w_i^\ell}$ corresponding to symbol w_i^ℓ is $\mathbf{X}_\ell[i]$.*

Proof. We use proof by induction on ℓ. For $\ell = 0$, consider an arbitrary $1 \leq i \leq L$ and recall that $\mathbf{X}_0[i]$ is the row of $\mathbf{W}_{enc}$ corresponding to symbol a_i, which is w_i^0 by Definition 6, and so the lemma holds. For the induction step, suppose the lemma holds for some $0 \leq \ell \leq L - 1$, and consider

an arbitrary $1 \leq i \leq n$; we prove that $\mathbf{X}_{\ell+1}[i]$ corresponds to $w_i^{\ell+1}$. By Definition 6, $w_i^{\ell+1}$ is the symbol $s_{a,b} \in \Sigma_{\ell+1}$ where $a = w_i^{\ell}$ and $b = w_j^{\ell}$ with i attending to j in layer $\ell + 1$. By Definition 2, the vector corresponding to $s_{a,b}$ is $\mathbf{x}_{s_{a,b}} = \sigma(\mathbf{x}_b \mathbf{V}^{\ell+1}\mathbf{O}^{\ell+1}) + \mathbf{x}_a$, for $\mathbf{x}_b$ and $\mathbf{x}_a$ the vectors corresponding to b and a, respectively. It remains to argue that $\mathbf{X}_{\ell+1}[i] = \mathbf{x}_{s_{a,b}}$. By Equation 2, $\mathbf{X}_{\ell+1}[i] = \sigma(\mathbf{A}^{\ell+1}[i]\,\mathbf{X}_\ell \mathbf{V}^{\ell+1}\mathbf{O}^{\ell+1}) + \mathbf{X}_\ell[i]$. However, since i attends to j in layer $\ell+1$, all elements in $\mathbf{A}^{\ell+1}[i]$ are 0 except the j-th, which is 1, and so $\mathbf{A}^{\ell+1}[i]\,\mathbf{X}_\ell = \mathbf{X}_\ell[j]$, which is $\mathbf{x}_b$ by induction hypothesis; similarly, $\mathbf{X}_\ell[i] = \mathbf{x}_a$. Thus, we obtain $\mathbf{X}_{\ell+1}[i] = \sigma(\mathbf{x}_b\mathbf{V}^{\ell+1}\mathbf{O}^{\ell+1}) + \mathbf{x}_a = \mathbf{x}_{s_{a,b}}$. □

Next, we show that each round of unfolding (as per Definition 4) preserves the relevant semantics.

Lemma 3. *For each $1 \leq \ell \leq L$, position $1 \leq i \leq n$, word w over Σ, and LTL$^-$ formula ϕ over Σ_L, it holds that $w^L, i \models \phi$ iff $w^{L-\ell}, i \models \phi^{\circ,\ell}$.*

Proof. The lemma is a direct consequence of the following claim.

Claim 1. *For each layer $1 \leq \ell \leq L$, position $1 \leq i \leq n$, word w over Σ_0 and LTL$^-$ formula ϕ over Σ_ℓ, it holds that $w^\ell, i \models \phi$ iff $w^{\ell-1}, i \models \phi^{\circ,1}$.*

The argument is by structural induction. In the base case, if $\phi = \top$ then, by definition, $\phi = \phi^{\circ,1}$, so $w^\ell, i \models \phi$ iff $w^{\ell-1}, i \models \phi^{\circ,1}$, since $\top$ always holds. Now consider $\phi = s_{a,b}$ for some $s_{a,b} \in \Sigma_\ell$, with $\ell > 0$. By Definition 6, $w^\ell, i \models s_{a,b}$ iff $w^{\ell-1}, i \models a$, i attends to some j in layer $\ell - 1$ and $w^{\ell-1}, j \models b$ (or equivalently $w_j^{\ell-1} = b$). Whenever i attends to j and $w_j^{\ell-1} = b$, it cannot be that there is $j' < j$ such that $w_{j'}^{\ell-1} = b'$ with $\mathbf{x}_{b'} = \mathbf{x}_b$ (as in that case i would attend j' instead of j). Also, it cannot be that there is b' and $j' \neq j$ with $\alpha_{a,b'} = \mathbf{x}_a\mathbf{Q}^\ell(\mathbf{x}_{b'}\mathbf{K}^\ell)^T > \alpha_{a,b} = \mathbf{x}_a\mathbf{Q}^\ell(\mathbf{x}_b\mathbf{K}^\ell)^T$ and $w_{j'}^{\ell-1} = b'$ (in that case i would attend to j' instead of j). Then, $w^\ell, i \models s_{a,b}$ iff

$$w^{\ell-1}, i \models a \wedge \Diamond^\star \left(\bigwedge_{b' \in \beta_{a,j}} (\neg b' \,\mathsf{U}\, b) \right) \wedge \Diamond^\star \left(\bigwedge_{b' \in \beta_{a,j'}, j' > j} \neg(\top \,\mathsf{U}\, b') \right).$$

Hence, $w^{\ell-1}, i \models \mathsf{body}(r_{s_{a,b}})$ where $r_{s_{a,b}}$ is the (unique by Lemma 1) rule with $\mathsf{head}(r_{s_{a,b}}) = s_{a,b}$. That is, $w^\ell, i \models s_{a,b}$ iff $w^{\ell-1}, i \models {s_{a,b}}^{\circ,1}$. Now, suppose the claim holds for LTL$^-$ formulas ϕ_1, ϕ_2. We argue for the case of "until". The remaining cases are similar (the full proof is in the Appendix).

- $\phi = \phi_1 \,\mathsf{U}\, \phi_2$. We have that $w^\ell, i \models \phi$ iff $w^\ell, i \models \phi_1 \,\mathsf{U}\, \phi_2$ (since in this case $\phi = \phi_1 \,\mathsf{U}\, \phi_2$) iff there is $j \geq i$ such that $w^\ell, j \models \phi_2$ and, for all k such that $i \leq k < j$, we have $w^\ell, k \models \phi_1$, by the semantics of LTL$^-$. By the inductive hypothesis, $w^\ell, i \models \phi_l$ iff $w^{\ell-1}, i \models {\phi_l}^{\circ,1}$, for $l \in \{1, 2\}$. By the semantics of LTL$^-$, $w^{\ell-1}, i \models {\phi_1}^{\circ,1} \,\mathsf{U}\, {\phi_2}^{\circ,1}$. By definition of $\cdot^{\circ,1}$, we have that ${\phi_1}^{\circ,1} \,\mathsf{U}\, {\phi_2}^{\circ,1} = (\phi_1 \,\mathsf{U}\, \phi_2)^{\circ,1}$. So $w^{\ell-1}, i \models (\phi_1 \,\mathsf{U}\, \phi_2)^{\circ,1}$.

This concludes the proof of Claim 1. Back to the proof of the Lemma, we want to show that $w^L, i \models \phi$ iff $w^{L-\ell}, i \models \phi^{\circ,\ell}$, for all $1 \leq i \leq n$, $1 \leq \ell \leq L$, w over Σ_0 and LTL$^-$ formula ϕ over Σ_L. By Claim 1, $w^L, i \models \phi$ iff $w^{L-1}, i \models \phi^{\circ,1}$ iff $w^{L-1-1}, i \models \phi^{\circ,1^{\circ,1}}$ and so on. By definition of $\cdot^{\circ,\ell}$, $\phi^{\circ,\ell^{\circ,1}} = \phi^{\circ,\ell+1}$. So $w^L, i \models \phi$ iff $w^{L-\ell}, i \models \phi^{\circ,\ell}$, as required. □

We are finally ready to provide the proof of our main theorem.

Theorem 3.1. *For all words w over Σ, it holds that $T(w) = \texttt{True}$ iff $w \models \Phi_T$.*

Proof. By definition of T, we have that $T(w) = \texttt{True}$ iff all elements of $\mathbf{X}_L[1]$ are strictly greater than θ. By Lemma 2, $\mathbf{X}_L[1]$ is the vector corresponding to w_1^L, and so by Definition 7 all elements of $\mathbf{X}_L[1]$ are strictly greater than θ iff $w_1^L \in \Sigma_L^+$. The statement $w_1^L \in \Sigma_L^+$ is equivalent to the statement that $w^L, 1 \models a$ for some $a \in \Sigma_L^+$. By Lemma 3 with $\ell = L$ (and using $w^0 = w$), the last statement holds iff $w, 1 \models a^{\circ,L}$ for some $a \in \Sigma_L^+$. By the semantics of LTL$^-$, the last statement holds iff $w \models \bigvee_{a \in \Sigma_L^+} a^{\circ,L}$. Therefore, we obtain that $T(w) = \texttt{True}$ iff $w \models \bigvee_{a \in \Sigma_L^+} a^{\circ,L} = \Phi_T$. □

3.3 SUFFICIENT REASON

We next show how to extract a *sufficient reason* for why an input w was classified in a certain way. The global formula Φ_T may be seen as one such reason, but it encodes the entire transformer, and so it typically includes plenty of information not relevant to its execution on w. We instead show how to extract a shorter and more relevant formula Φ_T^w, depending on both T and w, such that every input that satisfies Φ_T^w is classified in the same way as w.

Definition 7. *Let w be a word over Σ and let $w^0, \ldots, w^L$ be as in Def. 6. Then, $\Phi_T^w := w_1^{L^\circ, L}$.*

Intuitively, rather than unfolding a rule for each possible symbol representing a positive classification, we unfold only the rule that generates the symbol that *actually* corresponds with the first position in the output residual of T on w. Each word satisfies its sufficient reason formula, as expected.

Proposition 1. *For each word w over Σ, it holds that $w \models \Phi_T^w$.*

We finally show that Φ_T^w is a sufficient reason in the sense described above.

Theorem 3.2. *For all words w, w' over Σ, if $w' \models \Phi_T^w$ then $T(w) = T(w')$.*

Proof. Let w, w' be words over Σ such that $w' \models \Phi_T^w$. We argue that $\mathbf{X}_L[1]$ is the same for both w and w'. Indeed, as $\Phi_T^w = w_1^{L^\circ, L}$, we have that $w' \models w_1^{L^\circ, L}$, which means $w'^{L-L} \models w_1^{L^\circ, L}$ since $w'^0 = w'$. By Lemma 3, $w^{L-\ell}, i \models \phi^{\circ, \ell}$ iff $w^L, i \models \phi$ for every LTL$^-$ formula ϕ. Then, taking $i = 1$ and $\ell = L$, we have $w'^{L-L} \models w_1^{L^\circ, L}$ iff $w'^L \models w_1^L$. By semantics of LTL$^-$, we have that $w'^L \models w_1^L$ iff $w_1'^L = w_1^L$. By Lemma 2, $\mathbf{X}_L[1]$ is the same for w and w'. □

We conclude this section with a property of our HAT transformers that we rely on in the experiments.

Proposition 2. *For each word w over Σ_0, $T(w) =$ `True` iff $T(w') =$ `True`, where w' is the result of removing from w all but the first occurrence of each symbol in Σ_0.*

4 EXPERIMENTS

We evaluate whether the HAT transformer of Section 2.1 learns the balanced parenthesis task and the task of correctly classifying sequences created from LTL$^-$ formulas. We also evaluate and compare the extraction time and output length of the global and sufficient reason procedures of Section 3.

Parenthesis Data. This dataset is an adaptation of the one by Friedman et al. (2023), containing 2,000 words of lengths 10–31 over a vocabulary of two elements–an opening parenthesis and a closing one–equally split into positive and negative examples. Examples are split into 1,400/300/300 for training/validation/testing, respectively. The benchmark is *noisy*: its labels agree with the lenient (prefix validity) matching standard only 91.95% of the time, and only 59.60% for the strict standard of fully balanced parentheses (see the Appendix). Moreover, Proposition 2 implies that our model cannot learn to recognize parenthesis data in general, since words like `()` and `(())` have the same reduction but different classification.

Synthetic Data. We create benchmarks based on five LTL$^-$ target formulas over $\Sigma = \{p_1, \ldots, p_6\}$.

$$\phi_m^U = \bigvee_{i=1}^{m} (\neg a_i \,\mathsf{U}\, b_i), \qquad \phi_m^\wedge = \bigvee_{i=1}^{m} [(\neg a_i \,\mathsf{U}\, b_i) \wedge (\neg a_i \,\mathsf{U}\, c_i)], \tag{5}$$

The first and second targets, ϕ_2^U and ϕ_3^U, are drawn from the left template in Equation (5) with $m \in \{2, 3\}$, respectively, whereas the third and fourth targets, $\phi_2^\wedge$ and $\phi_3^\wedge$, are drawn from the right with $m \in \{2, 3\}$; we ensure that symbols with the same index i are different in each target. The fifth target is $\phi_{pr} = p_1 \wedge (\neg p_3 \mathsf{U}\, p_2) \wedge (\neg p_4 \mathsf{U}\, p_2) \wedge \neg(\top \,\mathsf{U}\, p_5)$, modeled after the priority condition in Equation 4. Datasets are created by sampling and verifying words; we use seeded splitting $70/15/15\%$ for training, validation, and testing. We specify all five targets and explain the generation of the datasets in the Appendix.

All experiments use an AMD EPYC 7642 CPU and NVIDIA A40 GPU. Global extraction runs on one CPU thread, sufficient reason extraction uses the GPU to calculate attention scores and discard

Table 1: Balanced parentheses extraction results over the five seeds. Balanced accuracy is reported as the five-seed mean with $[\min, \max]$ range.

Metric	$L = 1$	$L = 2$	$L = 3$
Balanced accuracy	74.65% $[73.29, 76.80]$	74.65% $[73.29, 76.80]$	74.65% $[73.29, 76.80]$
Median Formula Length	10	110	17,251
Average Formula Length	10.0	105.2	23,860.0

incompatible combinations, while the CPU constructs the formula. We implemented the HAT of Section 2.1 with ReLU. We use the Black LTL$^-$ solver Geatti et al. (2021) to confirm the correctness of word satisfaction and formula equivalence in our implementation. We use AdamW (Loshchilov & Hutter, 2019) for optimization. In both the attention layer and the min operation, we use a straight-through estimator: the forward pass applies the exact discrete operation, while the backward pass uses the gradient of a softmax-based relaxation. For attention, this is the standard softmax derivative; for the minimum, it is the smooth gradient with temperature $\tau = 0.1$. Prediction and extraction always use the hard operations. Unless stated otherwise, we use $d = 32$, $d' = 16$, batch size 16, learning rate 0.001, weight decay 0.0001, and gradient clipping at norm one. Training stops after reaching 100 epochs or after 15 consecutive epochs without improvement in balanced accuracy over the validation dataset; we then select the model with the best balanced accuracy over validation. We consider models with 1–3 layers and seeds $42, 7, 13, 21, 29$ for each of our six datasets (parenthesis + 5 formulas), for a total of 90 runs.

The length of a formula is the number of occurrences of each alphabet symbol, $\top$, and LTL$^-$ operators. For example, formula $(\top \,\mathsf{U}\, a) \vee (\top \,\mathsf{U}\, b) \wedge \neg(\top \,\mathsf{U}\, c)$ has length 12: it has three alphabet symbols a, b, c, three occurrences of $\top$, three occurrences of U and one occurrence each of $\vee$, $\wedge$, $\neg$.

4.1 Global Extraction Results

Here we present the extraction of the global encoding (Definition 5). Each run trained one model and extracted its global encoding. We report the median and mean length across the five seeds.

Results for Parenthesis Data. Table 1 shows that all models achieve the same mean test balanced accuracy of 74.65%, ranging from 73.29% to 76.80% across the five seeds. We verified that each extracted formula agrees with its model on all dataset examples as expected.

We next show the formulas extracted for $L = 1$, using LP and RP for the opening and closing parenthesis symbols, respectively:

$$L = 1, \text{ seeds } 13, 21, 42: \quad \mathsf{LP} \wedge \Diamond^\star (\neg \mathsf{RP} \,\mathsf{U}\, \mathsf{RP}) \wedge \Diamond^\star \top,$$
$$L = 1, \text{ seeds } 7, 29: \quad \mathsf{LP} \wedge \Diamond^\star (\neg \mathsf{LP} \,\mathsf{U}\, \mathsf{LP}) \wedge \Diamond^\star \top.$$

The second formula is equivalent to LP i.e. the word starts with an opening parenthesis. The first additionally requires a closing parenthesis to occur. These two classes agree on every word of each dataset; the shortest word on which they differ is the single opening parenthesis LP. Neither captures balanced parenthesis membership, as both accept `(()`.

For $L = 2$, the five seeds yield two classes of semantically equivalent formulas: seeds 21 and 42 in one, and seeds 7, 13, and 29 in the other. Formula lengths range from 37 to 176. For $L = 3$, the seeds again split into two classes: seeds 7, 21, and 29 in one, and seeds 13 and 42 in the other, with lengths ranging from 3,841 to 64,568. Although the formulas extracted from two- and three-layer transformers are considerably longer, each is semantically equivalent the short formulas for $L = 1$.

Results for Synthetic Data. For this experiment, we create a dataset of 1,000 words per each of the five target formulas, split by seeded shuffling into 700 training, 150 validation, and 150 test examples. Predictive balanced accuracy is evaluated on the test split, and agreement between formula and model is verified on all 1,000 examples. Table 2 shows the results for all five synthetic datasets.

We observe that none of the extracted formulas is logically equivalent to the original target formula. Since the extracted formula is equivalent to the model, this shows that the models did not exactly learn the target formulas, despite their overall good predictive accuracy. The extracted formulas were

Table 2: Synthetic formulas extraction results over the five seeds. Balanced accuracy is reported as the five-seed mean with $[\min, \max]$ range.

Target	L	Balanced accuracy (%)	Median Length	Mean Length
ϕ_2^U	1	77.01 [72.44, 81.83]	402	369.2
ϕ_2^U	2	76.98 [72.15, 80.50]	281,742	271,746.2
ϕ_2^U	3	**78.68** [**74.00**, **81.83**]	$1.493 \cdot 10^{11}$	$1.298 \cdot 10^{11}$
ϕ_3^U	1	**82.45** [**76.77**, **85.86**]	531	522.0
ϕ_3^U	2	80.87 [76.92, 85.86]	258,778	311,985.8
ϕ_3^U	3	82.42 [77.78, 85.23]	$1.251 \cdot 10^{11}$	$1.340 \cdot 10^{11}$
$\phi_2^\wedge$	1	69.13 [64.85, 74.51]	153	256.8
$\phi_2^\wedge$	2	73.40 [69.33, 74.52]	141,963	141,633.0
$\phi_2^\wedge$	3	**74.11** [**70.00**, **75.93**]	$1.082 \cdot 10^{11}$	$1.160 \cdot 10^{11}$
$\phi_3^\wedge$	1	66.66 [61.18, 74.21]	203	216.8
$\phi_3^\wedge$	2	**73.14** [**71.33**, **74.83**]	163,839	176,017.0
$\phi_3^\wedge$	3	72.28 [69.33, 77.03]	$2.312 \cdot 10^{10}$	$7.454 \cdot 10^{10}$
ϕ_{pr}	1	76.83 [65.33, 86.36]	13	24.4
ϕ_{pr}	2	84.90 [74.34, 90.00]	857	1,693.0
ϕ_{pr}	3	**85.18** [**79.87**, **90.00**]	$1.491 \cdot 10^8$	$3.820 \cdot 10^9$

Table 3: Formula sizes across the five synthetic targets and five seeds per target.

Number of layers	Median formula length	Average formula length
1	287	277.8
2	163 839	180 615.0
3	$7.917 \cdot 10^{10}$	$9.164 \cdot 10^{10}$

generally longer than the target formulas. For example, the original $\phi_2^\wedge$ has length 19:

$$\Big(\big((\neg p_6 \,\mathsf{U}\, p_1) \wedge (\neg p_6 \,\mathsf{U}\, p_5)\big) \vee \big((\neg p_6 \,\mathsf{U}\, p_3) \wedge (\neg p_6 \,\mathsf{U}\, p_2)\big)\Big),$$

whereas the median length across its five seeds was 153. For each of the four target formulas $\phi_2^\wedge, \phi_3^\wedge, \phi_2^U, \phi_3^U$, the extracted formulas were all semantically different (i.e. non-equivalent) across all five seeds and models with 1–3 layers. For the fifth target formula ϕ_{pr}, we extracted four, two, and three semantically different formulas for the models with one, two, and three layers, respectively.

To verify the agreement between each model and its extracted formula, we first observe that by Proposition 2, we can restrict our attention to words over Σ without repeated symbols only. There exist $6!/(6-k)!$ such sequences of any given length $k \leq 6$; summing for $k = 1$ through $k = 6$, we obtain a total of 1956 nonempty sequences, on which we check equivalence.

We conclude our analysis in this section by studying how the size of the extracted formulas grows with the number of layers of the transformer. Recall that unfolding replaces symbols from later layers with formulas from earlier layers, potentially producing many repeated parts. As a result, formula size grows rapidly with the number of layers; this can be seen in Table 2, where we aggregate the results across all five target formulas and their respective seed runs.

4.2 Sufficient reason extraction

We describe here our experiments extracting sufficient reasons, using the unfolding from Definition 7. We perform two experiments: in the first experiment, we extract sufficient reasons from the models trained for the synthetic datasets in Section 4.1, whereas in the second experiment, we test the scalability of our approach: we create new formulas and datasets following the templates described at the start of Section 4, we then train models for them and extract sufficient reasons from them.

Synthetic Data Experiments. For this task, we use the models from the synthetic dataset analysis from Section 4.1; in particular, we use only models trained for seed 42, for a total of 15 models (one for each of the five target formulas and 3 possible values of L). For each model, we sample 10

Table 4: Average extraction time and formula length for global extraction runs over all 3 values of L, and average extraction time and formula length over 20 sufficient reasons per target.

Target	Global time [ms]	Suff. Reason time [ms]	Global length	Suff. Reason length
ϕ_2^U	$25,502.23$	12.17	$7.9174 \cdot 10^{10}$	30.05
ϕ_3^U	$31,052.19$	10.44	$1.2513 \cdot 10^{11}$	39.05
$\phi_2^\wedge$	$32,152.52$	10.78	$2.4027 \cdot 10^{11}$	50.10
$\phi_3^\wedge$	$21,093.14$	10.90	$2.2954 \cdot 10^{10}$	30.00
ϕ_{pr}	$27,741.45$	13.66	$1.7936 \cdot 10^{10}$	40.90
Mean	$27,508.31$	11.59	$9.7064 \cdot 10^{10}$	38.02

Table 5: Larger alphabet sufficient reason extraction at $L = 3$. Each row summarizes 10 words, grouped by model prediction. Times exclude verification and formula serialization.

Target	Prediction	Mean time [s]	Median time [s]	Mean length	Median length
ψ_3^U	Positive	41.22	48.03	470.1	585.0
ψ_3^U	Negative	60.17	16.69	$7.139 \cdot 10^8$	$1,037.5$
$\psi_3^\wedge$	Positive	24.64	13.79	$8.084 \cdot 10^6$	245.0
$\psi_3^\wedge$	Negative	204.02	99.95	$4.417 \cdot 10^9$	$5.894 \cdot 10^6$

words classified positive and 10 classified negative, and extract a sufficient reason for each. Table 4 compares sufficient reason extraction with global formula extraction on the five target formulas. Global extraction times are averages over the three runs for each formula (one for each value of L). Sufficient reason times and lengths are averages over the 20 sufficient reasons. Sufficient reason extraction was substantially faster for every target. As expected, all 100 sufficient reasons were considerably smaller and had shorter times than their corresponding global formula.

Large Alphabet Experiments. We also extracted sufficient reasons from two new datasets created by fixing a new alphabet of 1000 symbols and two new formulas ψ_3^U and $\psi_3^\wedge$, constructed following the template in Equation (5), with $m = 3$. For each of these two target formulas, we create a dataset of $20,000$ words of length 1000 each, equally split into positive and negative examples obtained by sampling. We split each dataset into $14,000$ training, 3000 validation, and 3000 test examples. For this experiment, we only used seed 42 and models with $L = 3$. The balanced accuracy on test instances was 67.51% for ψ_3^U and 62.40% for $\psi_3^\wedge$. For each model, we sample 10 distinct words with 10 positive and 10 negative predictions. No global formula was extracted for these larger datasets. Table 5 presents the extraction times and formula lengths. The negative $\psi_3^\wedge$ averages are dominated by one word whose extraction took 11.20 minutes and produced a formula of length $43,017,887,694$. The substantially smaller median lengths in three of the four groups show that a few very large formulas dominate those averages. The cost of extraction depends on which symbols the model attends to when processing a word. For some words, these choices imply that certain symbols must be absent, because they would otherwise receive higher attention scores. The extractor can then skip rules involving these symbols, quickly producing shorter formulas. For others, many candidates remain and must be considered. In particular, for the outlier formula of length $43,017,887,694$, the extractor retained $9,385$ different ways of combining symbols through the first two attention layers. For another negative word under the same model, only 2 combinations remained.

5 Conclusion

We established theoretical results on the exact global encoding of a HAT (with left-most hard attention, no positional encoding, and no masking) and on the notion of a sufficient reason. The HAT restrictions simplify the transformer, yielding exact encodings amenable to implementation. We provide the first implementations of the global and sufficient reason notions into LTL$^-$. While global exact encodings are difficult to scale, sufficient reasons can be extracted for many inputs on larger alphabets. As future work, it would be interesting to explore potential further optimizations of our code and other kinds of transformer models, including those with some kind of positional encoding and masking.

AI USE STATEMENT

In this work, we used generative AI tools to support code development and to aid and polish writing; specifically, we used them to check the grammar of some sentences and/or rephrase them, and to convert results printed by the Python implementation into LaTeX tables. We have reviewed all AI-assisted content and take responsibility for the final content of this work.

REFERENCES

Marcelo Arenas, Pablo Barceló, Miguel A. Romero Orth, and Bernardo Subercaseaux. On computing probabilistic explanations for decision trees. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b8963f6a0a72e686dfa98ac3e7260f73-Abstract-Conference.html.

Dan Friedman, Alexander Wettig, and Danqi Chen. Learning transformer programs. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *NeurIPS*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/995f693b73050f90977ed2828202645c-Abstract-Conference.html.

Luca Geatti, Nicola Gigante, and Angelo Montanari. BLACK: A fast, flexible and reliable LTL satisfiability checker. In Dario Della Monica, Gian Luca Pozzato, and Enrico Scala (eds.), *Proceedings of the 3rd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis hosted by the Twelfth International Symposium on Games, Automata, Logics, and Formal Verification (GandALF)*, volume 2987 of *CEUR Workshop Proceedings*, pp. 7–12. CEUR-WS.org, 2021. URL https://ceur-ws.org/Vol-2987/paper2.pdf.

Christopher Hahn, Frederik Schmitt, Jens U. Kreber, Markus Norman Rabe, and Bernd Finkbeiner. Teaching temporal logics to neural networks. In *ICLR*. OpenReview.net, 2021. URL https://openreview.net/forum?id=dOcQK-f4byz.

Selim Jerad, Anej Svete, Jiaoda Li, and Ryan Cotterell. Unique hard attention: A tale of two sides. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 977–996. Association for Computational Linguistics, 2025. ISBN 979-8-89176-252-7. doi: 10.18653/v1/2025.acl-short.76. URL https://aclanthology.org/2025.acl-short.76/.

Jiaoda Li and Ryan Cotterell. Characterizing the Expressivity of Fixed-Precision Transformer Language Models. In *NeurIPS*, 2025. URL https://mlanthology.org/neurips/2025/li2025neurips-characterizing/.

Orna Lichtenstein, Amir Pnueli, and Lenore D. Zuck. The glory of the past. In Rohit Parikh (ed.), *Logics of Programs, Conference*, volume 193 of *Lecture Notes in Computer Science*, pp. 196–218. Springer, 1985. doi: 10.1007/3-540-15648-8_16.

Anthony W. Lin and Pablo Barcelo. The role of logic and automata in understanding transformers. In Pierre Ganty and Alessio Mansutti (eds.), *Reachability Problems*, pp. 17–30. Springer Nature Switzerland, 2026. ISBN 978-3-032-09524-4.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Andy Shih, Arthur Choi, and Adnan Darwiche. A symbolic approach to explaining bayesian network classifiers. In Jérôme Lang (ed.), *IJCAI*, pp. 5103–5111. ijcai.org, 2018. doi: 10.24963/IJCAI.2018/708. URL https://doi.org/10.24963/ijcai.2018/708.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *NeurIPS*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Andy Yang, David Chiang, and Dana Angluin. Masked hard-attention transformers recognize exactly the star-free languages. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *NeurIPS*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/13d7f172259b11b230cc5da8768abc5f-Abstract-Conference.html.

Yihao Zhang, Zeming Wei, and Meng Sun. Automata extraction from transformers. *CoRR*, abs/2406.05564, 2024. doi: 10.48550/ARXIV.2406.05564. URL https://doi.org/10.48550/arXiv.2406.05564.