Logical Languages Accepted by Transformer Encoders with Hard Attention

Pablo Barceló[1], Alexander Kozachinskiy[1], Anthony Widjaja Lin[2], and Vladimir Podolskii[3]

[1] *Institute for Mathematical and Computational Engineering, Universidad Católica de Chile & IMFD Chile & CENIA*
[2] *TU Kaiserslautern, Kaiserslautern, Germany & Max Planck Institute for Software Systems, Kaiserslautern, Germany*
[3] *Courant Institute of Mathematical Sciences, NewYorkUniversity, NY, USA & Steklov Mathematical Institute of Russian Academy of Sciences, Moscow, Russia*

Abstract

We contribute to the study of formal languages that can be recognized by transformer encoders. We focus on two self-attention mechanisms: (1) UHAT (Unique Hard Attention Transformers) and (2) AHAT (Average Hard Attention Transformers). UHAT encoders are known to recognize only languages inside the circuit complexity class AC^0, i.e., accepted by a family of poly-sized and depth-bounded boolean circuits with unbounded fan-ins. On the other hand, AHAT encoders can recognize languages outside AC^0), but their expressive power still lies within the bigger circuit complexity class TC^0, i.e., AC^0-circuits extended by majority gates. We first show a negative result that there is an AC^0-language that cannot be recognized by an UHAT encoder. On the positive side, we show that UHAT encoders can recognize a rich fragment of AC^0-languages, namely, all languages definable in first-order logic with arbitrary unary numerical predicates. This logic, includes, for example, all regular languages from AC^0. We then show that AHAT encoders can recognize all languages of our logic even when we enrich it with counting terms. We apply these results to derive new results on the expressive power of UHAT and AHAT up to permutation of letters (a.k.a. Parikh images).

1 Introduction

Transformers have revolutionized natural language processing by facilitating the efficient and effective modeling of intricate contextual relationships within text [19]. This remarkable capability has sparked numerous investigations into the potential boundaries of transformers' power [11, 22, 17, 21, 12, 6, 5, 7]. One natural method for addressing this question is to explore the classes of formal

1

languages that these architectures can recognize. This approach provides an insight into their strengths and limitations. The response to this question naturally relies on the specific features allowed within transformer encoders. These encompass the interplay between encoders and decoders, the kind of functions used for positional encodings and attention mechanisms, and considerations of fixed or unbounded precision, among other factors.

While the capacity of transformers that incorporate both encoders and decoders to recognize languages is well understood today (indeed, such architectures are Turing-complete and can thus recognize any computable language [17]), the expressive power of transformer encoders has not been fully elucidated to date. *Unique Hard Attention Transformers (UHAT)* are a class of transformer encoders that has been a subject of many recent papers. As was shown by [12], UHATs recognize only languages in AC^0, i.e., recognized by families of Boolean circuits of unbounded fan-in that have constant depth and polynomial size. Intuitively, this means that UHATs are rather weak at "counting" (more precisely, reasoning about the number of occurrences of various letters in the input word). For example, consider the following two languages: *majority* and *parity*. The first one corresponds to the set of words over alphabet $\{a, b\}$ for which the majority of positions are labeled by a, while the second checks if the number of positions labeled a is even. That these languages are not in AC^0 follows from a groundbreaking result in circuit complexity theory [9, 1]). Hence, they are neither accepted by UHATs. However, which fragment of the AC^0 languages can actually be recognized by UHATs remains an unresolved question.

We start by showing that not all AC^0 languages can be accepted by UHATs. This is obtained by combining results from [1] and [11]. Based on the previous observation, we focus on identifying a rich fragment of AC^0 that can in fact be embedded into the class of UHATs. To achieve this, we use the characterization of AC^0 as the class of languages expressible in FO(All), the extension of first-order logic (FO) with all numerical predicates defined in relation to the linear order of a word [13]. We show that UHATs recognize all languages definable in FO(Mon), the restriction of FO(All) with *unary* numerical predicates only [4]. The logic FO(Mon) is highly expressive. Unlike FO, it can express non-regular languages like $\{a^n b^n \mid n > 0\}$. Remarkably, it contains all *regular languages* within AC^0, which includes examples like $(aa)^*$ — a language not definable in FO. Additionally, our result subsumes the result of [22], where it is shown that *Dyck languages* of bounded nested depth can be recognized by UHATs. It is not hard to see that these languages are regular and belong to AC^0, hence they are expressible in FO(Mon). Our result also implies that UHAT is expressively more powerful than regular languages modulo letter-permutation (a.k.a. *Parikh images* [16, 15]).

To establish the result that UHATs recognize all languages definable in FO(Mon), we take a slightly circuitous route: rather than directly formulating FO(Mon) sentences as UHATs, we show that each formula in LTL(Mon), the extension of *linear temporal logic* (LTL) [8] with arbitrary unary numerical predicates, can be equivalently represented as an UHAT. The proof for FO(Mon) then derives from Kamp's seminal theorem [14], which establishes the equiva-

lence between languages definable in FO and LTL. The advantage of dealing with LTL, in contrast to FO, lies in the fact that all LTL formulas are unary in nature, i.e., they are interpreted as sets of positions on a word, unlike FO formulas which possess arbitrary arity. This property aligns well with the expressive capabilities of UHATs, facilitating a proof through structural induction.

While the fact that UHAT is in AC^0 implies limited counting abilities of such encoders, recent work has shown that a slight extension of the hard attention mechanism can help in recognizing languages outside AC^0 [12]. Instead of using unique hard attention, this model uses *average hard attention* (AHAT), which refers to the idea that the attention mechanism returns the uniform average value among all positions that maximize the attention. *To what extent does AHAT enrich the counting ability of UHAT?* In answering this question, we introduce a logic named $\mathrm{LTL}(\mathbf{C}, +)$, which is an extension of $\mathrm{LTL}(\mathsf{Mon})$ that naturally incorporates counting features. We show that any language that can be defined within $\mathrm{LTL}(\mathbf{C}, +)$ can also be identified by an AHAT. The logic $\mathrm{LTL}(\mathbf{C}, +)$ can express interesting languages lying outside AC^0 including majority and parity (as far as we know, it have been shown before that parity can be accepted by an AHAT). More generally, our result implies that AHATs are equipped with a powerful counting ability: all permutation-closed languages over a binary alphabet and all permutation closures of regular languages (which are in general not context-free) can be recognized by AHATs.

Related work. There has been very little research on identifying logical languages that can be accepted by transformers. The only example we are aware of is the recent work by [7], in which a variant of first-order logic with counting quantifiers is demonstrated to be embeddable into transformer encoders with a *soft attention* mechanism. The primary distinction between their work and our results is the choice of the attention mechanism. Additionally, the logic examined in their paper does not have access to the underlying word order being considered. This implies that some simple languages, such as $a^* b^*$, which are definable in FO, are not definable in their logic.

Proviso. Some of the proofs in the paper are rather technical and lengthy. For such a reason we have relegated them to the appendix.

2 Background notions and results

2.1 Transformer encoders

We utilize a streamlined version of transformers, simplifying the model by abstracting certain features employed in more real-world scenarios.

An *encoder layer* is a function that takes a sequence of vectors, $\mathbf{v}_0, \ldots, \mathbf{v}_{n-1}$, in $\mathbb{R}^d$ as input, where $d \geq 0$. It produces an output sequence of vectors, $\mathbf{v}'_0, \ldots, \mathbf{v}'_{n-1}$, in $\mathbb{R}^e$, with $e \geq 0$. We consider two types of encoder layers:

standard and *ReLU*. Standard encoder layers resemble those found in most formalizations of transformer encoders. For the first part of the paper we assume that they employ a *unique hard* attention mechanism, meaning that a position only attends to the element with the highest attention score (breaking ties arbitrarily). On the other hand, ReLU encoder layers simply apply a ReLU function to the kth coordinate of each vector $\mathbf{v}_i$. ReLU layers serve as a practical method for encoding logical formulas into transformers. A *transformer encoder* is then a concatenation of encoder layers. We define all these notions below.

Standard encoder layer with unique hard attention. A standard encoder layer is defined by three affine transformations, $A, B \colon \mathbb{R}^d \to \mathbb{R}^d$ and $C \colon \mathbb{R}^{2d} \to \mathbb{R}^e$. For $i \in \{0, \ldots, n-1\}$, we set

$$\mathbf{a}_i \leftarrow \mathbf{v}_{j_i},$$

where $j_i \in \{0, \ldots, n-1\}$ is the minimum element that maximizes the *attention score* $\langle A\mathbf{v}_i, B\mathbf{v}_j \rangle$ over $j \in \{0, \ldots, n-1\}$. The a_is are often known as *attention vectors*. After that, we set

$$\mathbf{v}'_i \leftarrow C(\mathbf{v}_i, \mathbf{a}_i), \qquad i = 0, \ldots, n-1.$$

It is useful to note that standard layers can do arbitrary position-wise affine transformations.

ReLU encoder layer. A ReLU layer is given by $k \in \{1, 2, \ldots, d\}$. It just applies the ReLU function to the kth coordinate of each vector $\mathbf{v}_i$. That is, assuming that $\mathbf{v}_i = (v_i^1, \ldots, v_i^d)$, then $\mathbf{v}'_i \leftarrow (v_i^1, \ldots, v_i^{k-1}, \max\{0, v_i^k\}, v_i^{k+1}, \ldots, v_i^d)$, for $i = 0, \ldots, n-1$. The ReLU function can express the max of two numbers: $\max(x, y) = \max(0, x-y) + y$. This shows that with a constant number of ReLU layers, we can implement position-wise any function which is a composition of affine transformations and max.

Transformer encoder. A *unique hard attention transformer encoder* (UHAT)[1] is defined simply as the repeated application of standard encoder layers with unique hard attention and ReLU encoder layers (with independent parameters).

2.2 Languages accepted by transformer encoders

Next, we define how a transformer can be used to accept languages over a finite alphabet. This requires extending transformer encoders with three features: a function for representing alphabet symbols as vectors (which, for the purposes of this paper, we represent as one-hot encodings), another function

---

[1] Some of the previous papers, for instance [12], allow to use in UHAT only rational numbers. We find this too restrictive because functions such as cos and sin are widely used in practice. Nevertheless, we stress that our results hold with this restriction, by taking good-enough approximations by rational numbers.

that provides information about the absolute positions of these symbols within the input word, and a vector that is used for checking whether the word should be accepted or not. The function that provides information about positions is often referred to as a *positional encoding*, and it is essential for recognizing properties of ordered sequences of vectors. In fact, without positional encoding, encoders treat input sequences as invariant to permutations [17].

Consider a finite alphabet Σ and let T be an UHAT that takes a sequence of vectors over $\mathbb{R}^d$ as input and converts it into a sequence of vectors over $\mathbb{R}^e$. A language $L \subseteq \Sigma^+$ is *accepted* by T, if there is an embedding function $f \colon \Sigma \to \mathbb{R}^d$, a positional encoding function $p \colon \mathbb{N} \times \mathbb{N} \to \mathbb{R}^d$, and a vector $\mathbf{t} \in \mathbb{R}^e$, such that for every $\bar{w} \in L$ we have $T(\bar{w}) > 0$, and for every $w \in \Sigma^+ \setminus L$ we have $T(\bar{w}) < 0$. Here, $T \colon \Sigma^+ \to \mathbb{R}$ is defined as follows. Let $\bar{w} = a_0 \ldots a_{n-1} \in \Sigma^n$, and suppose the output of T when given the input sequence $f(a_0) + p(0, n), \ldots, f(a_{n-1}) + p(n-1, n)$ is the sequence $\mathbf{v}_0, \ldots, \mathbf{v}_{n-1}$. Then we set $T(\bar{w}) = \langle \mathbf{t}, \mathbf{v}_0 \rangle$.

2.3 First order logic on words

We assume familiarity with first-order logic (FO). Let Σ be a finite alphabet. A word $\bar{w} = a_0 \cdots a_{n-1}$ in Σ^+ is represented as a structure $S_{\bar{w}}$ whose domain is $\{0, \ldots, n-1\}$. This structure includes a binary relation $<$ that is interpreted as the linear order on the domain, and for each symbol $a \in \Sigma$, there is a unary relation P_a containing positions $i = 0, \ldots, n-1$ where $a_i = a$. Given an FO *sentence* over words, that is, an FO formula without free variables, we denote the language of all words $\bar{w} \in \Sigma^+$ satisfying $S_{\bar{w}} \models \phi$ as $L(\phi)$. If an $L \subseteq \Sigma^+$ satisfies $L = L(\phi)$, for some FO sentence ϕ, then we say that L is *definable in* FO.

Example 1. First-order logic (FO) enables us to define certain languages of interest. Here, we present an illustrative example. Initially, we recognize that we can employ FO to define a relation $\mathsf{first}(x) := \neg \exists y (y < x)$ that exclusively holds true at the first position of a word. Correspondingly, we can define a relation $\mathsf{last}(x) := \neg \exists y (x < y)$ that holds solely at the last position of the word. Moreover, it is possible to define a binary relation $\mathsf{succ}(x, y) := x < y \land \neg \exists z (x < z \land z < y)$, which defines the successor relation within the domain. With these expressions, we can show that FO is capable of defining the language $(ab)^+$:

$$\exists x \left(\mathsf{first}(x) \land P_a(x) \right) \land \exists x \left(\mathsf{last}(x) \land P_b(x) \right) \land \forall x \forall y \left(\mathsf{succ}(x, y) \to (P_a(x) \leftrightarrow P_b(y)) \right).$$

That is, the first symbol of the word is an a, the last one is a b, every a is followed by a b, and every b is preceded by an a. $\quad\blacksquare$

2.4 Unary numerical predicates

It is known that FO sentences can only define regular languages. In turn, there are regular languages that are not definable in FO. An example is the language $(aa)^*$, which contains those words formed solely by the symbol a that are of even length. However, there is a straightforward extension of FO that can

5

define this language: all we need to do is add unary predicate $\mathsf{even}(x)$, which holds true at position i in a word if and only if i is even. In fact, extending FO with the predicate $\mathsf{even}(x)$ allows us to define the language $(aa)^*$ using the following formula, which indicates that the last symbol in the word satisfies the unary predicate even: $\forall x P_a(x) \wedge \forall y(\mathsf{last}(y) \rightarrow \mathsf{even}(y))$.

The extension of FO with unary numerical predicates can then be useful for defining languages. We define a *unary numerical predicate* Θ as an infinite family of functions

$$\theta_n : \{0, \ldots, n\} \rightarrow \{0, 1\}, \qquad n > 0.$$

Given a word $\bar{w}$ in Σ^+ of length n, for $n > 0$, we have that the predicate $\Theta(x)$ holds in position i in $\bar{w}$ if and only if $\theta_n(i) = 1$ (so far, we do not use the value of θ_n at n as positions are numbered from 0 to $n - 1$. We will use this value in Section 4). Notice that under our definition, the truth of a unary numerical predicate at position i in the word $\bar{w}$ depends not only on i but also on the length of the word $\bar{w}$. As we will explore further, this characteristic is advantageous for defining interesting languages in FO extended with arbitrary unary numerical predicates. Following the literature, we write FO(Mon) for such an extension [4].

Example 2. Consider, for example, the non-regular language $\{a^n b^n \mid n > 0\}$. We show that it can be expressed in FO(Mon) with the help of a unary numerical predicate $\Theta(x)$ such that $\theta_n(i) = 1$ iff n is even and $i = n/2 - 1$. In fact, it suffices to use the formula:

$$\exists x \left(\Theta(x) \wedge P_a(x) \wedge \forall y(y < x \rightarrow P_a(y)) \wedge \forall y(x < y \rightarrow P_b(y)) \right).$$

This formula expresses that the middle point i of $\bar{w}$ exists, is labeled as a, and all positions smaller than i are also labeled a, while all positions larger than i are labeled as b. This example illustrates the significance of unary numerical predicates depending on both the position and the length of the word over which the formula is evaluated. $\square$

The definition of the language $L(\phi) \subseteq \Sigma^+$ defined by an FO(Mon) sentence ϕ is analogous to the one we provided for FO.

3 AC^0 languages accepted by UHATs

3.1 Not all languages in AC^0 are accepted by UHATs.

[12] proved that languages accepted by UHATs belong to the circuit complexity class AC^0 , i.e., the class of languages accepted by families of Boolean circuits of unbounded fan-in, constant depth, and polynomial size. We combine results by [1] and [11] to show that the opposite is not the case, i.e., there are AC^0 languages that are not accepted by UHATs.

As shown in [1], there is an AC^0-family of circuits $\{C_n \colon \{0,1\}^n \rightarrow \{0,1\}\}_{n \in \mathbb{N}}$ such that for all n, the circuit C_n accepts all strings with at at least $2n/3$ ones

and rejects all strings with at most $n/3$. Consider a language *approximate majority*, consisting of strings accepted by circuits from $\{C_n\}$. This language is in AC^0 by construction. However, as we state next, it cannot be recognized by an UHAT. This result is proved by using a property of UHATs established in [11].

Proposition 1. *There is no UHAT that accepts the language* approximate majority*.*

[20] shows that $\{C_n\}$ can be made polynomial-time computable, which implies the existence of a *polynomial-time computable* language from AC^0 that cannot be accepted by an UHAT.

3.2   Main result: $\mathrm{FO}(\mathsf{Mon})$ languages are accepted by UHATs

Proposition 1 tells us that not all AC^0 languages are accepted by UHATs. In this section, we identify a significant subset of AC^0 languages that can be accepted by UHATs. To accomplish this, we rely on the characterization of the class AC^0 as those languages that can be defined in FO extended with arbitrary numerical predicates. Our main result establishes that as long as we restrict ourselves to unary numerical predicates, translation into UHATs is possible.

Theorem 1. *Let Σ be a finite alphabet and ϕ an $\mathrm{FO}(\mathsf{Mon})$ sentence over words from the alphabet Σ. There is an UHAT that accepts $L(\phi)$.*

Proving this result by induction on $\mathrm{FO}(\mathsf{Mon})$ formulas, which would be the most natural approach to tackle the problem, turns out to be difficult. The challenge arises because the $\mathrm{FO}(\mathsf{Mon})$ formulas obtained by induction can have arbitrary arity, and transformer encoders do not seem capable of handling the requirements imposed by such formulas. To address this issue, we take a different approach. We employ Kamp's Theorem, which establishes that the languages definable in FO are precisely those that are definable in *linear temporal logic* (LTL) [14].

3.3   Using $\mathrm{LTL}(\mathsf{Mon})$ to prove our main result

We first explain how LTL is defined, as this is crucial to understanding the remainder of the paper. Let Σ be a finite alphabet. LTL formulas over Σ are defined as follows: if $a \in \Sigma$, then a is an LTL formula. Additionally, LTL formulas are closed under Boolean combinations. Finally, if ϕ and ψ are LTL formulas, then $\mathbf{X}\phi$ and $\phi\mathbf{U}\psi$ are also LTL formulas. Here, $\mathbf{X}$ is referred to as the *next* operator, and $\mathbf{U}$ as the *until* operator.

LTL formulas are unary, i.e., they are evaluated over positions within a word. Let $\bar{w} = a_0 \cdots a_{n-1}$ be a word in Σ^+, and let $i = 0, \ldots, n-1$. We define the satisfaction of an LTL formula ϕ over $\bar{w}$ at position i, written as $(\bar{w}, i) \models \phi$, inductively as follows (omitting Boolean combinations):

- $(\bar{w}, i) \models a$ if and only if $a = a_i$, for $a \in \Sigma$.

- $(\bar{w}, i) \models \mathbf{X}\phi$ if and only if $i < n - 1$ and $(\bar{w}, i+1) \models \phi$. In other words, ϕ holds in the next position after i (if such a position exists).

- $(\bar{w}, i) \models \phi\mathbf{U}\psi$ if and only if there exists a position $j = i, \ldots, n - 1$ for which $(\bar{w}, j) \models \psi$ and such that $(\bar{w}, k) \models \phi$ for every k with $i \leq k < j$. That is, ϕ holds starting from position i until the first position where ψ holds (and a position where ψ holds must exist).

We can extend LTL with unary numerical predicates in the same way we did it for FO. Formally, we define LTL(Mon) as the extension of LTL with every formula of the form Θ, for Θ a unary numerical predicate. We write $(\bar{w}, i) \models \Theta$ to denote that $\theta_n(i) = 1$, where n is the length of $\bar{w}$. If ϕ is an LTL(Mon) formula over Σ, we write $L(\phi)$ for the set of words $\bar{w} \in \Sigma^+$ with $(\bar{w}, 0) \models \phi$.

Kamp's Theorem establishes that for every FO sentence ϕ there exists an LTL formula ψ such that $L(\phi) = L(\psi)$, and vice-versa. It is straightforward to see that this property extends to the logics FO(Mon) and LTL(Mon).

Proposition 2. *[14] For every* FO(Mon) *sentence ϕ there exists an* LTL(Mon) *formula ψ such that $L(\phi) = L(\psi)$, and vice-versa.*

Our proof of Theorem 1 is then derived directly from Proposition 2 and the following result.

Proposition 3. *Let Σ be a finite alphabet and ϕ an* LTL(Mon) *formula defined over words from the alphabet Σ. There is an UHAT T that accepts $L(\phi)$.*

Before proving this result, we make the following important remark regarding the positional encoding p used by T to accept $L(\phi)$. On a pair $(i, n) \in \mathbb{N} \times \mathbb{N}$ with $i < n$, we have that $p(i, n)$ is composed of elements i, $1/(i+1)$, $(-1)^i$, $\cos\left(\pi(1-2^{-i})/10\right)$, $\sin\left(\pi(1-2^{-i})/10\right)$, and $\theta_n(i)$, for every unary numerical predicate Θ mentioned in ϕ.

Proof of Proposition 3. Let ϕ be a formula of LTL(Mon). We say that a UHAT *realizes ϕ position-wise* if, given a word $\bar{w} = a_0 \ldots a_{n-1} \in \Sigma^+$, the UHAT outputs a sequence:

$$\mathbb{I}\{(\bar{w}, 0) \models \phi\}, \; \mathbb{I}\{(\bar{w}, 1) \models \phi\}, \; \ldots \; , \; \mathbb{I}\{(\bar{w}, n-1) \models \phi\};$$

that is, a binary word indicating for which positions ϕ is true on $\bar{w}$ and for which is false. We show by structural induction that every LTL(Mon) formula is realizable position-wise by some UHAT.

Let us consider first the base cases. If $\phi = a$, for some $a \in \Sigma$, our goal is to obtain a sequence:

$$\mathbb{I}\{a_0 = a\}, \; \mathbb{I}\{a_1 = a\}, \; \ldots \; , \; \mathbb{I}\{a_{n-1} = a\}.$$

This can easily be achieved by using a one-hot encoding as the embedding function. In turn, if $\phi = \Theta$, for Θ a unary numerical predicate, then ϕ can be realized position-wise using the corresponding positional encoding $p(i, n) = \theta_n(i)$.

8

We continue with Boolean combinations. They can be implemented with a composition of ReLU layers and point-wise affine transformation: $\neg x = 1 - x$ and $x \vee y = \frac{\max\{2x-1, 2y-1\}+1}{2}$.

For the cases when our formula is of the form $\mathbf{X}\phi$ or $\phi\mathbf{U}\psi$, we need the following lemma.

Lemma 1. *There is an UHAT that transforms each $x_0, \ldots, x_{n-1} \in \{0,1\}$ as follows:*

$$x_0, \ldots, x_{n-2}, x_{n-1} \mapsto x_0, \ldots, x_{n-2}, 0.$$

Let us assume now that our formula is of the form $\mathbf{X}\phi$. It is enough to design a unique hard attention layer in which attention is always maximized at the next position. More precisely, we construct an UHAT that outputs a sequence of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^3$, and a linear transformation $A \colon \mathbb{R}^3 \to \mathbb{R}^3$, such that $\arg\max_{j\in\mathbb{N}}\langle A\mathbf{v}_i, \mathbf{v}_j\rangle = \{i+1\}$, for $i = 0, \ldots, n-2$. This will allow us to "send" $\mathbb{I}\{(\bar{w}, i+1) \models \phi\} = \mathbb{I}\{(\bar{w}, i) \models \mathbf{X}\phi\}$ to the ith position, for $i = 0, \ldots, n-2$. It only remains then to apply Lemma 1 to obtain $0 = \mathbb{I}\{(\bar{w}, n-1) \models \mathbf{X}\phi\}$ at the last position.

Using our positional encoding and an affine position-wise transformation, we can obtain:

$$\mathbf{v}_i = \left(\cos\left(\frac{\pi(1-2^{-i})}{10}\right), \ \sin\left(\frac{\pi(1-2^{-i})}{10}\right), \ (-1)^i \cdot 10 \right).$$

Let A be a linear transformation that reverses the third coordinate. Observe that:

$$\langle A\mathbf{v}_i, \mathbf{v}_j\rangle = \cos\left(\frac{\pi(2^{-i} - 2^{-j})}{10}\right) + (-1)^{i+j+1} \cdot 10.$$

We claim that, for a fixed i, this quantity is maximized at $j = i+1$. First, those js that have the same parity as i (in particular, $j = i$) cannot achieve the maximum because the second term is -10. For js with a different parity, we have $\langle A\mathbf{v}_i, \mathbf{v}_j\rangle = \cos\left(\pi(2^{-i} - 2^{-j})/10\right) + 10$. Since all angles are in $[-\pi/10, \pi/10]$, this quantity is maximized when $|2^{-i} - 2^{-j}|$ is minimized. For $j < i$, the last quantity is at least 2^{-i}, and for $j > i$, the minimum of this quantity is 2^{-i-1}, achieved at $j = i+1$.

Let us finally assume that our formula is of the form $\phi\mathbf{U}\psi$. For a given $i = 0, \ldots, n-1$, let j_i be the minimal $j \in \{i, \ldots, n-1\}$ such that $(\bar{w}, j) \not\models \phi$, and if no such j exists, $j_i = n-1$. Observe that $(\bar{w}, i) \models \phi\mathbf{U}\psi$ if and only if $(\bar{w}, j_i) \models \psi$. To show the lemma, it is enough to create a unique hard attention layer, where for every position i the attention is maximized at j_i.

Due to the Lemma 1, we may assume, without loss of generality, that $(\bar{w}, n-1) \not\models \phi$. Then for every i, there exists at least one $j \in \{i, \ldots, n-1\}$ such that $(\bar{w}, j) \not\models \phi$, and then j_i can be defined as the minimal such j, without any clauses.

Using our positional encoding and the induction hypothesis, we can obtain a sequence of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^4$ such that:

$$\mathbf{v}_i = \left(\cos\left(\frac{\pi(1-2^{-i})}{10}\right), \ \sin\left(\frac{\pi(1-2^{-i})}{10}\right), \ 1, \ \mathbb{I}\{w, i \models \phi\} \right).$$

Consider a linear transformation $B \colon \mathbb{R}^4 \to \mathbb{R}^4$ such that

$$B\mathbf{v}_i = \left(\cos\left(\frac{\pi(1 - 2^{-i})}{10} \right), \ \sin\left(\frac{\pi(1 - 2^{-i})}{10} \right), \ -10 \cdot \mathbb{I}\{w, i \models \phi\}, \ 0 \right).$$

Observe that

$$\langle \mathbf{v}_i, B\mathbf{v}_j \rangle = \cos\left(\frac{\pi(2^{-i} - 2^{-j})}{10} \right) - 10 \cdot \mathbb{I}\{\bar{w}, j \models \phi\}.$$

We claim that this expression is maximized at $j = j_i$. First, because of the last term in it, it cannot be maximized at j with $(\bar{w}, j) \models \phi$. It remains to show that among the js with $(\bar{w}, j) \not\models \phi$, this quantity is minimized on the minimal j which is at least i. In fact, in this case we have $\langle \mathbf{v}_i, B\mathbf{v}_j \rangle = \cos\left(\frac{\pi(2^{-i} - 2^{-j})}{10} \right)$. All the angles in question are in $[-\pi/10, \pi/10]$, so the cosine is maximized when $|2^{-i} - 2^{-j}|$ is minimized. Now, this absolute value is at least 2^{-i} when $j < i$. In turn, this absolute value is smaller than 2^{-i} for $j \geq i$, and it is the smaller the smaller is j, as required. $\qquad\square$

3.4 Applications of our main result

We show two applications of our main result. First, UHATs accept all regular languages in AC^0. Second, UHATs are strictly more expressive than regular and context-free languages in terms of the acceptance of languages up to letter-permutation.

Regular languages in AC^0. There is an important fragment of FO(Mon) which is interesting in its own right. This is the logic FO(Mod), i.e., the extension of FO with unary numerical predicates of the form Mod_p^r, for $p > 1$ and $0 \leq r \leq p-1$. We have that $\mathsf{Mod}_p^r(i) = 1$ if and only if $i \equiv r \,(\mathrm{mod}\, p)$. In fact, by using a characterization given in [3], one can show that the languages definable in FO(Mod) are precisely the regular languages within AC^0. Then:

Corollary 1. *Let $L \subseteq \Sigma^+$ be a regular language in AC^0. There is an UHAT that accepts L.*

Recognizing regular languages up to letter-permutation. Although not all regular languages are accepted by UHATs (e.g. *parity*), we can use Theorem 1 to show that, up to letter-permutation, UHAT is in fact strictly more powerful than regular and context-free languages.

To formalize our result, we recall the notion of semilinear sets and the Parikh image of a language. A *linear set* S is a subset of $\mathbb{N}^d$ (for some positive integer d, called *dimension*) of the form

$$\mathbf{v}_0 + \sum_{i=1}^{r} \mathbf{v}_i \mathbb{N} \ := \ \{\mathbf{v}_0 + \sum_{i=1}^{r} k_i \mathbf{v}_i : k_1, \dots, k_r \in \mathbb{N}\}$$

10

for some vectors $\mathbf{v}_0, \ldots, \mathbf{v}_r \in \mathbb{N}^d$. A *semilinear set* S over $\mathbb{N}^d$ is a finite union of linear sets over $\mathbb{N}^d$. Semilinear sets have a very tight connection to formal languages through the notion of the *Parikh image* a language L [16], which intuitively corresponds to the set of "letter-counts" of L. More precisely, consider the alphabet $\Sigma = \{a_1, \ldots, a_d\}$ and a language L over Σ. For a word $w \in \Sigma$, let $|w|_{a_i}$ denotes the number of occurrences of a_i in w. The *Parikh image* $\mathcal{P}(L)$ of L is defined to be the set of tuples $\mathbf{v} = (|w|_{a_1}, \ldots, |w|_{a_d}) \in \mathbb{N}^d$ for some word $w \in L$. For example, if $L = \{a^n b^n : n \geq 0\}$ and $L' = (ab)^*$, then $\mathcal{P}(L) = \mathcal{P}(L')$. In this case, we say that L and L' are *Parikh-equivalent*. Note that L' is regular, while L is context-free but not regular. This is not a coincidence based on the celebrated Parikh's Theorem (cf. [16], also see [15]).

Proposition 4 ([16])**.** *The Parikh images of both regular and context-free languages coincide with semilinear sets.*

In other words, although context-free languages are strict superset of regular languages, they are in fact equally powerful up to letter-permutation. What about UHATs? We have that they are strictly more powerful than regular and context-free languages up to letter-permutation.

Proposition 5. *Each regular language has a Parikh-equivalent language accepted by an UHAT. In turn, there is an UHAT language with no Parikh-equivalent regular language.*

4   Languages beyond AC^0

Transformer encoders with unique hard attention can only recognize languages in AC^0, but a slight extension of the attention mechanism allows to recognize languages lying outside such a class [12]. In this section, we show that in fact such an extended model can recognize all languages definable in a powerful logic that extends LTL with counting features. This logic can express interesting languages outside AC^0, such as *majority* and *parity*.

4.1   Average hard attention

For the results in this section, we consider an extended version of transformer encoders that utilize an *average hard attention mechanism* [17, 12]. Following the literature, we call these AHAT. The difference between UHAT and AHAT only lies at the level of the standard encoder layers, which are now defined as follows.

Standard encoder layer with average hard attention.   As before, these layers are defined by three affine transformations, $A, B\colon \mathbb{R}^d \to \mathbb{R}^d$ and $C\colon \mathbb{R}^{2d} \to \mathbb{R}^e$. For every $i \in \{0, \ldots, n-1\}$, we define S_i as the set of positions $j \in \{0, \ldots, n-1\}$ that maximize $\langle A\mathbf{v}_i, B\mathbf{v}_j \rangle$. We then set

$$\mathbf{a}_i \;\leftarrow\; \Big(\sum_{j \in S_i} \mathbf{v}_j \Big)/|S_i|.$$

After that, we set $\mathbf{v}'_i \leftarrow C(\mathbf{v}_i, \mathbf{a}_i)$, for each $i = 0, \ldots, n-1$. That is, attention scores under average hard attention return the uniform average value among all positions that maximize attention.

We also use *future positional masking* that allows us to take into account only positions up to i. If the future positional masking is used, the sets S_i are defined as sets of positions $j \in \{0, 1, \ldots, i\}$ that maximize $\langle A\mathbf{v}_i, B\mathbf{v}_j \rangle$. Positional masks have been employed on several occasions in theoretical papers [22, 5, 12] as well as in practice, for example, for training GPT-2 [18].

4.2 LTL extended with counting terms

We present here LTL($\mathbf{C}, +$), an extension of LTL(Mon) that allows us to define counting properties over words in a simple manner. This requires the introduction of *counting terms* as defined next.

Counting terms. Suppose ϕ is a unary formula. Then $\overleftarrow{\#\phi}$ and $\overrightarrow{\#\phi}$ are counting terms. The interpretation of these terms in position i of a word $\bar{w}$ of length n is defined as follows:

$$\overleftarrow{\#\phi}(\bar{w}, i) \ = \ |\{j \in \{0, \ldots, i\} \mid (\bar{w}, j) \models \phi\}|,$$
$$\overrightarrow{\#\phi}(\bar{w}, i) \ = \ |\{j \in \{i, \ldots, n-1\} \mid (\bar{w}, j) \models \phi\}|.$$

That is, $\overleftarrow{\#\phi}(\bar{w}, i)$ is the number of positions to the left of i (including i) that satisfy ϕ, while $\overrightarrow{\#\phi}(\bar{w}, i)$ is the number of positions to the right of i (including i) that satisfy ϕ. Notice that, for words of length n, counting terms take values in $\{0, 1, \ldots, n\}$.

Counting formulas. With counting terms and unary numerical predicates we can create new formulas in the following way. Let ϕ be a unary formula and Θ a unary numerical predicate. We define new formulas $\Theta(\overleftarrow{\#\phi})$ and $\Theta(\overrightarrow{\#\phi})$. The interpretation of such formulas on position i of a word $\bar{w}$ of length n is as follows:

$$(\bar{w}, i) \models \Theta(\overleftarrow{\#\phi}) \ \Leftrightarrow \ \theta_n(\overleftarrow{\#\phi}(\bar{w}, i)) = 1 \qquad (\bar{w}, i) \models \Theta(\overrightarrow{\#\phi}) \ \Leftrightarrow \ \theta_n(\overrightarrow{\#\phi}(\bar{w}, i)) = 1.$$

That is, the number of positions to the left (resp., right) of i (including i) that satisfy ϕ satisfies the predicate Θ. As counting terms can take value n, the value of θ_n on n becomes useful.

We also incorporate into our logic the possibility of checking linear inequalities with integer coefficients over counting terms. More specifically, for any finite set of unary formulas $\phi_1, \ldots, \phi_k, \psi_1, \ldots, \psi_k$, and for any coefficients $c_1, \ldots, c_k, d_1, \ldots, d_k \in \mathbb{Z}$ we can create a formula:

$$\sum_{j=1}^{k} c_j \cdot \overleftarrow{\#\phi_j} \ + \ \sum_{j=1}^{k} d_j \cdot \overrightarrow{\#\psi_j} \ \geq \ 0,$$

12

which is interpreted as follows:

$$(\bar{w}, i) \models \sum_{j=1}^{k} c_j \cdot \overleftarrow{\#\phi_j} + \sum_{j=1}^{k} d_j \cdot \overrightarrow{\#\psi_j} \geq 0 \iff$$

$$\sum_{j=1}^{k} c_j \cdot \overleftarrow{\#\phi_j}(\bar{w}, i) + \sum_{j=1}^{k} d_j \cdot \overrightarrow{\#\psi_j}(\bar{w}, i) \geq 0.$$

The logic $\mathrm{LTL}(\mathbf{C}, +)$**.** We denote by $\mathrm{LTL}(\mathbf{C}, +)$ the logic that is recursively defined as follows:

- Every formula $\mathrm{LTL}(\mathsf{Mon})$ is also an $\mathrm{LTL}(\mathbf{C}, +)$ formula.

- Boolean combinations of $\mathrm{LTL}(\mathbf{C}, +)$ formulas are $\mathrm{LTL}(\mathbf{C}, +)$ formulas.

- If ϕ and ψ are $\mathrm{LTL}(\mathbf{C}, +)$ formulas, then so are $\mathbf{X}\phi$ and $\phi\mathbf{U}\psi$.

- If ϕ is an $\mathrm{LTL}(\mathbf{C}, +)$ formula and Θ is a unary numerical predicate, then $\Theta(\overleftarrow{\#\phi})$ and $\Theta(\overrightarrow{\#\phi})$ are $\mathrm{LTL}(\mathbf{C}, +)$ formulas.

- If $\phi_1, \ldots, \phi_k, \psi_1, \ldots, \psi_k$ are formulas of $\mathrm{LTL}(\mathbf{C}, +)$, then $\sum_{j=1}^{k} c_j \cdot \overleftarrow{\#\phi_j} + \sum_{j=1}^{k} d_j \cdot \overrightarrow{\#\psi_j} \geq 0$, is a formula of $\mathrm{LTL}(\mathbf{C}, +)$.

4.3 $\mathrm{LTL}(\mathbf{C})$ definable languages are accepted by encoders

Next, we state the main result of this section: languages definable by $\mathrm{LTL}(\mathbf{C}, +)$ formulas are accepted by transformer encoders with average hard attention.

Theorem 2. *Let Σ be a finite alphabet and ϕ an $\mathrm{LTL}(\mathbf{C}, +)$ formula defined over words from the alphabet Σ. There is an AHAT T that accepts $L(\phi)$.*

As a corollary to Theorem 2, we show that AHATs are rather powerful in counting. To make this claim more formal, we study *permutation-closed* languages, i.e., languages L such that $\bar{v} \in L$ iff any letter-permutation of $\bar{v}$ is in L. For a language L, we write $perm(L)$ to be the permutation-closure of L, i.e., $perm(L) = \{\bar{w} : \mathcal{P}(\bar{w}) = \mathcal{P}(\bar{v}), \text{ for some } \bar{v} \in L\}$. Observe that $perm((abc)^*)$ consists of all strings with the same number of occurrences of a, b, and c; this is not even context-free. Owing to Parikh's Theorem, to recognize $perm(L)$, where L is a regular language, an ability to perform letter-counting and linear arithmetic reasoning (i.e. semilinear set reasoning) is necessary. AHATs possess such an ability, as shown by the following corollary.

Corollary 2. *The permutation closure $perm(L)$ of any regular language L is accepted by an AHAT. Moreover, any permutation-closed language over a binary alphabet is accepted by an AHAT.*

Both *majority* and *parity* are permutation-closed and are over a binary alphabet. Hence, by the previous result, they are both accepted by AHATs. While for *majority* this was known [12], the result for *parity* is new.

5 Conclusions and future work

We have conducted an investigation of the problem of which languages can be accepted by transformer encoders with hard attention. For UHATs, we have demonstrated that while they cannot accept all languages in AC^0, they can still accept all languages in a 'monadic' version of it defined by the logic FO(Mon). Crucial to the proof of this result is the equivalence between FO and LTL, as provided by Kamp's Theorem. In turn, we have shown that AHATs are capable of expressing any language definable in a powerful counting logic, $\mathrm{LTL}(\mathbf{C},+)$, that can express properties beyond AC^0. This implies, among other things, that the *parity* language can be accepted by an AHAT.

Several interesting problems remain open in our work, especially regarding characterizations of the classes we have studied. To begin, are there languages accepted by UHATs that cannot be defined in FO(Mon)? Additionally, does there exist a language in the circuit complexity class TC^0, the extension of AC^0 with majority gates, that cannot be recognized by AHATs? Lastly, is there a language that can be accepted by an AHAT but cannot be defined in $\mathrm{LTL}(\mathbf{C},+)$?

References

[1] AJTAI, M. $\sum_1^1$-formulae on finite structures. *Ann. Pure Appl. Log. 24*, 1 (1983), 1–48.

[2] ANDERTON, H. *A Mathematical Introduction to Logic*, 2 ed. Academic Press, 2001.

[3] BARRINGTON, D. A. M., COMPTON, K. J., STRAUBING, H., AND THÉRIEN, D. Regular languages in nc^1. *J. Comput. Syst. Sci. 44*, 3 (1992), 478–499.

[4] BARRINGTON, D. A. M., IMMERMAN, N., LAUTEMANN, C., SCHWEIKARDT, N., AND THÉRIEN, D. First-order expressibility of languages with neutral letters or: The crane beach conjecture. *J. Comput. Syst. Sci. 70*, 2 (2005), 101–127.

[5] BHATTAMISHRA, S., AHUJA, K., AND GOYAL, N. On the ability and limitations of transformers to recognize formal languages. In *EMNLP* (2020), pp. 7096–7116.

[6] CHIANG, D., AND CHOLAK, P. Overcoming a theoretical limitation of self-attention. In *ACL* (2022), pp. 7654–7664.

[7] CHIANG, D., CHOLAK, P., AND PILLAY, A. Tighter bounds on the expressivity of transformer encoders. In *ICML* (2023), vol. 202, PMLR, pp. 5544–5562.

[8] CLARKE, E. M., GRUMBERG, O., KROENING, D., PELED, D. A., AND VEITH, H. *Model checking, 2nd Edition*. MIT Press, 2018.

[9] FURST, M. L., SAXE, J. B., AND SIPSER, M. Parity, circuits, and the polynomial-time hierarchy. In *FOCS* (1981), IEEE Computer Society, pp. 260–270.

[10] HAASE, C. A survival guide to presburger arithmetic. *ACM SIGLOG News 5*, 3 (2018), 67–82.

[11] HAHN, M. Theoretical limitations of self-attention in neural sequence models. *Trans. Assoc. Comput. Linguistics 8* (2020), 156–171.

[12] HAO, Y., ANGLUIN, D., AND FRANK, R. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Trans. Assoc. Comput. Linguistics 10* (2022), 800–810.

[13] IMMERMAN, N. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

[14] KAMP, H. W. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.

[15] KOZEN, D. C. *Automata and Computability*. Springer, 1997.

[16] PARIKH, R. On context-free languages. *J. ACM 13*, 4 (1966), 570–581.

[17] PÉREZ, J., BARCELÓ, P., AND MARINKOVIC, J. Attention is turing-complete. *J. Mach. Learn. Res. 22* (2021), 75:1–75:35.

[18] RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D., SUTSKEVER, I., ET AL. Language models are unsupervised multitask learners. *OpenAI blog 1*, 8 (2019), 9.

[19] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention is all you need. In *NeurIPS* (2017), pp. 5998–6008.

[20] VIOLA, E. On approximate majority and probabilistic time. *Computational Complexity 18* (2009), 337–375.

[21] WEISS, G., GOLDBERG, Y., AND YAHAV, E. Thinking like transformers. In *ICML* (2021), vol. 139, pp. 11080–11090.

[22] YAO, S., PENG, B., PAPADIMITRIOU, C. H., AND NARASIMHAN, K. Self-attention networks can process bounded hierarchical languages. In *ACL/IJCNLP* (2021), pp. 3770–3785.

Appendix

Proof of Proposition 1. As Hahn showed, for every $\varepsilon > 0$ and $L > 0$ there exists $c \geq 0$ such that, for all larger enough n, if we consider as inputs binary strings of length n, for every UHAT T consisting of L layers, there exists a fixation of εn input bits such that, under this fixation, the output of T is determined by c unfixed bits [11]. However, it cannot hold for an UHAT recognizing *approximate majority*, for example, when $\varepsilon = 1/10$. Regardless of how we fix $n/10 + c$ input bits, if we fix the remaining bits to 0s, the circuit C_n rejects our string, and if we fix them to 1s, it accepts our string, even though the output of the UHAT remains unchanged. $\qquad\square$

Proof of Lemma 1. At position $i = 0, \ldots, n - 1$, this transformation can be written as follows:

$$x_i \mapsto x_i - \max\{0, x_i + i - (n+1)\}.$$

It can easily be done with ReLU layer, using a positional encoding $p(i, n) = i - n$. However, it can also be done with a positional encoding that does not depend on n, for example $p(i) = (i, 1/(i+1))$. We just have to "transmit" $n - 1$ to every position in the UHAT. For that, it is enough to have a unique hard attention layer, where attention in every position is maximized at $j = n - 1$ (which allows that to "send" n to every position). For instance, consider $\mathbf{v}_i = 1/(i+1)$, $A(x) = -x$, and observe that:

$$\arg\max_{j=0,\ldots,n-1} \langle A\mathbf{v}_i, \mathbf{v}_j \rangle = \arg\max_{j=0,\ldots,n-1} -\frac{1}{(i+1)(j+1)} = \{n - 1\}$$

for every $i = 0, \ldots, n - 1$. This finishes the proof of the lemma. $\qquad\square$

Proof of Proposition 5. **Upper bound:** We first show that every regular language over $\Sigma = \{a_1, \ldots, a_d\}$ has a Parikh-equivalent language in UHAT. By Parikh's Theorem, the Parikh image of this given regular language is represented by a semilinear set S in dimension d. Our proof employs Theorem 1. Since FO(Mon) is closed under disjunction, it suffices to consider only linear sets S. Thus, take an arbitrary linear set $S = \mathbf{v}_0 + \sum_{i=1}^{r} \mathbf{v}_i \mathbb{N}$, where $\mathbf{v}_i$ $(i > 0)$ is a non-zero vector. We will give a language L over the alphabet of $\Sigma = \{a_1, \ldots, a_d\}$ definable in FO(Mon) (thus UHAT-recognizable, by Theorem 1) such that $\mathcal{P}(L) = S$. We will use the linear set $S = (1, 1, 0) + (2, 0, 1)\mathbb{N}$ as a running example.

For $i = 0, \ldots, r$ and $j = 1, \ldots, d$, define v_i^j to be the natural number corresponding to the jth argument of $\mathbf{v}_i$. Define w_i^j to be the string $a_j^{v_i^j}$, i.e., a_j repeated v_i^j times, while ℓ_i denotes the "length abstraction" of $\mathbf{v}_i$, i.e., $\ell_i := \sum_{j=1}^{d} v_i^j$. Finally, let w_i be the concatenation of $w_i^1, \ldots, w_i^d$. Using our example of $S = (1, 1, 0) + (2, 0, 1)\mathbb{N}$, then we have $w_0 = a_1 a_2$ and $w_1 = a_1 a_1 a_3$. We also have $\ell_0 = 2$ and $\ell_1 = 3$.

Next we define the language L as follows:

$$L := w_0 \cdot w_1^* \cdots w_r^*$$

Using our running example, L would be $a_1 a_2 (a_1 a_1 a_3)^*$. It is easy to see that $\mathcal{P}(L) = S$.

To show that this language is in FO(Mon)-definable, we demonstrate that it is regular and belongs to AC^0. It is regular because it is defined through concatenation and Kleene star. Since AC^0 is closed under concatenation[2] it remains to show that languages of the form w^*, where w is a word, are in AC^0. We only have to care about input lengths that are multiples of $|w|$, for other input lengths the language is empty. Then we split the input into blocks of $|w|$ letters. We just need an AC^0-circuit, checking that every block coincides with w. For example, this can be done with an AND over blocks of constant-size circuits, checking equality to w.

Lower bound: An example of a language that is in FO(Mon) (and so in UHAT) whose Parikh image is not semilinear (and therefore, no Parikh-equivalent regular language) is

$$L = \{a^k : k \text{ is a prime number}\}.$$

Note that $\Sigma = \{a\}$. This can be easily defined in FO(Mon) using the unary predicate $\Theta := \{k \in \mathbb{N} : k + 1 \text{ is a prime number}\}$ as follows: $\exists x \Theta(x) \wedge \neg \exists y > x$. $\qquad \square$

Proof of Theorem 2. As before, we are proving that every formula ϕ of LTL($\mathbf{C}, +$) can be computed position-wise by some AHAT encoder, via structural induction. We have already shown how to do induction for all operators of LTL(Mon). In our proof, attention was always maximized at the unique j, and in this case, there is no difference between unique and average hard attention.

It remains to show the same for operators that are in LTL($\mathbf{C}, +$) but not in LTL(Mon). First, we show that given a formula ϕ, computed position-wise by some AHAT, there is also an AHAT that computes $\overleftarrow{\#\phi}$ and $\overrightarrow{\#\phi}$ position-wise.

Using future positional masking and equal weights, we can compute at position i the quantity:

$$y_i = \frac{\phi(w, 0) + \ldots + \phi(w, i)}{i + 1} = \frac{\overleftarrow{\#\phi}(w, i)}{i + 1}, \qquad i = 0, 1, \ldots, n - 1.$$

Next, we have to compute

$$z_i = \frac{\left(\overleftarrow{\#\phi}(w, i) - \phi(w, i) \right)}{i + 1}.$$

This can be achieved as follows:

$$z_i = y_i - \frac{\phi(w, i)}{i + 1} = y_i - \min \left\{ \phi(w, i), \frac{1}{i + 1} \right\}.$$

---

[2] if we have AC^0-circuits C_1, C_2 for languages L_1, L_2, we can construct an AC^0-circuit C for their concatenation as follows: $C(x_1 \ldots x_n) = \bigvee_{i=1,\ldots,n} (C(x_1 \ldots x_i) \wedge C(x_{i+1} \ldots x_n))$.

As our positional encoding includes $1/(i+1)$, this computation is a composition of ReLU and affine transformations.

Our next goal is to get rid of the coefficient $1/(i+1)$. For that, we create a layer with the following attention function:

$$\langle A\mathbf{v}_i, B\mathbf{v}_j \rangle = 2j \cdot z_i - \frac{j^2}{i+1}, \qquad i, j = 0, \ldots, n-1. \tag{1}$$

Such attention function is possible because (1) is a bilinear form of $\mathbf{v}_i$ and $\mathbf{v}_j$. Indeed, $\mathbf{v}_i$ contains $1/(i+1)$ and $\mathbf{v}_j$ contains j, j^2 due to our positional encoding, and also $\mathbf{v}_i$ contains $z_i = \frac{\left(\overleftarrow{\#\phi}(w,i) - \phi(w,i)\right)}{i+1}$.

Denoting $d_i = \overleftarrow{\#\phi}(w,i) - \phi(w,i)$, we get that (1) is equal to

$$\langle A\mathbf{v}_i, B\mathbf{v}_j \rangle = 2j \cdot \frac{d_i}{i+1} - \frac{j^2}{i+1} = \frac{-(d_i - j)^2 + d_i^2}{i+1}.$$

Observe that $d_i = \overleftarrow{\#\phi}(w,i) - \phi(w,i)$ takes values in $\{0, \ldots, n-1\}$. Hence, for a fixed i, the quantity (1) is uniquely maximized at $j = d_i$. In this way, we get $j = d_i$ to position i. Adding $\phi(w,i)$ to d_i, we get $\overleftarrow{\#\phi}(w,i)$. To get $\overrightarrow{\#\phi}(w,i)$ to position i, we observe that:

$$\overrightarrow{\#\phi}(w,i) = (\phi(w,0) + \ldots + \phi(w,n-1)) - ((\phi(w,0) + \ldots + \phi(w,i-1)))$$
$$= \overleftarrow{\#\phi}(w,n-1) - d_i.$$

This is computable at position i because $\overleftarrow{\#\phi}(w,n-1)$ can be "sent" to all positions via the attention function, always maximized at the last position (see the proof of Lemma 1).

Our next goal is: given a formula ϕ, computable position-wise by some AHAT, and a unary numerical predicate Θ, provide an AHAT that computes $\Theta(\overleftarrow{\#\phi})$ and $\Theta(\overrightarrow{\#\phi})$ position-wise. As we have already shown, we can assume that we already have counting terms $\overleftarrow{\#\phi}$ and $\overrightarrow{\#\phi}$ computed position-wise. Next, we create a layer with the following attention function:

$$\langle A\mathbf{v}_i, B\mathbf{v}_j \rangle = 2j \cdot \overrightarrow{\#\phi}(w,i) - j^2 = -(j - \overrightarrow{\#\phi}(w,i))^2 + \overrightarrow{\#\phi}(w,i)^2.$$

Again, this is possible because this expression is a bilinear form of $\mathbf{v}_i$ and $\mathbf{v}_j$, due to our positional encoding. It is maximized at $j_i = \min\{n-1, \overrightarrow{\#\phi}(w,i)\}$ (when the counting term is equal to n, since we do not have a position indexed by n, the maximizing position will be $j_i = n-1$). Having Θ included in the positional encoding, we can get j_i and $\theta_n(j_i)$ to the ith position. Observe that:

$$\theta_n(\overrightarrow{\#\phi}(w,i)) = (\mathbb{I}\{\overrightarrow{\#\phi}(w,i) \le n-1\} \wedge \theta_n(j_i)) \vee (\neg\mathbb{I}\{\overrightarrow{\#\phi}(w,i) \le n-1\} \wedge \theta_n(n))$$

Since in our positional encoding, $\theta_n(n)$ is included in every position, and since position-wise Boolean operations can be done by an AHAT, it remains to compute the indicator $\mathbb{I}\{\overrightarrow{\#\phi}(w,i) \le n-1\}$. Transmitting n once again to every

position, we can write:

$$\mathbb{I}\{\overrightarrow{\#\phi}(w,i) \le n-1\} = \min\{1, n - \overrightarrow{\#\phi}(w,i)\}.$$

This quantity can be computed by a composition of ReLU and affine transformations. We can get $\theta_n(\overleftarrow{\#\phi}(w,i))$ to the ith position analogously.

Finally, we have to check that linear inequalities over counting terms can be done in AHAT. Given formulas $\phi_1, \ldots, \phi_k, \psi_1, \ldots, \psi_k$ already computed position-wise by some AHAT, we have to provide an AHAT that computes the formula $\sum_{j=1}^{k} c_j \cdot \overleftarrow{\#\phi_j} + \sum_{j=1}^{k} d_j \cdot \overrightarrow{\#\psi_j} \ge 0$ position-wise. After computing counting terms for $\phi_1, \ldots, \phi_k, \psi_1, \ldots, \psi_k$, we first can compute their linear combination, using affine position-wise transformations:

$$l_i = \sum_{j=1}^{k} c_j \cdot \overleftarrow{\#\phi_j}(w,i) + \sum_{j=1}^{k} d_j \cdot \overrightarrow{\#\psi_j}(w,i).$$

Since coefficients are integral, l_i is integral as well, so we get:

$$\mathbb{I}\{l_i \ge 0\} = \max\{\min\{0, l_i\} + 1, 0\}.$$

The last expression can be computed via composition of ReLU and affine transformations. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Proof of Corollary 2. We show that permutation-closed languages over binary alphabets and languages of the form $perm(L)$, where L is a regular language, are expressible in $\text{LTL}(\mathbf{C}, +)$.

First, assume that L is a permutation-closed language over a binary alphabet $\{a, b\}$. Then whether or not a word $\bar{w}$ belongs to L is determined by the length of w and the number of a's in w. In other words, there a numerical predicate Θ such that for every n and for every $\bar{w} \in \{a, b\}^n$, we have $\bar{w} \in L$ if and only if $\theta_n(|\bar{w}|_a) = 1$ (recall that for a word $\bar{w}$ and for a letter a, the expression $|\bar{w}|_a$ denotes the number of occurrences of a in $\bar{w}$). Thus, L is expressible by the formula $\Theta(\overrightarrow{\#a})$.

We now show that every language of the form $perm(L)$, where L is regular, is expressible in $\text{LTL}(\mathbf{C}, +)$.

As shown in [16], if L is a regular language over the alphabet $\Sigma = \{a_1, \ldots, a_d\}$, then

$$perm(L) = \{w : \mathcal{P}(w) \in S\},$$

for some semilinear set S of dimension d. Semilinear sets correspond precisely to sets of tuples that are definable in Presburger Arithmetic (e.g. see [10]). See standard textbook in mathematical logic for more details on Presburger Arithmetic (e.g. see [2]). Since Presburger Arithmetic admits quantifier-elimination, we may assume that S is a boolean combination of (a) inequalities of linear combination of counting terms, and (b) modulo arithmetic on counting terms (i.e. an expression of the form $|w|_{a_i} \equiv k \pmod{c}$, for some concrete natural

numbers $0 \leq k < c$ and $c > 0$). For (b), one simply handles this using the formula $\Theta(\overrightarrow{\#a_i})$, where Θ is a unary numerical predicate consisting of all numbers n such that $n \equiv k \pmod{c}$. For (a), take a linear inequality of the form

$$\psi(|w|_{a_1}, \ldots, |w|_{a_d}) := \sum_{i=1}^{d} c_i |w|_{a_i} \geq 0,$$

where $c_1, \ldots, c_d \in \mathbb{Z}$. Such a formula ψ is already an atom permitted in $\text{LTL}(\mathbf{C}, +)$. Since $\text{LTL}(\mathbf{C}, +)$ is closed under boolean combination, it follows that $perm(L)$ is also in $\text{LTL}(\mathbf{C}, +)$ and therefore, by Theorem 2, is in AHAT. $\square$