NoPE: The Counting Power of Transformers with No Positional Encodings

Chris Köcher
MPI-SWS
Kaiserslautern, Germany
ckoecher@mpi-sws.org

Alexander Kozachinskiy
Centro Nacional de Inteligencia Artificial
Santiago, Chile
alexander.kozachinskyi@cenia.cl

Anthony Widjaja Lin
MPI-SWS and RPTU Kaiserslautern-Landau
Kaiserslautern, Germany
awlin@mpi-sws.org

Marco Sälzer
RPTU Kaiserslautern-Landau
Kaiserslautern, Germany
marco.saelzer@rptu.de

Georg Zetzsche
MPI-SWS
Kaiserslautern, Germany
georg@mpi-sws.org

Abstract

Positional Encodings (PEs) seem to be indispensable for ensuring expressiveness of transformers; without them attention transformers reduce to a bag-of-word model. NoPE-transformers (i.e. with No PEs) with unique hard attention mechanisms were very recently shown to only be able to express regular languages, i.e., with limited counting ability. This paper shows that, with average hard attention mechanisms, NoPE-transformers are still surprisingly expressive: they can express counting languages corresponding to nonnegative integer solutions to multivariate polynomial equations (i.e. Diophantine equations), reasoning about which is well-known to be undecidable. In fact, we provide a precise characterization of languages expressible by Average Hard Attention NoPE-Transformers (NoPE-AHATs): they correspond precisely to what we call *semi-algebraic sets*, i.e., finite unions of sets of nonnegative integer solutions to systems of multivariate polynomial inequations. We obtain several interesting consequences of our characterization. Firstly, NoPE-transformers can express counting properties that are far more complex than established models like simplified counter machines and Petri nets, but cannot express a very simple counting property of PARITY. Secondly, the problem of analyzing NoPE-transformers is undecidable, e.g., whether a given NoPE transformer classifies all input strings in one class. To complement our results, we exhibit a counting language that is not expressible by average hard attention transformers even with arbitrary PEs but is expressible in the circuit complexity class TC^0, answering an open problem.

1 Introduction

Transformers [42] have emerged in recent years as a powerful model with a plethora of successful applications including (among others) natural language processing, computer vision, and speech recognition. Despite the success of transformers, the question of what transformers can express

Preprint. Under review.

is still not well-understood and has in recent years featured in a rich body of research works (e.g. [17, 19, 33, 40]).

Formal language theory has proven to be extremely useful in understanding such expressiveness issues (e.g. see [40]). More precisely, a transformer T is said to express a formal language L (i.e. a set of strings over an alphabet Σ), when T can classify those input strings that are in L and those input strings that are not in L. One class of formal languages that have recently featured in the study of expressiveness of neural networks — in particular, Recurrent Neural Networks (RNN) and Transformers — are the so-called *counting languages*, e.g., see [2, 3, 6, 10, 17, 18, 30, 40, 44, 45]. Intuitively, counting languages (a.k.a. counter languages) require counting the numbers of occurrences of certain characters in the input string and, perhaps, additionally compare them. Of special interests are *permutation-invariant* (counting) languages, i.e., languages that are closed under shuffling the positions of the letters in the string (e.g. $\texttt{aba} \in L$ implies $\texttt{baa} \in L$). An example is the language MAJ over the alphabet $\{\mathtt{a}, \mathtt{b}\}$ consisting of strings with more $\texttt{a}$'s than $\texttt{b}$'s (e.g. $\texttt{aab} \in \mathsf{MAJ}$ but $\texttt{abb} \notin \mathsf{MAJ}$). Another example is the language PARITY consisting of all strings over the alphabet $\{\mathtt{a}, \mathtt{b}\}$ with an even number of occurrences of $\texttt{a}$. A simple application[1] of counting is sentiment analysis, where the number of positive words should exceed the number of negative words in a text.

Which counting languages can transformers express? Answering this question depends on two crucial parameters: (1) the type of attention mechanism (2) the type of *Positional Encodings (PEs)*. Most theoretical research results (see the survey [40]) focus on *hard-attention mechanisms*, which replace softmax by picking the leftmost position with the maximum attention score (i.e. *unique hard attention*) or averaging those (not necessarily leftmost) positions (i.e *average hard attention*, a.k.a., saturated attention). In the sequel, we will write UHAT (resp. AHAT) for Unique (resp. Average) Hard Attention Transformers. AHATs have been argued to be a realistic approximation of how transformers function in practice [27]. In this paper, we primarily focus on AHATs, though we will also discuss implications on soft attention transformers. We now proceed to PEs. Attention is an operation that aggregates an input sequence via a weighted sum, which treats the elements in the sequence as a *bag* (i.e. permutation invariant). PEs — which essentially annotate elements in the sequence by some positional information like i and $\sin(2\pi \cdot i)$ (for positive integers i) — are often used to recover the ordering of the elements in the sequence. PEs are, however, known to substantially increase the expressive power of transformers in theory since essentially all computable PEs of the form $p : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^d$ (or $p : \mathbb{N} \times \mathbb{N} \to \mathbb{Q}^d$) are permitted[2]. This has led some researchers to use *positional masking* (a.k.a. causal attention) [39, 45] — which essentially filters elements in the sequence (relative to the current position) before applying attention — instead of PEs.

Several recent results shed some lights on the expressivity of transformers on counting languages. Firstly, UHATs (even with PEs) are known to be strictly contained in the (nonuniform) circuit complexity class AC^0 [2, 19], which characterizes the "inability of counting", e.g., the most basic counting language PARITY is not in AC^0 [1]. In fact, NoPE-UHATs with positional masking can express only star-free regular languages [45]. What about AHATs? We know that AHAT languages (with PEs) are subsumed in the circuit complexity class $\mathsf{TC}^0 \supseteq \mathsf{AC}^0$ [19, 29], which essentially enriches AC^0 circuits with "counting" and "arithmetics". Whether AHAT languages (with PEs) are strictly contained in TC^0 was posed an open problem [2]. Furthermore, AHAT (with PEs) can express all counting languages corresponding to the permutation closures of all regular languages [2]. The question of precisely which counting languages AHATs can express remains open. Furthermore, since some of these constructions employed non-trivial PEs (e.g. trigonometry functions), the question arises as well on the role of PEs in recognizing these languages.

Contributions. Our main contribution is to show that NoPE-AHATs are extremely expressive, in contrast to the case of UHATs [45]: they can express nonnegative integer solution sets to multivariate polynomial equations (i.e. *Diophantine equations*), which play an important role in mathematics (particularly, number theory, and algebraic geometry) and theoretical computer science (particularly, computability theory [25]). A simple example of such a permutation-invariant language is

$$\mathsf{SQRT} = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w|_{\mathtt{a}} < |w|/\sqrt{2}\}, \tag{1}$$

[1] https://medium.com/data-science/sentiment-analysis-with-text-mining-13dd2b33de27

[2] Some results even refrain from any computability assumption since commonly used PEs (e.g. trigonometry functions) are not rational, e.g., [2]

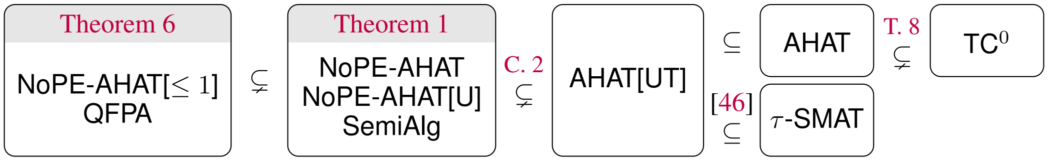

Figure 1: Visualization of our results.

containing words whose proportion of the positions with letter a is at most $\sqrt{2}$. More precisely, we provide a *precise characterization* of languages expressible in NoPE-AHAT: finite unions of sets of non-negative integer solutions to multivariate polynomial inequalities, which we call *semi-algebraic sets* (henceforth, written SemiAlg). To the best of our knowledge, this is the first result showing that a sequential model based on neural networks (including RNN and transformers) could represent all Diophantine equations. Our main result and its consequences (more below) are summarized in Fig. 1.

Key consequences of our main result: 1. Counting power of NoPE-AHATs in relation to other established models. 2. Undecidability of reasoning about transformers 3. Counting power of soft attention transformers

Firstly, NoPE-AHAT can express counting languages (e.g. SQRT) that cannot be expressed by established models in the literature of transformers (e.g. UHAT with PEs), circuit complexity (AC^0), and formal languages and concurrency theory (e.g. simplified counter machines, Petri nets, and models of higher-order recursion). In particular, simplified counter machines — frequently employed in the investigation of the counting power of RNN and Transformers [30, 44] — cannot recognize permutation-invariant languages beyond *linear integer arithmetics* (a.k.a. semilinear sets [32], meaning those definable in existential Presburger arithmetic). Interestingly, it follows from our characterization that PEs are indispensable for capturing PARITY $\notin$ SemiAlg using AHAT.

Secondly, we can apply our result to show undecidability of formally *verifying* (a.k.a. *interpreting* or *checking robustness*) of transformers, which has recently received attention (e.g. see [34]). For less formal approaches to verification, cf. [4, 11, 20, 36]. [More generally, formal verification of neural networks (including feed-forward neural networks and RNN) is an established research field (cf. [21, 24, 36]) with an annual solver competition (cf. [5]).] Especially, our result implies undecidability of verifying a very simple transformer model with no PEs, left as an open question in [34].

For these aforementioned results, we provide a detailed *parameteric analysis* in terms of the number of layers. In particular, with one layer, NoPE-AHAT expresses only linear Diophantine equations (more precisely, *quantifier-free Presburger Arithmetic (QFPA)*, and so cannot define SQRT. To achieve powerful counting ability (e.g. resulting in undecidability), we show sufficiency of NoPE-AHATs with only two layers.

Finally, it was recently shown [46] that soft attention transformers with either PEs or temperature scaling (written τ-SMAT) can simulate AHAT languages that have the so-called *uniform-tieless* property (AHAT[UT]): each layer is *either* uniform (U) or tieless (T). Our construction of NoPE-AHATs from SemiAlg in fact satisfies a stronger condition: no tieless layers are needed. As a result, τ-SMAT can also express counting languages corresponding to solutions to Diophantine equations.

Separation from TC^0. To complement our above results, we use an *information-theoretic argument* to exhibit a permutation-invariant language that is not expressible by AHAT even in the presence of PEs. In particular, we show that this implies that AHAT-definable languages are a *strict* subset of TC^0, answering an open problem from [2].

Organization. After recalling basic definitions in Section 2, we provide a summary of results in Section 3 as a roadmap for the rest of the paper. We then prove a characterization of the power of NoPE-AHATs in Section 4, and a parameteric analysis in Section 5. We separate TC^0 from AHAT with PEs in Section 6, and conclude in Section 7. Missing details can be found in the appendix.

2 Preliminaries

ReLU neural networks A *ReLU node* v is a function $\mathbb{Q}^m \to \mathbb{Q}$, where $m \in \mathbb{N}$ is referred to as the input dimension, and is defined as $v(x_1, \ldots, x_n) = \max(0, b + \sum_{i=1}^n w_i x_i)$, where $w_i \in \mathbb{Q}$ are the *weights*, and $b \in \mathbb{Q}$ is the *bias*. A *ReLU layer* ℓ is a tuple of ReLU nodes $(v_1, \ldots, v_n)$, all having the same input dimensionality, computing a function $\mathbb{Q}^m \to \mathbb{Q}^n$, where $n \in \mathbb{N}$ is referred to as the output dimension. Finally, a *ReLU neural network* $\mathcal{N}$ is a tuple of ReLU layers $(\ell_1, \ldots, \ell_k)$, such that the input dimension of ℓ_{i+1} is equal to the output dimension of ℓ_i. It computes a function $\mathbb{Q}^{m_1} \to \mathbb{Q}^{n_k}$, given by $\mathcal{N}(x_1, \ldots, x_{m_1}) = \ell_k(\cdots \ell_1(x_1, \ldots, x_{m_1}) \cdots)$.

Average hard attention layers An *AHA layer* is a function $\lambda\colon (\mathbb{Q}^d)^* \to (\mathbb{Q}^e)^*$, given by affine maps $Q, K\colon \mathbb{Q}^d \to \mathbb{Q}^m$, $V\colon \mathbb{Q}^d \to \mathbb{Q}^k$ (query, key, and value matrices) and a ReLU neural net $\mathcal{N}\colon \mathbb{Q}^{d+k} \to \mathbb{Q}^e$. Given an input sequence $x = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{Q}^d)^n$, the output sequence $y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n) \in (\mathbb{Q}^d)^n$ is computed as follows. First, one computes the sequences of key, query, and value vectors: $\boldsymbol{k}_i = K\boldsymbol{x}_i,\ \boldsymbol{q}_i = Q\boldsymbol{x}_i,\ \boldsymbol{v}_i = V\boldsymbol{x}_i, \qquad i = 1, \ldots, n$, then a sequence of *attention vectors* defined by $\boldsymbol{a}_i = \frac{1}{|P|}\sum_{j \in P} \boldsymbol{v}_j$, where $P \subseteq \{1, \ldots, n\}$ is the set of those positions j for which $\langle \boldsymbol{k}_i, \boldsymbol{q}_j \rangle$ is maximal. Finally, one sets: $\boldsymbol{y}_i = \mathcal{N}(\boldsymbol{x}_i, \boldsymbol{a}_i)$. We say λ is *uniform* iff the key and query maps K and Q are constant, it is called *tieless* iff for all input sequences and all positions the set of positions P with maximal attention is a singleton.

Average hard attention transformers An *AHA Transformer* (AHAT) with ℓ layers over a finite alphabet Σ is a function $T\colon \Sigma^+ \to \{0, 1\}$, given by: (i) the "input embedding" function $\iota\colon \Sigma \to \mathbb{Q}^{d_1}$, (ii) the positional encoding $p\colon \mathbb{N}^2 \to \mathbb{Q}^{d_1}$, and (iii) a sequence of AHA layers $\lambda_1\colon (\mathbb{Q}^{d_1})^* \to (\mathbb{Q}^{d_2})^*, \ldots, \lambda_\ell\colon (\mathbb{Q}^{d_\ell})^* \to (\mathbb{Q}^{d_{\ell+1}})^*$. Given an input word $w = a_1 \cdots a_n \in \Sigma^n$, the output $T(w)$ is computed as follows. First, we set $\boldsymbol{x}_1 = \iota(a_1) + p(n, 1),\ \ldots,\ \boldsymbol{x}_n = \iota(a_n) + p(n, n)$. Then we compute $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n) = \lambda_\ell(\lambda_{\ell-1}(\cdots \lambda_1(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \cdots))$, and we set $T(w) = 1$ if and only if $\boldsymbol{y}_n[1] > 0$, and $T(w) = 0$ otherwise. We say that T has *no positional encoding* (NoPE) if the positional encoding is a constant function.

AHAT language classes We study AHAT that accept languages *with an end marker*, i.e. the input word $w \in \Sigma^*$ is first extended by an end marker $\$ \notin \Sigma$, and then T is evaluated on $w\$$. Thus, by AHAT we denote the class of all languages $L \subseteq \Sigma^+$ such that there is an AHAT T over $\Sigma \cup \{\$\}$ (with $\$ \notin \Sigma$) so that $L = \{w \in \Sigma^+ \mid T(w\$) = 1\}$. If we restrict the AHAT to have only uniform (resp. only uniform or tieless layers), then we write $\mathsf{AHAT[U]}$ (resp. $\mathsf{AHAT[UT]}$). When we restrict to AHAT with at most ℓ attention layers, then we obtain $\mathsf{AHAT}[\leq \ell]$. Moreover, the restriction to all languages accepted by an AHAT *without positional encoding* is denoted as $\mathsf{NoPE\text{-}AHAT}$. The classes $\mathsf{NoPE\text{-}AHAT[U]}$, $\mathsf{NoPE\text{-}AHAT[UT]}$, $\mathsf{NoPE\text{-}AHAT}[\leq \ell]$ are then defined similarly to $\mathsf{AHAT[U]}$, etc.

Other language classes Let RE denote the class of recursively enumerable languages [38], i.e. those recognized by (not necessarily terminating) Turing machines. TC^0 denotes the class of all languages defined by a family of polynomially sized circuits of constant depth containing only Boolean and majority gates (see [43] for more details). For an alphabet Σ with $\Sigma = \{\mathtt{a}_1, \ldots, \mathtt{a}_m\}$, we define the *Parikh map* as the function $\Psi\colon \Sigma^* \to \mathbb{N}^m$, where $\Psi(w)[i] := |w|_{\mathtt{a}_i}$ is the number of $\mathtt{a}_i$'s in w. A language $L \subseteq \Sigma^*$ is *permutation-invariant* if for $u, v \in \Sigma^*$ with $\Psi(u) = \Psi(v)$, we have $u \in L$ if and only if $v \in L$. These have also been called "permutation-closed" or "proportion-invariant" e.g. in [33]. By PI, we denote the class of languages that are permutation-invariant. In particular, $\mathsf{RE} \cap \mathsf{PI}$ is the class of languages that are (i) recursively enumerable and (ii) permutation-invariant.

For a class $\mathcal{C}$ of languages, we denote by $\mathsf{Proj}(\mathcal{C})$ the class of *projections*, i.e. the languages of the form $\pi(L)$, where L is from $\mathcal{C}$ and π is a map that deletes a subset of the letters.

Presburger arithmetic (PA) refers to the first-order theory of the structure $\langle \mathbb{N}; +, 0, 1, < \rangle$ [7, 16]. The *quantifier-free* fragment of PA includes all PA formulas that do not contain any quantifiers; in other words, these are Boolean combinations of atomic formulas. We assume that all atomic formulas are of the form $c_1 x_1 + \cdots + c_n x_n \leq b$ with $c_i, b \in \mathbb{Q}$, which we refer to as *linear inequalities*. Additionally, we use common abbreviations such ans $=$ or $>$. Given a PA formula with m free variables, meaning variables not bound by any quantifier, we denote by $[\![\varphi]\!]$ the set of vectors in $\mathbb{N}^m$ that satisfy φ. Let QFPA denote the class of languages $L \subseteq \Sigma^+$ for which there exists a quantifier-free PA formula φ with $|\Sigma|$ free variables such that $L = \{w \in \Sigma^+ \mid \Psi(w) \in [\![\varphi]\!]\}$.

3 Summary of results

In this section, we provide a rather detailed summary of our results, which would serve as a roadmap for the rest of the paper. In particular, we specify results and defer proof sketches to Sections 4 to 6.

The power of NoPE-AHAT A subset $S \subseteq \mathbb{N}^m$ is *semi-algebraic* if it is a Boolean combination of sets of the form $S_p = \{\boldsymbol{x} \in \mathbb{N}^m \mid p(\boldsymbol{x}) > 0\}$ for some polynomial $p \in \mathbb{Z}[X_1, \ldots, X_m]$. A language $L \subseteq \Sigma^*$ is *semi-algebraic* if there is a semi-algebraic set $S \subseteq \mathbb{N}^m$ and $\Sigma = \{\mathsf{a}_1, \ldots, \mathsf{a}_m\}$ such that $L = \{w \in \{\mathsf{a}_1, \ldots, \mathsf{a}_m\}^* \mid \Psi(w) \in S\}$. Let $\mathsf{SemiAlg}$ denote the class of semi-algebraic languages. An example is the set SQRT as defined in (1), since $|w|_\mathsf{a} < |w|/\sqrt{2}$ if and only if $2|w|_\mathsf{a}^2 < |w|^2$.

Our first, and arguably most important, main result is the following.

Theorem 1. $\mathsf{NoPE\text{-}AHAT} = \mathsf{NoPE\text{-}AHAT}[\mathsf{U}] = \mathsf{SemiAlg}$.

Note that for every $p \in \mathbb{Z}[X_1, \ldots, X_m]$, the set $\{\boldsymbol{x} \in \mathbb{N}^m \mid p(\boldsymbol{x}) = 0\}$ is semi-algebraic, because $p(\boldsymbol{x}) = 0$ if and only if $-p(\boldsymbol{x})^2 + 1 > 0$. Thus, Theorem 1 implies that every solution set to polynomial equations belongs to $\mathsf{NoPE\text{-}AHAT}$.

Let us see some consequences of Theorem 1. We begin with positive results, i.e. examples of languages that can be recognized by NoPE-AHAT. First, Theorem 1 implies that NoPE-AHAT are *almost* Turing-complete: Up to projections, they can recognize all permutation-invariant recursively enumerable languages. More precisely, we will deduce that $\mathsf{Proj}(\mathsf{NoPE\text{-}AHAT}) = \mathsf{RE} \cap \mathsf{PI}$. The latter implies that $\mathsf{NoPE\text{-}AHAT}$ itself goes beyond extremely powerful formalisms in the literature on automata theory, verification, and neural networks: There is a language in $\mathsf{NoPE\text{-}AHAT}$ that is not recognized by a higher-order recursion scheme, a Petri net, a simplified counter machine, nor an $\mathsf{LTL}[\mathsf{Count}]$ formula. This will be formalized in Corollary 4 (which is even stronger). Moreover, the equation $\mathsf{Proj}(\mathsf{NoPE\text{-}AHAT}) = \mathsf{RE} \cap \mathsf{PI}$ implies that the emptiness problem for $\mathsf{NoPE\text{-}AHAT}$ is undecidable. Again, we will later have a stronger statement in Corollary 5.

We can also use Theorem 1 to show that a language that is well-known not to be accepted by a UHAT, is also not accepted by a NoPE AHAT. Let $\mathsf{PARITY} = \{w \in \{\mathsf{a}, \mathsf{b}\}^+ \mid |w|_\mathsf{a} \text{ is even}\}$.

Corollary 2. PARITY *does not belong to* $\mathsf{NoPE\text{-}AHAT}$.

PARITY is known to be accepted by an AHAT with PE [2]. Thus surprisingly, PEs increase the power of AHAT, even among languages that are permutation-invariant.

NoPE-AHAT with two attention layers The AHAT we construct in for Theorem 1, and thus for the equality $\mathsf{Proj}(\mathsf{NoPE\text{-}AHAT}) = \mathsf{RE} \cap \mathsf{PI}$, employ several attention layers. We will show that for the latter "almost Turing-completeness", just two attention layers suffice:

Theorem 3. $\mathsf{Proj}(\mathsf{NoPE\text{-}AHAT}[\leq 2]) = \mathsf{RE} \cap \mathsf{PI}$.

From Theorem 3, we can deduce that already with two attention layers, NoPE-AHAT go beyond very powerful models from automata theory, verification, and neural networks:

Corollary 4. *There is a language in* $\mathsf{NoPE\text{-}AHAT}[\leq 2]$ *that is not recognized by (i) higher-order recursion schemes, (ii) Petri nets, (iii) simplified multi-counter machines, (iv)* $\mathsf{LTL}[\mathsf{Count}]$.

Here, *higher-order recursion schemes* (HORS) are a prominent model for programs with higher-order recursion. HORS are the central model in the area of higher-order model checking [31]. Moreover, *Petri nets* (also known as VAS) are a widely used and studied model of concurrent programs [12]. The two models are some of the most powerful models (short of Turing-complete and thus undecidable ones) in the overall area of automata theory; moreover, they are expressively incomparable. Their accepted languages are defined in, e.g. [31] for HORS and, e.g. in [9, 23] for Petri nets.

Corollary 4 also compares NoPE-AHAT with simplified multi-counter machines (SMCM) [44] and $\mathsf{LTL}[\mathsf{Count}]$. SMCM have been studied in the literature on neural networks [30, 44]. Roughly speaking, SMCM are multi-counter machines where counter updates depend only on the current input letter; see Section A.1 for details. Moreover, $\mathsf{LTL}[\mathsf{Count}]$ is a variant of the logic $\mathsf{LTL}[\mathsf{C}, +]$ of [2]. $\mathsf{LTL}[\mathsf{C}, +]$ is a powerful logic introduced in [2] to showcase the expressiveness of AHAT: Every language definable in $\mathsf{LTL}[\mathsf{C}, +]$ is accepted by an AHAT [2, Thm. 2]. In our variant $\mathsf{LTL}[\mathsf{Count}]$, we remove the feature of *arbitrary unary predicates*, which can be captured by AHAT thanks to PE. In the absence of PE, there is no hope to capture these; hence our comparison against $\mathsf{LTL}[\mathsf{Count}]$. See Section A.2 for a detailed definition of $\mathsf{LTL}[\mathsf{Count}]$.

Corollary 4 follows from Theorem 3 with two arguments. First, the languages of higher-order recursion schemes and of Petri nets are closed under projections (e.g. [8]; for Petri nets, this is immediate from the definition) and each of them has a decidable membership problem [26, 31]. But Theorem 3 tells us that there is a language K in $\mathsf{NoPE\text{-}AHAT}[\leq 2]$ and a projection π such that $\pi(K)$ has an undecidable membership problem. This implies that K can not be recognized by a HORS nor by a Petri net. For SMCM and $\mathsf{LTL}[\mathsf{Count}]$, we have to argue differently, because they are not closed under projection. However, we prove in Section A.1 (for SMCM) and in Section A.2 (for $\mathsf{LTL}[\mathsf{Count}]$) that permutation-invariant languages in these two classes must have semilinear Parikh images. Since $\pi(K)$ is permutation-invariant and undecidable and thus not semilinear, K cannot be semilinear either. Hence, K is not recognized by an SMCM nor by $\mathsf{LTL}[\mathsf{Count}]$.

Moreover, Theorem 3 implies that emptiness is undecidable, already with two attention layers:

Corollary 5. *The emptiness problem for* $\mathsf{NoPE\text{-}AHAT}[\leq 2]$ *is undecidable.*

NoPE-AHAT with one attention layer We have seen that NoPE AHAT are extremely powerful already with two attention layers. The same is not true for a single attention layer. Indeed, we have a characterization of NoPE AHAT with one layer in terms of Presburger arithmetic:

Theorem 6. $\mathsf{NoPE\text{-}AHAT}[\leq 1] = \mathsf{QFPA}$.

For example, this implies that with just one attention layer NoPE-AHAT are strictly less powerful than with two: $\mathsf{Proj}(\mathsf{NoPE\text{-}AHAT}[\leq 2])$ contains all permutation-invariant recursively enumerable languages, whereas $\mathsf{Proj}(\mathsf{NoPE\text{-}AHAT}[\leq 1]) = \mathsf{Proj}(\mathsf{QFPA})$ is the class of languages with semilinear Parikh images. Since not every recursively enumerable set of vectors is semilinear, the two classes $\mathsf{NoPE\text{-}AHAT}[\leq 1]$ and $\mathsf{NoPE\text{-}AHAT}[\leq 2]$ must differ.

Moreover, the translation between $\mathsf{NoPE\text{-}AHAT}[\leq 1]$ and QFPA can be done algorithmically. Since satisfiability of existential Presburger is decidable [16], we obtain:

Corollary 7. *The emptiness problem for* $\mathsf{NoPE\text{-}AHAT}[\leq 1]$ *is decidable.*

NoPE-AHAT without end marker Given that our definition of $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}$ uses an end marker, we also investigated the setting of NoPE-AHAT without end marker. Our results are given in Section D. For example, we reveal a surprising connection to an open problem in number theory: While emptiness is undecidable for $\mathsf{NoPE\text{-}AHAT}$ (Corollary 5), decidability of emptiness for $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}$ is equivalent to decidability of solvability of Diophantine equations over the rationals. Whether decidability holds here is a major open problem in number theory [37]. Moreover, without an end marker, NoPE-AHAT still go beyond $\mathsf{LTL}[\mathsf{Count}]$ and simplified multicounter machines, already with two layers. See Section D for a list of results.

Counting beyond AHAT We conclude by showing the existence of a permutation-invariant language that lies beyond AHAT, even in the presence of PEs.

Theorem 8. *There is a permutation-invariant language that cannot be captured by* AHAT*, even with PEs. Hence,* $\mathsf{AHAT} \subsetneq \mathsf{TC}^0$.

See Section 6 for a proof. The "Hence" part of the theorem is a consequence of a standard result in circuit complexity that each permutation-invariant language is in TC^0; for completeness, we provide a simple argument in Section A.3.

4 Characterizing the power of NoPE-AHAT

In this section, we give details on the proofs of Theorems 1 and 20 and Corollary 2. We begin with Theorem 1. Since the inclusion $\mathsf{NoPE\text{-}AHAT}[\mathsf{U}] \subseteq \mathsf{NoPE\text{-}AHAT}$ is trivial, there are two interesting inclusions: $\mathsf{NoPE\text{-}AHAT} \subseteq \mathsf{SemiAlg}$ and $\mathsf{SemiAlg} \subseteq \mathsf{NoPE\text{-}AHAT}[\mathsf{U}]$. The latter follows from:

Proposition 9. *For every polynomial* $p \in \mathbb{Z}[X_1, \ldots, X_m]$*, the language* $L_{p>0} = \{w \in \{\mathsf{a}_1, \ldots, \mathsf{a}_m\}^* \mid p(\Psi(w)) > 0\}$ *belongs to* $\mathsf{NoPE\text{-}AHAT}[\mathsf{U}]$.

Let us see why Proposition 9 implies $\mathsf{SemiAlg} \subseteq \mathsf{NoPE\text{-}AHAT}[\mathsf{U}]$. First, the complement of each language $L_{p>0}$ can be obtained, because $p(\boldsymbol{x}) > 0$ is violated if and only if $-p(\boldsymbol{x}) + 1 > 0$. Moreover, $\mathsf{NoPE\text{-}AHAT}$ is closed under union and intersection (see Section B.1). We can thus accept all Boolean combinations of languages of the form $L_{p>0}$, and hence $\mathsf{SemiAlg}$.

To show Proposition 9, we will use polynomials that are *homogeneous*, meaning all monomials have the same degree. Note that given an arbitrary polynomial $p \in \mathbb{Z}[X_1, \ldots, X_m]$ of degree d, we can consider the polynomial $q \in \mathbb{Z}[X_0, \ldots, X_m]$ with $q = X_0^d p(\frac{X_1}{X_0}, \ldots, \frac{X_m}{X_0})$, which is homogeneous. It has the property that $p(x_1, \ldots, x_m) > 0$ if and only if $q(1, x_1, \ldots, x_m) > 0$. Therefore, from now on, we assume that we have a homogeneous polynomial $q \in \mathbb{Z}[X_0, \ldots, X_m]$ and want to construct an AHAT for the language $K_q = \{w \in \{\mathsf{a}_1, \ldots, \mathsf{a}_m\}^* \mid q(1, \boldsymbol{x}) > 0 \text{ for } \boldsymbol{x} = \Psi(w)\}$.

To simplify notation, we denote the end marker by a_0. Thus, the input will be a string $w \in \{\mathsf{a}_0, \ldots, \mathsf{a}_m\}^+$ that contains a_0 exactly once, at the end. Since $|w|_{\mathsf{a}_0} = 1$ is satisfied automatically, our AHAT only has to check that $q(x_0, \ldots, x_m) > 0$, where $x_i = |w|_{\mathsf{a}_i}$. The input encoding is the map $\{\mathsf{a}_0, \ldots, \mathsf{a}_m\}^* \to \mathbb{Q}^m$ with $\mathsf{a}_i \mapsto \boldsymbol{e}_i$, where $\boldsymbol{e}_i \in \mathbb{Q}^m$ is the i-th unit vector.

Step I: Compute frequencies Our AHAT first uses an attention layer to compute $m + 1$ new components, where i-th component holds $\frac{x_i}{n+1}$, where $n + 1$ is the length of the input (including the end marker). This is easily done by attending to all positions and computing the averages of the first $m + 1$ components. To simplify notation, we will index vectors starting with index 0.

Step II: Multiplication gadgets Second, we have a sequence of gadgets (each consisting of two layers). Each gadget introduces a new component, and does not change the existing components. Between gadget executions, the following additional invariants are upheld: (i) Overall, a gadget does not change existing components: it introduces one new component. (ii) The components $\{0, \ldots, m\}$ are called the *initial* components. (iii) All other components are *uniform*, i.e. they are the same across all positions. (iv) The uniform components carry values in $[0, 1]$. Thus, we will call components $0, \ldots, m$ the *initial* components; and we call components $> m$ the *uniform* components.

Our gadgets do the following. Suppose we have already produced ℓ additional components. For each initial component $i \in [0, m]$ and uniform component $j \in [m + 1, m + 1 + \ell]$, gadget $\mathsf{omult}(\ell, i, j)$, which introduces a new component, will carry the value $\frac{x_i \cdot y_j}{n+1}$, where y_j is the value in component j of all vectors. Recall that we use x_i to denote the number of a_i occurrences in the input for $i \in [0, m]$.

We implement the gadget $\mathsf{omult}(\ell, i, j)$ using some ReLU layers and an attention layer. Suppose that before, we have the vector $\boldsymbol{u}_p \in \mathbb{Q}^{m+1+\ell}$ in position p. First, using ReLU layers, we introduce a new component that in position p has the value $\boldsymbol{u}_p[i] \cdot \boldsymbol{u}_p[j]$. This can be achieved since $\boldsymbol{u}_p[i]$ is in $\{0, 1\}$ and $\boldsymbol{u}_p[j] \in [0, 1]$: Notice that $\boldsymbol{u}_p[i] \cdot \boldsymbol{u}_p[j] = \mathrm{ReLU}(\boldsymbol{u}_p[j] - (1 - \boldsymbol{u}_p[i]))$. Indeed, if $\boldsymbol{u}_p[i] = 1$, then this evaluates to $\boldsymbol{u}_p[j]$; if $\boldsymbol{u}_p[i] = 0$, then we get $\mathrm{ReLU}(\boldsymbol{u}_p[j] - 1) = 0$. We then use uniform attention to compute the average of this new $\boldsymbol{u}_p[i] \cdot \boldsymbol{u}_p[j]$-component across all vectors. Since there are $n + 1$ vectors, exactly x_i of them have $\boldsymbol{u}_p[i] = 1$, and also $\boldsymbol{u}_p[j] = y_j$, we get the desired $\frac{x_i \cdot y_j}{n+1}$.

Step III: Computing the polynomial We now use our gadgets to compute the value of the polynomial. For each monomial of q, say $X_{i_1} \cdots X_{i_d}$, we use $d - 1$ gadgets to compute $x_{i_1} \cdots x_{i_d}/(n+1)^d$: The frequency computation in the beginning yields $x_{i_1}/(n + 1)$, and then we use gadgets to compute $x_{i_1} x_{i_2}/(n+1)^2$, $x_{i_1} x_{i_2} x_{i_3}/(n+1)^3$, etc. until $x_{i_1} \cdots x_{i_d}/(n+1)^d$. Finally, we use a ReLU layer to multiply each monomial with a rational coefficient, and compute the sum of all the monomials. Thus, we have computed $q(x_0, \ldots, x_m)/(n + 1)^d$. We accept if and only if $q(x_0, \ldots, x_m)/(n + 1)^d > 0$. Note that this is the case if and only if $q(x_0, \ldots, x_m) > 0$.

This completes Proposition 9 and thus $\mathsf{SemiAlg} \subseteq \mathsf{NoPE\text{-}AHAT}[\mathsf{U}]$. It remains to show:

Proposition 10. $\mathsf{NoPE\text{-}AHAT} \subseteq \mathsf{SemiAlg}$.

Proof. Suppose that $\Sigma = \{\mathsf{a}_1, \ldots, \mathsf{a}_m\}$ is our alphabet, a_0 the end marker, and $x_i \in \mathbb{N}$ the number of occurrences of a_i in the input. We say that a position p is an a_i*-position* if the input holds a_i at position p. Notice that an AHAT without positional encoding cannot distinguish vectors that come from the same input letter. This means, in any layer, any two a_i-positions will hold the same vector. Thus, the vector sequence on layer ℓ is described by rational vectors $\boldsymbol{u}_{\ell,0}, \ldots, \boldsymbol{u}_{\ell,m}$, where $\boldsymbol{u}_{\ell,i}$ is the vector at all the a_i-positions on layer ℓ. Moreover, for each i, the set of positions maximizing an attention score also either contains all a_i-positions, or none of them. Therefore, if the AHAT has a

attention layers, there are at most $((2^{m+1})^{m+1})^a = 2^{(m+1)^2 a}$ possible ways to choose the positions of maximal score: On each attention layer, and for each $i \in [0, m]$, we select a subset of the $m+1$ letters. For each ReLU node and each i, there are two ways its expression $\mathrm{ReLU}(v)$ can be evaluated: as 0 or as v. Thus, if there are r ReLU nodes, then there are 2^r ways to evaluate all those nodes.

For each of these $2^{r+(m+1)^2 a}$ choices, we construct a conjunction of polynomial inequalities that verify that (i) this choice actually maximized scores, (ii) the resulting vector at the right-most position in the last layer satisfies the accepting condition. This is easy to do by building, for each layer ℓ and each i, expressions in $x_1, \dots, x_m$ for the vectors $\boldsymbol{u}_{\ell,i}$, assuming our choice above. These expressions have the form $p(x_1, \dots, x_m)/q(x_1, \dots, x_m)$ (averaging can introduce denominators). Here, once we have expressions for $\boldsymbol{u}_{\ell,i}$, we can use them to build expressions for $\boldsymbol{u}_{\ell+1,i}$ by following the definition of AHAT. Checking (i) and (ii) is then also easy, because inequalities involving quotients $p(x_1, \dots, x_m)/q(x_1, \dots, x_m)$ can be turned into polynomial inequalities by multiplying with common denominators. Finally, we take a disjunction over all $2^{r+(m+1)a}$ conjunctions. □

Inexpressibility of PARITY Let us now show Corollary 2. By Theorem 1, it suffices to show that PARITY is not semi-algebraic. Suppose it is. Then there is a disjunction of conjunctions of polynomial inequalities that characterizes PARITY. The polynomials are over $\mathbb{Z}[X, Y]$, where X is the variable for a's and Y is the variable for b's. By plugging in $Y = 0$, we conclude that the set of even numbers is semi-algebraic. Hence, there is a disjunction $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} p_{i,j}(X) > 0$ of conjunctions that is satisfied exactly for the even numbers. This implies that for some i, there are infinitely many even numbers k such that $\bigwedge_{j=1}^{m} p_{i,j}(k) > 0$. Therefore, for every $j \in [1, m]$, the leading coefficient of $p_{i,j}$ must be positive. But then, $\bigwedge_{j=1}^{m} p_{i,j}(k) > 0$ must hold for all sufficiently large k, not just the even ones, a contradiction.

As mentioned in Section 3, we can deduce many more results from Theorem 1. Since we also prove stronger versions for NoPE-AHAT with at most two layers, we defer the proofs to Section 5.

5 Parametric analysis

Capturing RE **with two layers** We sketch the proof of Theorem 3 (details in Section C.1). The first ingredient is that by the MRDP theorem one Diophantine sets, every language in RE∩PI is a projection of a language of the form $L_p = \{w \in \{\mathtt{a}_1, \dots, \mathtt{a}_m\}^* \mid p(\Psi(w)) = 0\}$, where $p \in \mathbb{Z}[X_1, \dots, X_m]$ is a polynomial [25]. Thus, it suffices to place L_p in NoPE-AHAT[≤ 2]. First observe that in Theorem 1, we use one attention layer for each multiplication, so this avenue is closed if we want to stay within two attention layers. Instead, we use that for every polynomial $p \in \mathbb{Z}[X_1, \dots, X_m]$, there are *quadratic* (i.e. degree ≤ 2) polynomials $q_1, \dots, q_r \in \mathbb{Z}[X_1, \dots, X_{m+k}]$ for some $r, k \geq 0$ such that for $\boldsymbol{x} \in \mathbb{N}^m$, we have $p(\boldsymbol{x}) = 0$ if and only if there is some $\boldsymbol{y} \in \mathbb{N}^k$ with $q_1(\boldsymbol{x}, \boldsymbol{y}) = 0, \dots, q_r(\boldsymbol{x}, \boldsymbol{y}) = 0$: Just introduce a fresh variable for each multiplication in p and use the q_i to assign these fresh variables. Now as before, the first attention layer computes letter frequencies. Then, the score function (which, as a bilinear map, can evaluate quadratic polynomials) in the second attention layer evaluates the quadratic polynomials q_i.

NoPE AHAT with a single layer Let us briefly sketch the proof of Theorem 6. For the inclusion NoPE-AHAT[≤ 1] $\subseteq$ QFPA, we proceed similarly to Proposition 10, while observing that the inequalities we have to verify are all linear inequalities: This is because a single attention layer averages only once. Conversely, for the inclusion QFPA $\subseteq$ NoPE-AHAT[≤ 1], we use one attention layer to compute all letter frequencies, and then use ReLU layers to evaluate linear inequalities. The full proof of Theorem 6 can be found in Section C.2.

6 Beyond AHAT

We provide here a proof of Theorem 8, which consequently separates AHAT and TC^0. We do not give an explicit construction for this separating language, but rather a "counting" argument. More specifically, by *C-bounded AHATs* we mean AHATs where the embedding dimension and the number of layers do not exceed C. First, we establish the following lemma:

Lemma 11. *Let $C > 0$ and the size of the input alphabet Σ be fixed. Then for any n, and for any k inputs $x_1, \ldots, x_k \in \Sigma^n$ the number of Boolean functions that can be computed by C-bounded AHATs on $\{x_1, \ldots, x_k\}$ is at most $k^{O(n^3)}$.*

To derive the theorem from this lemma, observe the following. For $|\Sigma| = 5$, and for any n, there exist $k = \Theta(n^4)$ inputs from Σ^n that cannot be obtained from each other by permuting letters. Hence, permutation-invariant languages realize all $2^{\Omega(n^4)}$ Boolean functions on these inputs. On the other hand, when C is fixed, C-bounded AHATs realize just $2^{O(n^3 \log n)}$ different functions on these inputs. Thus, for any C, we can take n large enough and define our permutation-invariant language on Σ^n in such a way that no C-bounded AHAT computes the restriction of our language to inputs of length n. Doing this for all C using larger and larger n finishes the proof.

It remains to prove the lemma. We will view C-bounded AHATs, restricted to inputs of length n, as concept classes in the standard PAC-learning setting [15]: there are input vectors, consisting of coordinates of input embeddings (there are $O(n)$ coordinates); there are "parameters", consisting of coordinates of the positional encodings, elements of the attention matrices and weights of the neural networks (again, there are $O(n)$ of them); any assignment of parameters gives us a "concept" – a set of input vectors, evaluated positively with these values of parameters.

Computations in AHAT on an input from Σ^n can be seen as a sequence of standard arithmetic operations and comparisons with 0 (performed over parameters of our AHATs). Namely, if the values at some level are already computed, we do the following. First, compute the value, the query, and the key vectors for the new layer, and this computation is just a multiplication by elements of the V, K, and Q-matrices of the corresponding layers. Thus, it takes $O(n)$ operations. We then compute n^2 products of key and queries, giving n attention weights per position. For each position, we then compute the set of maximal attention weights, which is doable with n comparisons per position. We take averages ($O(n)$ operations per positions) and apply fixed-size ReLU networks to each position ($O(1)$ operations per positions). Overall, the number of operations is $O(n^2)$ and parameters is $O(n)$.

By Theorem 2.3 in [15], the VC dimension of the corresponding concept class is bounded by $O(n^2) \cdot O(n) = O(n^3)$. [*VC dimension* of a concept class is the maximal size of a set of inputs on which all Boolean functions are realizable by this concept class.] By the Sauer-Shelah lemma [35], the number of Boolean functions realizable on k different input vectors is $k^{O(n^3)}$, as required.

7 Concluding Remarks

We have identified the expressive power of average hard attention transformers with no Positional Encodings (PEs) with permutation-invariant languages representing integer points in semi-algebraic sets, which in turn generalize integer solution sets of multivariate polynomials (a.k.a. Diophantine equations). This result suggests the surprising expressivity of transformers, even in the absence of PEs. Despite this expressivity, our characterization shows that PARITY is not expressible by hard attention transformers with no PEs (but is expressible with PEs, e.g., see [2]). In addition, it also follows that average hard attention transformers with no PEs can express languages that are far complex than established models (including simplified counter automata, Petri nets, and higher-order recursion schemes). In particular, it also implies undecidability of reasoning about languages of transformers with no PEs, solving an open problem [34] on the verification of transformers. Using a recently proven result [46] connecting average hard attention transformers with soft attention transformers with PEs or temperature scaling, we show that the latter can capture solutions to Diophantine equations. Finally, we complement these results by providing a permutation-invariant language that is not expressible by AHAT even with PEs, thus solving an open problem [2] whether the inclusion $\mathsf{AHAT} \subseteq \mathsf{TC}^0$ is strict.

Limitation, Related Work, and Future Work: Formal models of transformers (e.g. see [40]) employ some assumptions that might be rather unrealistic. The first pertains to *unbounded precision*, which appears not only in positional encodings, but also in internal precision during computation. This has hitherto played a crucial role in deriving expressivity results. Results on transformers prohibitting PEs (e.g. [45] and our work) constitute the first step in addressing this modeling limitation. Secondly, one should be mindful that average hard attention mechanism is an approximation of practical transformers that is amenable to theoretical analysis, even though there is evidence [27] that they serve as a good approximation. That said, by employing a recent result [46], our result also yields expressivity of soft attention transformers. Thirdly, although our work focuses on NoPE-UHAT (recognizing only permutation-invariant languages), positional ordering can be recovered by a mild addition of positional masking (see [45]).

Recent results (e.g. [18]) suggest that expressivity results are only a first step towards understanding *trainability*. In particular, *sensitive languages* (e.g. PARITY) are not easily trainable, even though they might be expressible by AHAT. [Intuitively, PARITY is *sensitive* because changing one bit flips membership of a given string. This is not the case for MAJ and SQRT.] This suggests the future avenue of investigating trainable NoPE-AHAT languages, e.g., by employing sensitivity.

Finally, our work focuses on *transformer encoders*. It is known that transformer decoders which use *self-attention mechanisms* (i.e. intermediate generation of tokens) are Turing-complete [33]. This has recently [28] been simplified to the case with no PEs but strict positional masking. Such a result is not comparable to ours since, even with no PEs, decoders can generate some type of positional encodings during intermediate computation. Despite this, that transformer encoders can already capture solutions to Diophantine equations suggests the possibility of an alternative proof of Turing-completeness of transformers with a simpler decoder model. This we leave for future work.

References

[1] Ajtai, M. (1983). Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48.

[2] Barceló, P., Kozachinskiy, A., Lin, A. W., and Podolskii, V. V. (2024). Logical languages accepted by transformer encoders with hard attention. In *ICLR*. OpenReview.net.

[3] Bhattamishra, S., Ahuja, K., and Goyal, N. (2020). On the ability and limitations of transformers to recognize formal languages. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7096–7116. Association for Computational Linguistics.

[4] Bonaert, G., Dimitrov, D. I., Baader, M., and Vechev, M. (2021). Fast and precise certification of transformers. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021, pages 466–481. Association for Computing Machinery.

[5] Brix, C., Bak, S., Johnson, T. T., and Wu, H. (2024). The fifth international verification of neural networks competition (VNN-COMP 2024): Summary and results. *CoRR*, abs/2412.19985.

[6] Chiang, D. and Cholak, P. (2022). Overcoming a theoretical limitation of self-attention. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 7654–7664. Association for Computational Linguistics.

[7] Chistikov, D. (2024). An introduction to the theory of linear integer arithmetic (invited paper). In Barman, S. and Lasota, S., editors, *44th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2024, December 16-18, 2024, Gandhinagar, Gujarat, India*, volume 323 of *LIPIcs*, pages 1:1–1:36. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[8] Clemente, L., Parys, P., Salvati, S., and Walukiewicz, I. (2016). The diagonal problem for higher-order recursion schemes is decidable. In Grohe, M., Koskinen, E., and Shankar, N., editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 96–105. ACM.

[9] Czerwinski, W., Hofman, P., and Zetzsche, G. (2018). Unboundedness problems for languages of vector addition systems. In Chatzigiannakis, I., Kaklamanis, C., Marx, D., and Sannella, D., editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 119:1–119:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[10] Delétang, G., Ruoss, A., Grau-Moya, J., Genewein, T., Wenliang, L. K., Catt, E., Cundy, C., Hutter, M., Legg, S., Veness, J., and Ortega, P. A. (2023). Neural networks and the chomsky hierarchy. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[11] Dong, X., Luu, A. T., Ji, R., and Liu, H. (2021). Towards robustness against natural language word substitutions. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[12] Esparza, J. (1996). Decidability and complexity of petri net problems—an introduction. In *Advanced course on Petri nets*, pages 374–428. Springer.

[13] Furst, M., Saxe, J. B., and Sipser, M. (1984). Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27.

[14] Ginsburg, S. and Spanier, E. (1966). Semigroups, presburger formulas, and languages. *Pacific journal of Mathematics*, 16(2):285–296.

[15] Goldberg, P. and Jerrum, M. (1993). Bounding the vapnik-chervonenkis dimension of concept classes parameterized by real numbers. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 361–369.

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

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

[18] Hahn, M. and Rofin, M. (2024). Why are sensitive functions hard for transformers? In Ku, L., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 14973–15008. Association for Computational Linguistics.

[19] Hao, Y., Angluin, D., and Frank, R. (2022). Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Trans. Assoc. Comput. Linguistics*, 10:800–810.

[20] Hsieh, Y., Cheng, M., Juan, D., Wei, W., Hsu, W., and Hsieh, C. (2019). On the robustness of self-attentive models. In Korhonen, A., Traum, D. R., and Màrquez, L., editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1520–1529. Association for Computational Linguistics.

[21] Huang, X., Ruan, W., Huang, W., Jin, G., Dong, Y., Wu, C., Bensalem, S., Mu, R., Qi, Y., Zhao, X., Cai, K., Zhang, Y., Wu, S., Xu, P., Wu, D., Freitas, A., and Mustafa, M. A. (2023). A survey of safety and trustworthiness of large language models through the lens of verification and validation. *CoRR*, abs/2305.11391.

[22] Ibarra, O. H. (1978). Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM (JACM)*, 25(1):116–133.

[23] Keskin, E. and Meyer, R. (2024). On the separability problem of VASS reachability languages. In Sobocinski, P., Lago, U. D., and Esparza, J., editors, *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2024, Tallinn, Estonia, July 8-11, 2024*, pages 49:1–49:14. ACM.

[24] Marques-Silva, J. and Ignatiev, A. (2022). Delivering trustworthy AI through formal XAI. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 12342–12350. AAAI Press.

[25] Matiyasevich, Y. V. (1993). *Hilbert's Tenth Problem*. MIT Press, Cambridge, Massachusetts.

[26] Mayr, E. W. (1981). An algorithm for the general petri net reachability problem. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 238–246.

[27] Merrill, W., Ramanujan, V., Goldberg, Y., Schwartz, R., and Smith, N. A. (2021). Effects of parameter norm growth during transformer training: Inductive bias from gradient descent. In Moens, M., Huang, X., Specia, L., and Yih, S. W., editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 1766–1781. Association for Computational Linguistics.

[28] Merrill, W. and Sabharwal, A. (2024). The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

[29] Merrill, W., Sabharwal, A., and Smith, N. A. (2022). Saturated transformers are constant-depth threshold circuits. *Trans. Assoc. Comput. Linguistics*, 10:843–856.

[30] Merrill, W., Weiss, G., Goldberg, Y., Schwartz, R., Smith, N. A., and Yahav, E. (2020). A formal hierarchy of RNN architectures. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 443–459. Association for Computational Linguistics.

[31] Ong, L. (2015). Higher-order model checking: An overview. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 1–15. IEEE Computer Society.

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

[33] Pérez, J., Barceló, P., and Marinkovic, J. (2021). Attention is turing-complete. *J. Mach. Learn. Res.*, 22:75:1–75:35.

[34] Sälzer, M., Alsmann, E., and Lange, M. (2025). Transformer encoder satisfiability: Complexity and impact on formal reasoning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

[35] Sauer, N. (1972). On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147.

[36] Shi, Z., Zhang, H., Chang, K., Huang, M., and Hsieh, C. (2020). Robustness verification for transformers. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[37] Shlapentokh, A. (2006). *Hilbert's Tenth Problem: Diophantine Classes and Extensions to Global Fields*. New Mathematical Monographs. Cambridge University Press.

[38] Sipser, M. (2013). *Introduction to the Theory of Computation*. Course Technology, Boston, MA, third edition.

[39] Strobl, L., Angluin, D., Chiang, D., Rawski, J., and Sabharwal, A. (2025). Transformers as transducers. *Transactions of the Association for Computational Linguistics*, 13:200–219.

[40] Strobl, L., Merrill, W., Weiss, G., Chiang, D., and Angluin, D. (2024). What formal languages can transformers express? A survey. *Trans. Assoc. Comput. Linguistics*, 12:543–561.

[41] van Lint, J. H. and Wilson, R. M. (2001). *A Course in Combinatorics*. Cambridge University Press.

[42] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

[43] Vollmer, H. (1999). *Introduction to Circuit Complexity*. Springer.

[44] Weiss, G., Goldberg, Y., and Yahav, E. (2018). On the practical computational power of finite precision rnns for language recognition. In Gurevych, I. and Miyao, Y., editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 740–745. Association for Computational Linguistics.

[45] Yang, A., Chiang, D., and Angluin, D. (2024a). Masked hard-attention transformers recognize exactly the star-free languages. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 10202–10235. Curran Associates, Inc.

[46] Yang, A., Strobl, L., Chiang, D., and Angluin, D. (2024b). Simulating hard attention using soft attention. *CoRR*, abs/2412.09925.

A Omitted proofs from Section 3

A.1 Permutation-invariant languages of simplified multicounter machines

Simplified multicounter machines were first introduced by [44] as a non Turing complete version of Minsky machines that still allow incrementing, decrementing, and (non-)zero tests of multiple counters. Before defining these machines, we first need some more notations: the *masking function* $\operatorname{mask}\colon \mathbb{Z}^d \to \{0,1\}^d$ satisfies for each tuple $\boldsymbol{x} \in \mathbb{Z}^d$

$$\text{for all } 1 \le i \le d\colon\ \operatorname{mask}(\boldsymbol{x})[i] = 0 \iff \boldsymbol{x}[i] = 0\,.$$

Simplified multicounter machines can apply the following operations: incrementing $(+1)$, decrementing (-1), resets $(\cdot\,0)$, and no-operations $(\cdot\,1)$. A vector $\boldsymbol{o} \in \{+1, -1, \cdot\,0, \cdot\,1\}^d$ of operations induces the following function $\boldsymbol{o}\colon \mathbb{Z}^d \to \mathbb{Z}^d$ with: for each $\boldsymbol{x} \in \mathbb{Z}^d$ and $1 \le i \le n$:

- if $\boldsymbol{o}[i] = +1$ then $\boldsymbol{o}(\boldsymbol{x})[i] = \boldsymbol{x}[i] + 1$,
- if $\boldsymbol{o}[i] = -1$ then $\boldsymbol{o}(\boldsymbol{x})[i] = \boldsymbol{x}[i] - 1$,
- if $\boldsymbol{o}[i] = \cdot\,0$ then $\boldsymbol{o}(\boldsymbol{x})[i] = 0$, and
- if $\boldsymbol{o}[i] = \cdot\,1$ then $\boldsymbol{o}(\boldsymbol{x})[i] = \boldsymbol{x}[i]$.

A d-dimensional *simplified multicounter machine* is a tuple $\mathfrak{M} = (Q, \Sigma, q_0, \delta, u, F)$ where Q is a finite set of *states*, Σ is the *input alphabet*, $q_0 \in Q$ is the *initial* state, $\delta\colon Q \times \Sigma \times \{0,1\}^d \to Q$ a *transition function*, $u\colon \Sigma \to \{-1, +1, \cdot\,0, \cdot\,1\}^d$ a *counter update function*, and $F \subseteq Q \times \{0,1\}^d$ a set of masked *accepting configurations*. The set of configurations of $\mathfrak{M}$ is the set $Q \times \mathbb{Z}^d$. For two configurations $(p, \boldsymbol{x}), (q, \boldsymbol{y}) \in Q \times \mathbb{Z}^d$ and a letter $a \in \Sigma$ we write $(p, \boldsymbol{x}) \xrightarrow{a}_{\mathfrak{M}} (q, \boldsymbol{y})$ if $q = \delta(p, a, \operatorname{mask}(\boldsymbol{x}))$, and $\boldsymbol{y} = u(a)(\boldsymbol{x})$. For a word $w \in \Sigma^*$ and configurations $c, d \in Q \times \mathbb{Z}^d$ we also write $c \xrightarrow{w}_{\mathfrak{M}} d$ if there are $a_1, a_2, \ldots, a_\ell \in \Sigma$ and configurations $c_0, c_1, \ldots, c_\ell \in Q \times \mathbb{Z}^d$ with $w = a_1 a_2 \cdots a_\ell$, $c_0 = c$, $c_\ell = d$, and $c_{i-1} \xrightarrow{a_i}_{\mathfrak{M}} c_i$ for all $1 \le i \le \ell$. A word $w \in \Sigma^*$ is *accepted* by $\mathfrak{M}$ iff there is a configuration $(q, \boldsymbol{x}) \in Q \times \mathbb{Z}^d$ with $(q_0, \boldsymbol{0}) \xrightarrow{w}_{\mathfrak{M}} (q, \boldsymbol{x})$ with $(q, \operatorname{mask}(\boldsymbol{x})) \in F$. By $L(\mathfrak{M})$ we denote the set of all accepted words of $\mathfrak{M}$.

Lemma 12. *If $L \subseteq \Sigma^*$ is permutation-invariant and accepted by a simplified multicounter machine. Then the Parikh image of L is semilinear.*

Proof sketch. Suppose $\Sigma = \{a_1, \ldots, a_m\}$. Note that since L is permutation closed, it has the same Parikh image as $K = L \cap a_1^* \cdots a_m^*$. Moreover, K is also accepted by some simplified multicounter machine: Checking that the input belongs to $a_1^* \cdots a_m^*$ can be done in the state. Now note that in a simplified multicounter machine, since every counter update depends only on the input letter, a simplified multicounter machine for the language K is necessarily *reversal-bounded*: This means, over the course of the run, each counter switches between *incrementing*, *decrementing* and *zero-testing* at most $m - 1$ times. However, it is a well-known result in automata theory that counter machines with reversal-bounded counters have semilinear Parikh images [22, Theorem 2.3]. □

We would like to stress that the assumption of being permutation-invariant is crucial in Lemma 12 is crucial: Without it, simplified multi-counter machines can accept non-semilinear languages.

For example, take the language

$$L := \{\mathtt{a}^n((\mathtt{bc})^n(\mathtt{de})^n)^m\mathtt{f}^n \mid n,m \in \mathbb{N}, m,n \geq 2\}.$$

Then L is accepted by a simplified multi-counter machine: It counts up its first counter while reading $\mathtt{a}$'s and thus stores n in it. Upon reading $\mathtt{b}$, it decrements the first counter, and $\mathtt{c}$ increments the second counter. Thus, after reading $(\mathtt{bc})^n$, the first counter is empty and the second counter holds n. Then, $\mathtt{d}$ decrements the second counter, and $\mathtt{e}$ increments the first. Thus, after reading $(\mathtt{de})^n$, we are back at holding n in the first counter (and the second being empty). Finally, $\mathtt{f}$ just decrements the first counter.

However, the language L does not have a semilinear Parikh image: Projecting away the components of the letters $\mathtt{a}, \mathtt{f}$ yields the set of vectors

$$S = \{(b,c,d,e) \in \mathbb{N}^4 \mid \exists m,n \in \mathbb{N}\colon m,n \geq 2,\ b = c = d = e = 4mn\},$$

where b,c,d,e are the entries corresponding to $\mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}$, resp. However, S is not semilinear: If we project to one of the components, we obtain the set $\{4mn \mid m,n \geq 2\}$. If the latter were definable in Presburger arithmetic, then so would $\{mn \mid m,n \geq 2\}$, but this is the set of composite numbers, which cannot be semilinear.

Thus, we observe:

Proposition 13. *There is a simplified multi-counter machine that accepts a non-semilinear language.*

A.2 Permutation-invariant languages of LTL with counting

$\mathsf{LTL}[\mathsf{Count}]$ has the following syntax:

$$\phi ::= a \mid t \leq t \mid \neg\phi \mid \phi \vee \phi \mid \mathrm{X}\,\phi \mid \phi\,\mathrm{U}\,\phi$$
$$t ::= k \mid k \cdot \overleftarrow{\#\phi} \mid k \cdot \overrightarrow{\#\phi} \mid t + t$$

where $a \in \Sigma$ and $k \in \mathbb{Z}$. Next we define the semantics of $\mathsf{LTL}[\mathsf{Count}]$. For any word $w = a_1 a_2 \cdots a_\ell \in \Sigma^*$ with $a_1, a_2, \ldots, a_\ell \in \Sigma$, for each $1 \leq i \leq \ell$, and each formula $\phi \in \mathsf{LTL}[\mathsf{Count}]$ we write $w, i \models \phi$ if the formula ϕ is satisfied in w at position i. Formally, this relation is defined inductively as follows:

- $w, i \models a$ (for $a \in \Sigma$) iff $a_i = a$,
- $w, i \models \neg\phi$ iff $w, i \not\models \phi$,
- $w, i \models \phi \vee \psi$ iff $w, i \models \phi$ or $w, i \models \psi$,
- $w, i \models \mathrm{X}\,\phi$ iff $i < \ell$ and $w, i+1 \models \phi$,
- $w, i \models \phi\,\mathrm{U}\,\psi$ iff there is $i \leq j \leq k$ with $w, j \models \psi$ and for all $i \leq k < j$ we have $w, k \models \phi$,
- $w, i \models t_1 \leq t_2$ iff $[\![t_1]\!](w,i) \leq [\![t_2]\!](w,i)$ where the semantics $[\![t]\!]\colon \Sigma^* \times \mathbb{N} \to \mathbb{Z}$ of a term t is defined as follows: $[\![k]\!](w,i) = k$, $[\![t_1 + t_2]\!](w,i) = [\![t_1]\!](w,i) + [\![t_2]\!](w,i)$, $[\![k \cdot \overleftarrow{\#\phi}]\!] = k \cdot |\{1 \leq j < i \mid w, j \models \phi\}|$, and $[\![k \cdot \overrightarrow{\#\phi}]\!] = k \cdot |\{i \leq j \leq \ell \mid w, j \models \phi\}|$.

Our main result on $\mathsf{LTL}[\mathsf{Count}]$ is the following:

Theorem 14. *Every permutation-invariant language definable in* $\mathsf{LTL}[\mathsf{Count}]$ *has a semilinear Parikh image.*

Before we can prove Theorem 14, we need a few more definitions. For an alphabet Σ write Σ_ε for the set $\Sigma \cup \{\varepsilon\}$. A ($d$-dimensional) *Parikh automaton* is a tuple $\mathfrak{A} = (Q, \Sigma, \iota, \Delta, (C_q)_{q \in Q})$ where Q is a finite set of *states*, Σ is the *input alphabet*, $\iota \in Q$ is an *initial* state, $\Delta \subseteq Q \times \Sigma_\varepsilon \times \mathbb{N}^d \times Q$ is a finite *transition* relation, and $C_q \subseteq \mathbb{N}^d$ are semilinear *target* sets. A word $w \in \Sigma^*$ is *accepted* by $\mathfrak{A}$ if there are $a_1, a_2, \ldots, a_\ell \in \Sigma_\varepsilon$, states $q_0, q_1, \ldots, q_\ell \in Q$, and vectors $\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell \in \mathbb{N}^d$ such that (i) $q_0 = \iota$ and $\boldsymbol{v}_0 = \mathbf{0}$, (ii) for each $0 \leq i < \ell$ there is a transition $(q_i, a_i, \boldsymbol{x}_i, q_{i+1}) \in \Delta$ with $\boldsymbol{v}_{i+1} = \boldsymbol{v}_i + \boldsymbol{x}_i$, and (iii) $\boldsymbol{v}_\ell \in C_{q_\ell}$. The accepted language $L(\mathfrak{A})$ of $\mathfrak{A}$ is the set of all words accepted by $\mathfrak{A}$. It is a well-known fact that for each Parikh automaton $\mathfrak{A}$ the accepted language $L(\mathfrak{A})$

has a semilinear Parikh image. Observe that 0-dimensional Parikh automata are essentially NFA and, hence, accept exactly the regular languages.

A *Parikh transducer* is a Parikh automaton with input alphabet $\Sigma_\varepsilon \times \Gamma_\varepsilon$ where Σ and Γ are two alphabets. The accepted language $L(\mathfrak{A}) \subseteq \Sigma^* \times \Gamma^*$ of a Parikh transducer can also be seen as a map: if $(v, w) \in L(\mathfrak{A})$ then we can see v as the input and w as the output of the transducer. Formally, for an input language $L \subseteq \Sigma^*$ a Parikh transducer computes the output $T_{\mathfrak{A}}(L) = \{w \in \Gamma^* \mid \exists v \in L\colon (v, w) \in L(\mathfrak{A})\}$. If L is accepted by a Parikh automaton then $T_{\mathfrak{A}}(L)$ is also accepted by a Parikh automaton. To see this, we can take the synchronized product of the Parikh automaton $\mathfrak{B}$ accepting L and $\mathfrak{A}$ (i.e., $\mathfrak{B}$ reads the same letter from the input as $\mathfrak{A}$ in its first component). Accordingly, cascading of Parikh transducers is also possible, i.e., if $\mathfrak{A}$ and $\mathfrak{B}$ are Parikh transducers over $\Sigma_\varepsilon \times \Gamma_\varepsilon$ and $\Gamma_\varepsilon \times \Pi_\varepsilon$, we can also construct a Parikh transducer $\mathfrak{C}$ over $\Sigma_\varepsilon \times \Pi_\varepsilon$ computing $T_{\mathfrak{C}} = T_{\mathfrak{B}} \circ T_{\mathfrak{A}}$.

With the definition of Parikh automata and Parikh transducers we are no able to prove Theorem 14.

Proof. Let $\phi \in \mathsf{LTL}[\mathsf{Count}]$ be a formula such that the described language $L(\phi)$ is permutation-invariant. We will prove by induction on the structure of ϕ that the Parikh image of $L(\phi)$ (or actually a *bounded* subset of $L(\phi)$) is semilinear. Here, a language $L \subseteq \Sigma^*$ is *bounded* if there are letters $a_1, a_2, \ldots, a_n \in \Sigma$ with $L \subseteq a_1^* a_2^* \cdots a_n^*$. So, let $a_1, a_2, \ldots, a_n \in \Sigma$ be distinct letters with $\Sigma = \{a_1, a_2, \ldots, a_n\}$. Then $L(\phi) \cap a_1^* a_2^* \cdots a_n^*$ is clearly bounded and has the same Parikh image as $L(\phi)$.

For each subformula ψ of ϕ we construct a Parikh transducer that labels each position satisfying ψ. In the base case, we decorate each letter a by $\boldsymbol{b} \in \{0, 1\}^n$ where $\boldsymbol{b}[i] = 1$ iff $a_i = a$. Note that this transducer handles all atomic formulas $a \in \Sigma$ at once. For $\psi = \chi_1 \vee \chi_2$ we add the decoration $b \in \{0, 1\}$ to each letter where $b = 1$ iff one of the decorations corresponding to χ_1 and χ_2 is 1. There are similar transducers (which do not introduce counters) for the cases $\psi = \neg\chi$, $\psi = \mathrm{X}\,\chi$, and $\psi = \chi_1 \mathrel{\mathrm{U}} \chi_2$. Note that applying these transducers to a bounded language always yields another bounded language.

Now, consider a counting subformula, i.e. $\psi = \sum_{i=1}^{\ell_1} k_i \cdot \overleftarrow{\#\chi_i} + \sum_{i=\ell_1+1}^{\ell_2} k_i \cdot \overrightarrow{\#\chi_i} \leq k$. Observe that the set of positions satisfying ψ is convex in the set of positions satisfying any χ_i. This is true since we consider only a bounded input language. Hence, we can split the input word into three (possibly empty) intervals: (i) the positions at the beginning of the input that do not satisfy ψ, (ii) the positions where all positions satisfying a χ_i also satisfy ψ, and (iii) the positions at the end of the input that do not satisfy ψ. We describe in the following a Parikh transducer with $3 \cdot \ell_2$ many counters - one for each of these three intervals and each formula χ_i. The transducer guesses the three intervals (note that this is non-deterministic), counts positions satisfying a χ_i accordingly, decorates only the positions in the second interval labeled with a χ_i with 1 (and everything else with a 0), and validates in the end our choice of the intervals (via appropriate semilinear target sets ensuring that the equation in ϕ is not satisfied in the first and third interval and is satisfied in the second interval). Clearly, this all can be done in one (non-deterministic) Parikh transducer.

Finally, we have a cascade of (Parikh) transducers decorating each position in a bounded input word with a Boolean value indicating whether ϕ holds in that position. If we use $a_1^* a_2^* \cdots a_n^*$ as input language for our transducers (note that this language is regular) and intersect the output with all words decorated with a 1 in the first position, we obtain a Parikh automaton accepting exactly the language $L(\phi) \cap a_1^* a_2^* \cdots a_n^*$. Since Parikh automata accept only languages with semilinear Parikh image, we infer that $L(\phi) \cap a_1^* a_2^* \cdots a_n^*$ and, hence, $L(\phi)$ have a semilinear Parikh image. □

A.3 Proof that each permutation-invariant language is in TC^0

We assume that $\Sigma = \{a_1, \ldots, a_k\}$. Using majority gates, one can turn a given string w into a Parikh-equivalent word $w' \in a_1^* \cdots a_k^*$, i.e., $\Psi(w) = \Psi(w')$. This essentially amounts to performing counting using TC^0 circuits (see Section 1.4.3 of [43]). Now, observe that there are at most $poly(n)$ many strings in $a_1^* \cdots a_k^*$ of length n. Indeed, each such string corresponds to an ordered integer partition of n into k parts, and there are precisely $\binom{n+k-1}{k-1} = O(n^k)$ many of them [41, Chapter 13]. Each such string can then be treated separately using AC^0 circuits.

B Omitted proofs from Section 4

B.1 Closure under union and intersection

Lemma 15. NoPE-AHAT *is closed under union and intersection.*

Proof. Let $T_A, T_B \in$ NoPE-AHAT, each with k_A and k_B layers, both operating over the same alphabet Σ and employing the endmarker \$. We outline the straightforward construction of $T_{A\cup B}$, which represents the union of the languages accepted by T_A and T_B.

Let $\iota_A : \Sigma \cup \{\$\} \to \mathbb{Q}^m$ denote the embedding of T_A and $\iota_B : \Sigma \cup \{\$\} \to \mathbb{Q}^n$ represent the embedding of T_B. Then, $T_{A\cup B}$ uses the embedding defined by $\iota(a) = \iota_A(a) \parallel \iota_B(a)$, where $\parallel$ stands for the concatenation of vectors, implying that $\iota_A(a)$ is placed atop $\iota_B(a)$. Subsequently, $T_{A\cup B}$ operates as follows: initially, it simulates T_A on the first m dimensions of each vector $\iota(a)$ while carrying the remaining n dimensions through. This is accomplished by employing the k_A layers of T_A, where the query and key matrices are augmented with all-zero rows and columns in dimensions exceeding m. The value matrices of T_A are extended with identity rows and columns, and we exploit the residual connections in the ReLU networks of an attention layer. Apart from these modifications, T_A remains unchanged. Thereafter, $T_{A\cup B}$ simulates T_B on the latter n dimensions using its k_B layers, adjusted similarly. Consequently, for all inputs $w\$$, the vector $\boldsymbol{y}$ related to the endmarker \$ produced in layer $k_A + k_B$ before the ReLU network application is assumed to be given by $\boldsymbol{y}_A \parallel \boldsymbol{y}_B$, where $\boldsymbol{y}_A$ is the corresponding vector produced by T_A after layer k_A but without application of ReLU network $\mathcal{N}_{k_A}$ and $\boldsymbol{y}_B$ is the vector produced by T_B after layer k_B, but without application of ReLU network $\mathcal{N}_{k_B}$. Finally, we use the ReLU network representing $(\mathcal{N}_{k_A})_1 + (\mathcal{N}_{k_B})_1$ in layer $k_A + k_B$, which denotes the sum of the first output dimensions of the final ReLU networks of T_A and T_B, respectively.

For the case of $T_{A\cap B}$, the NoPE-AHAT recognising the intersection of the languages accepted by T_A and T_B, we employ the same construction. However, we use the final ReLU network computing $\min((\mathcal{N}_{k_A})_1, (\mathcal{N}_{k_B})_1)$ as the overall output of $T_{A\cap B}$. Note that $\min(x, y) = y - \max(0, y - x)$, meaning it can be realised by ReLU networks. □

C Omitted proofs from Section 5

C.1 Capturing recursively enumerable languages

In this subsection, we prove Theorem 3. The inclusions $\mathsf{Proj}(\text{NoPE-AHAT}[\leq 2]) \subseteq \mathsf{Proj}(\text{NoPE-AHAT}) \subseteq \mathsf{RE} \cap \mathsf{PI}$ are obvious: Given an NoPE-AHAT, one can clearly decide the membership problem—just run the AHAT. Thus, the languages in NoPE-AHAT are decidable, and hence their projections are recursively enumerable. They are also clearly permutation-invariant, hence contained in $\mathsf{RE} \cap \mathsf{PI}$. Thus, it remains to show the inclusion $\mathsf{RE} \cap \mathsf{PI} \subseteq \mathsf{Proj}(\text{NoPE-AHAT}[\leq 2])$.

Our proof relies on the fact that $\mathsf{RE} \cap \mathsf{PI}$ is precisely the set of projections of solution sets of Diophantine equation systems. The following is a direct consequence of the "MRDP theorem" (also known as the undecidability of integer Diophantine equations) due to Matiyasevich, Robinson, Davis, and Putnam [25]:

Theorem 16. *Let $\Sigma = \{\mathtt{a}_1, \ldots, \mathtt{a}_m\}$. A language $L \subseteq \Sigma^*$ belongs to $\mathsf{RE} \cap \mathsf{PI}$ if and only if there is a $k \in \mathbb{N}$ and a polynomial $p \in \mathbb{Z}[X_1, \ldots, X_{m+k}]$ such that $L = \pi_{\mathtt{a}_1,\ldots,\mathtt{a}_m}(K)$, where*

$$K = \{w \in \{\mathtt{a}_1, \ldots, \mathtt{a}_{m+k}\}^* \mid p(\Psi(w)) = 0\}.$$

Because of this, it suffices to show that every language K as in Theorem 16 belongs to $\mathsf{Proj}(\text{NoPE-AHAT}[\leq 2])$: Indeed, if K belongs to $\mathsf{Proj}(\text{NoPE-AHAT}[\leq 2])$, then L belongs to $\mathsf{Proj}(\text{NoPE-AHAT}[\leq 2])$.

Systems of quadratic inequalities The first step in our construction is to observe every language K as in Theorem 16 can also be defined by quadratic inequalities, if we allow several of them. Formally, a *simple quadratic polynomial* is a polynomial of the form $aWX + bY + cZ + d$, where W, X, Y, Z are pairwise distinct variables, and $a, b, c, d \in \mathbb{Z}$.

Lemma 17. *For every polynomial $p \in \mathbb{Z}[X_1, \ldots, X_m]$, there are $k, t \in \mathbb{N}$ and simple quadratic polynomials $q_1, \ldots, q_m \in \mathbb{Z}[X_1, \ldots, X_{m+k}]$ such that for every $\boldsymbol{x} \in \mathbb{N}^m$, we have $p(\boldsymbol{x}) = 0$ if and only if there is a $\boldsymbol{y} \in \mathbb{N}^k$ with $q_i(\boldsymbol{x}, \boldsymbol{y}) < 0$ for every $i \in [1, t]$.*

Proof. First, observe that it suffices to prove the lemma to reduce to *equations*, i.e. "$q_i(\boldsymbol{x}, \boldsymbol{y}) = 0$", because such an equation can be turned into inequalities by requiring $q_i(\boldsymbol{x}, \boldsymbol{y}) - 1 < 0$ and $-q_i(\boldsymbol{x}, \boldsymbol{y}) - 1 < 0$.

Given the polynomial $p \in \mathbb{Z}[X_1, \ldots, X_m]$, we write $p = M_1 + \cdots + M_r$ with monomials $M_1, \ldots, M_r$. We introduce variables $Y_0, Y_1, \ldots, Y_r, Z_0, \ldots, Z_r$ and equations $Y_0 = 1$, $Z_0 = 0$, and $M_i = Y_i$ and $Z_i = Z_{i-1} + Y_0 Y_i$ for $i \in [1, r]$, and $Z_r = 0$. Thus, we now have a set of equations that are either already simple quadratic, or of the form $Y = M$, where M is a monomial.

To express $Y = M$ for a monomial M, suppose the monomial M is $M = aZ_1 \cdots Z_r$ for variables $Z_1, \ldots, Z_r$, and $a \in \mathbb{Z}$. Then, we introduce variables $Y_1, \ldots, Y_r$ and equations $Y_0 = a$, $Y_i = Y_{i-1} Z_i$, for $i \in [1, r]$, and $Y = Z_r$. Now, all equations are clearly of the form $q_i = 0$, where q_i is simple quadratic. □

Because of Lemma 17, it suffices to prove the following:

Proposition 18. *Let $q_1, \ldots, q_t \in \mathbb{Z}[X_1, \ldots, X_m]$ be simple quadratic polynomials. Then the language $M = \{w \in \{\mathtt{a}_1, \ldots, \mathtt{a}_m\}^* \mid q_i(\Psi(w)) < 0$ for each $i \in [1, t]\}$ belongs to* $\mathsf{Proj}(\mathsf{NoPE\text{-}AHAT}[\leq 2])$.

Suppose we are given t inqualities over the set $\mathcal{X} = \{X_1, \ldots, X_m\}$ of variables. Our input alphabet is $\Sigma = \{\mathtt{a}_1, \ldots, \mathtt{a}_t, \mathtt{b}_1, \ldots, \mathtt{b}_m\}$. Moreover, we have a special endmarker symbol \$. Intuitively, a vector $\boldsymbol{x} \in \mathbb{N}^t$ is represented by an input word where $\mathtt{a}_i$ occurs exactly $\boldsymbol{x}[i]$ times, and each $\mathtt{b}_i$ occurs exactly once. Intuitively, the role of the letters $\mathtt{b}_i$ is that during the run of the transformer, the (unique) occurrence of $\mathtt{b}_i$ will evaluate the i-th inequality. Then clearly, a projection map that deletes all occurrences of $\mathtt{b}_1, \ldots, \mathtt{b}_m$ will yield exactly the desired language M.

Each input leter is encoded into an $(t + m + 1)$-dimensional vector, where $\mathtt{a}_i \mapsto \boldsymbol{e}_i$, $\mathtt{b}_i \mapsto \boldsymbol{e}_{t+i}$, and $\$ \mapsto \boldsymbol{e}_{t+m+1}$. Here $\boldsymbol{e}_j$ is the vector with 1 in coordinate j and 0 everywhere else.

Stage I: Computing frequencies using attention Given an input $w\$ = w_1 \cdots w_{n+1}$, $w_1, \ldots, w_n \in \Sigma$, $w_{n+1} = \$$, we denote $n = |w|$. Thus, each layer processes $n + 1$ vectors. Let $x_i := |w|_{\mathtt{a}_i}$ for each $i \in [1, m]$. In the beginning, we use an attention layer where each position computes the average over *all* positions. In the average vector, we thus obtain $|w|_{\mathtt{a}_i}/(n+1) = x_i/(n+1)$ in component i; and $|w|_{\mathtt{b}_i}/(n+1)$ in component $m+i$; and $1/(n+1)$ in component $m+t+1$. In each position, we keep this average vectors in new components.

After this, each vector has dimension $(m + t + 1) + (m + t + 1)$. We write the vector at position $p \in [1, n+1]$ as $(\boldsymbol{u}_p, \boldsymbol{f})$, where $\boldsymbol{u}_p$ is the original input vector, and $\boldsymbol{f} \in \mathbb{Q}^{m+t+1}$ is the vector of frequencies. Note that the frequency vector is the same at every position.

Stage II: Collecting factors Using ReLU layers, we expand every vector by additional $m + 1$ components. These new vectors $\boldsymbol{v}_p \in \mathbb{Q}^{m+1}$ hold the following:

1. In positions p with $w_p \in \{\mathtt{a}_1, \ldots, \mathtt{a}_t, \$\}$, we have $\boldsymbol{v}_p = \boldsymbol{0}$.
2. In positions p with $w_p = \mathtt{b}_i$ (equivalently, $\boldsymbol{u}_p[t + i] = 1$): If the i-th inequality is $aX_jX_k + bX_\ell + cX_h + d < 0$, then we have $\boldsymbol{v}_p[j] = ax_k/(n+1)$, $\boldsymbol{v}_p[t+1] = (bx_\ell + cx_h + d)/(n+1)$; all other components of $\boldsymbol{v}_p$ are zero.

We achieve this with separate ReLU neural network for each i. Here, if the i-th inequality is $aX_jX_k + bX_\ell + cX_h + d < 0$, then the i-th ReLU neural network will compute in position p the vector $\boldsymbol{w}_{i,p}$ with

$$\boldsymbol{w}_{i,p}[j] = \boldsymbol{u}_p[t + i] \cdot \frac{ax_k}{n+1} = \boldsymbol{u}_p[t + i] \cdot a\boldsymbol{f}[k], \tag{2}$$

$$\boldsymbol{w}_{i,p}[t + 1] = \boldsymbol{u}_p[t + i] \cdot \frac{bx_\ell + cx_h + d}{n+1} = \boldsymbol{u}_p[t + i] \cdot (b\boldsymbol{f}[\ell] + c\boldsymbol{f}[h] + d\boldsymbol{f}[m + t + 1]), \tag{3}$$

and $\boldsymbol{w}_{i,p}$ is zero in all other components. Note that if we manage to compute each $\boldsymbol{w}_{i,p}$, then the desired vectors $\boldsymbol{v}_p$ can be obtained as $\boldsymbol{w}_{1,p} + \cdots + \boldsymbol{w}_{m,p}$. Thus, it suffices to compute each $\boldsymbol{w}_{i,p}$ using a ReLU neural network. Here, the only difficulty is to compute the products $\boldsymbol{u}_p[r] \cdot \boldsymbol{f}[s]$ for some indices $r, s \in [0, t+m+1]$. By exploiting that $\boldsymbol{u}_p[r] \in \{0,1\}$ and $\boldsymbol{f}[s] \in [0,1]$, this can be done using ReLU applications, since then

$$\boldsymbol{u}_p[r] \cdot \boldsymbol{f}[s] = \mathrm{ReLU}(\boldsymbol{f}[s] - (1 - \boldsymbol{u}_p[r])).$$

Indeed, if $\boldsymbol{u}_p[r] = 1$, this expression evaluates to $\boldsymbol{f}[s]$. And if $\boldsymbol{u}_p[r] = 0$, then this expression evaluates to $\mathrm{ReLU}(\boldsymbol{f}[s] - 1) = 0$.

Now each vector has dimension $(m+t+1) + (m+t+1) + (m+1)$. We write the vector at position $p \in [1, n+1]$ as $(\boldsymbol{u}_p, \boldsymbol{f}, \boldsymbol{v}_p)$.

Stage III: Quadratic polynomials as scores We now use an attention layer where each quantity

$$\frac{ax_j x_k + bx_\ell + cx_h + d}{(n+1)^2},$$

for some inequality $aX_jX_k + bX_\ell + cX_h + d < 0$, is computed as a score. To this end, we use affine key and query maps $K, Q \colon \mathbb{Q}^{2(m+t+1)+(m+1)} \to \mathbb{Q}^{m+1}$ where

$$K(\boldsymbol{u}_p, \boldsymbol{f}, \boldsymbol{v}_p) = \left(\tfrac{x_1}{n+1}, \ldots, \tfrac{x_n}{n+1}, \tfrac{1}{n+1}\right), \qquad Q(\boldsymbol{u}_p, \boldsymbol{f}, \boldsymbol{v}_p) = \boldsymbol{v}_p.$$

Note that then for positions $p, q \in [1, n+1]$, we have

1. If $w_q \in \{\mathtt{a}_1, \ldots, \mathtt{a}_m, \$\}$, then $Q(\boldsymbol{u}_q, \boldsymbol{f}, \boldsymbol{v}_q) = \boldsymbol{v}_q = \boldsymbol{0}$, and thus $\langle K(\boldsymbol{u}_p, \boldsymbol{f}, \boldsymbol{v}_p), Q(\boldsymbol{u}_q, \boldsymbol{f}, \boldsymbol{v}_q)\rangle = 0$.
2. If $w_q = \mathtt{b}_i$, and the i-th inequality is $aX_jX_k + bX_\ell + cX_h + d < 0$, then we have

$$\begin{aligned}\langle K(\boldsymbol{u}_p, \boldsymbol{f}, \boldsymbol{v}_p), Q(\boldsymbol{u}_q, \boldsymbol{f}, \boldsymbol{v}_q)\rangle &= \left\langle \left(\tfrac{x_1}{n+1}, \ldots, \tfrac{x_n}{n+1}, \tfrac{1}{n+1}\right), \boldsymbol{v}_q \right\rangle \\ &= \frac{x_j}{n+1} \cdot \frac{ax_k}{n+1} + \frac{1}{n+1} \cdot \frac{bx_\ell + cx_r + d}{n+1} = \frac{ax_jx_k + bx_\ell + cx_h + d}{(n+1)^2}.\end{aligned}$$

Thus, if the i-th inequality is violated, then an occurrence of $\mathtt{b}_i$ will yield a score ≥ 0. If all inequalities are satisfied, an occurrence of $\mathtt{b}_i$ will yield a negative score.

Moreover, the value map $V \colon \mathbb{Q}^{2(m+t+1)+(m+1)} \to \mathbb{Q}$ is defined so that

1. if $w_p \in \{\mathtt{a}_1, \ldots, \mathtt{a}_m, \$\}$, then $V(\boldsymbol{u}_p, \boldsymbol{f}, \boldsymbol{v}_p) = 0$, and
2. if $w_p \in \{\mathtt{b}_1, \ldots, \mathtt{b}_t\}$, then $V(\boldsymbol{u}_p, \boldsymbol{f}, \boldsymbol{v}_p) = 1$.

Thus, as soon as there is an occurrence of $\mathtt{b}_i$ and the i-th inequality is violated, the value 1 participates in the average, and thus the average becomes positive; even $\geq \frac{1}{n+1}$. However, if all inequalities are satisfied, then none of the positions with letters x_i maximize the score, and thus the average value is exactly 0.

We keep this average in a new component. Thus, all vectors are of the form $(\boldsymbol{u}_p, \boldsymbol{f}, \boldsymbol{v}_p, g)$, where $g \in \mathbb{Q}$ is the computed average. Note that the average is the same in every position.

Stage IV: Check all conditions We now use ReLU layers to check all conditions. Note that our input corresponds to a solution if and only if the following conditions are met:

1. $g \leq 0$; which is the case iff $g < \frac{1}{n+1}$.
2. there is exactly one occurrence of each $\mathtt{b}_i$, i.e. $\boldsymbol{f}_p[m+i] = \frac{1}{n+1}$.

Therefore, we compute using ReLU layers the quantity

$$y := \tfrac{1}{n+1} - g \;-\; |\boldsymbol{f}[m+1] - \tfrac{1}{n+1}| \;-\; \cdots \;-\; |\boldsymbol{f}[m+t] - \tfrac{1}{n+1}|.$$

Here, note that since $|z| = \mathrm{ReLU}(z) + \mathrm{ReLU}(-z)$, we can compute absolute values. Moreover, since \$ occurs exactly once, we have $\frac{1}{n+1}$ available, since $\boldsymbol{f}[m+t+1] = \frac{1}{n+1}$.

Now observe that if our input is a solution, then $g = 0$ and $\boldsymbol{f}[m+i] = \frac{1}{n+1}$ or $i \in [1, t]$, and thus $y' = \frac{1}{n+1}$. If our input is not a solution (or some letter $\mathtt{b}_i$ occurs not exactly once), then at least one of the terms g, $|\boldsymbol{f}[m+i] - \frac{1}{n+1}|$ will be $\geq \frac{1}{n+1}$, and thus $y \leq 0$. Therefore, we have $y' > 0$ if and only if out input encodes a solution.

C.2 One layer NoPE-AHAT

Theorem 6. $\mathsf{NoPE\text{-}AHAT}[\leq 1] = \mathsf{QFPA}$.

Proof of Theorem 6. We begin by proving that $\mathsf{NoPE\text{-}AHAT}[\leq 1] \subseteq \mathsf{QFPA}$. Let T be an AHAT with input embedding $\iota : \Sigma \cup \{\$\} \to \mathbb{Q}^d$, a single AHA layer λ utilising affine maps $Q, K \in \mathbb{Q}^{m \times d}$, $V \in \mathbb{Q}^{k \times d}$, given as matrices, and the ReLU network $\mathcal{N} : \mathbb{Q}^{d+k} \to \mathbb{Q}^e$. Our goal is to construct a quantifier-free PA formula φ_T with variables x_i for $i \in \{1, \dots, |\Sigma|\}$ such that $\Psi^{-1}([\![\varphi]\!]) = \{w \in \Sigma^* \mid T \text{ accepts } w\$\}$. In the following, we assume $\Sigma = \{a_1, \dots, a_m\}$ and denote $\Sigma \cup \{\$\}$ by Σ'.

First, we observe that for all words $w \in \Sigma^*$, the output of T given $w\$$ is computed by

$$\mathcal{N}\left(\iota(\$), \frac{1}{|w\$|_{a_{i_1}} + \cdots + |w\$|_{a_{i_h}}} \sum_{j=1}^{h} |w\$|_{a_{i_j}} V\iota(a_{i_j})\right),$$

where $\Gamma = \{a_{i_1}, \dots, a_{i_h}\} \subseteq \Sigma'$ is exactly the subset of symbols a_{i_j} occurring in $w\$$ that maximise $\langle Q\iota(a_{i_j}), K\iota(\$)\rangle$. We construct φ_T such that it mirrors exactly this computational structure. We have $\varphi_T = \bigvee_{\Gamma \subseteq \Sigma'} \varphi_\Gamma$, where $\bigvee$ ranges over those subsets Γ where $\langle Q\iota(a_{i_j}), K\iota(\$)\rangle$ is maximal for precisely the $a_{i_j} \in \Gamma$. The subformula φ_Γ is defined as follows. For now, we assume that $\$ \notin \Gamma$ and introduce some auxiliary formulas. Throughout the following construction steps, we assume that atomic formulas are normalised to the form $c_1 x_1 + \cdots + c_n x_n \leq b$.

Given the ReLU network $\mathcal{N}$, it is straightforward to construct a quantifier-free PA formula $\varphi^{\mathcal{N}}$ such that $[\![\varphi^{\mathcal{N}}]\!]$ exactly includes those $x_1, \dots, x_{d+k} \in \mathbb{N}^{d+k}$ satisfying $\mathcal{N}(x_1, \dots, x_{d+k})_1 > 0$, where $\mathcal{N}(\cdot)_1$ denotes the first output dimension of $\mathcal{N}$. The key idea here is that the computation of a single ReLU node $v(x_1, \dots, x_{d+k}) = y$, with weights c_i and bias b of $\mathcal{N}$, is described by the quantifier-free PA formula: $(c_1 x_1 + \cdots + c_{d+k} x_{d+k} + b \leq 0 \wedge 0 = y) \vee (c_1 x_1 + \cdots + c_{d+k} x_{d+k} + b > 0 \wedge c_1 x_1 + \cdots + c_{d+k} x_{d+k} + b = y)$. Then, by nesting this construction iteratively from the last layer to the first layer of $\mathcal{N}$, and finally replacing $= y$ with > 0 in the atomic formulas related to the first output dimension of $\mathcal{N}$, we achieve the construction of $\varphi^{\mathcal{N}}$. This nesting and replacement also ensures that $\varphi^{\mathcal{N}}$ includes only the variables $x_1, \dots, x_{d+k}$.

Let $\Gamma \subseteq \Sigma$ such that $\Gamma = \{a_{i_1}, \dots, a_{i_h}\}$. Consider the ReLU network $\mathcal{N}$, the value matrix V, and the embedding ι. We construct a quantifier-free PA formula $\varphi_\Gamma^{\mathcal{N},V}$ such that $[\![\varphi_\Gamma^{\mathcal{N},V}]\!]$ exactly includes those $(x_{i_1}, \dots, x_{i_h}) \in \mathbb{N}^h$ satisfying $\mathcal{N}(\iota(\$), \frac{1}{x_{i_1} + \cdots + x_{i_h}} \sum_{j=1}^{h} x_{i_j} V\iota(a_{i_j}))_1 > 0$. To do so, we adjust the formula $\varphi^{\mathcal{N}}$ as described in the following. To account for the fixed input $\iota(\$)$, we replace each occurrence of x_1 to x_d in $\varphi^{\mathcal{N}}$ by the respective entry of $\iota(\$)$. Furthermore, to handle the specific form of the input $\frac{1}{x_{i_1} + \cdots + x_{i_h}} \sum_{j=1}^{h} x_{i_j} V\iota(a_{i_j})$, we first replace each occurrence of x_{d+l} with $l \in \{1, \dots, k\}$ in the already modified $\varphi^{\mathcal{N}}$ by:

$$(v_{l1}\iota(a_{i_1})_1 + \cdots + v_{ld}\iota(a_{i_1})_d)x_{i_1} + \cdots + (v_{l1}\iota(a_{i_h})_1 + \cdots + v_{ld}\iota(a_{i_h})_d)x_{i_h},$$

where v_{lj} are the respective entries of V. Lastly, we replace each atomic constraint $c_1 x_{i_1} + \cdots + c_h x_{i_h} \leq b$ in the adjusted formula with $(c_1 - b)x_{i_1} + \cdots + (c_h - b)x_{i_h} \leq 0$ to adjust for the factor $\frac{1}{x_{i_1} + \cdots + x_{i_h}}$ present in the input.

Now, we define φ_Γ as $\varphi_\Gamma^{\mathcal{N},V,\iota}$. If $\$ \in \Gamma$, we adjust $\varphi_\Gamma^{\mathcal{N},V,\iota}$ slightly. Assuming $\$ = a_{i_j} \in \Gamma$, we replace the variable x_{i_j} with the constant 1 in $\varphi_\Gamma^{\mathcal{N},V,\iota}$. Given this construction, it is clear that $\Psi^{-1}([\![\varphi_T]\!]) = \{w \in \Sigma^+ \mid T \text{ accepts } w\$\}$, as φ_T mimics the computation of T for all possible attention situations Γ.

Next, let φ be a quantifier-free PA with m variables x_1 and k atomic subformulas $c_{j1}x_1 + \cdots + c_{jm}x_m \leq b_j$. We assume, without loss of generality, that all negations $\neg$ occur in front of atomic subformulas. As before, but the other way, we construct AHAT T_φ with a single attention layer such that $\Psi^{-1}(\llbracket\varphi\rrbracket) = \{w \in \{a_1, \dots, a_m\}^* \mid T_\varphi \text{ accepts } w\$\}$. To ease notation, we write vectors as row vectors in the following.

First, we observe the following: there is $o_\varphi \in \mathbb{Q}$ with $o_\varphi > 0$ such that for all atomic subformulas $\psi = c_{j1}x_1 + \cdots + c_{jm}x_m \leq b_j$ of φ and all $\boldsymbol{x} \in \mathbb{N}^m$ that do not satisfy ψ, the inequality $c_1x_1 + \cdots + c_mx_m - b_i \geq o_\varphi$ holds. This follows from the fact that all possible solutions are from $\mathbb{N}^m$ and that there are only finitely many different atomic subformulas in φ. Now, the embedding $\iota : \Sigma \cup \{\$\} \to \{0,1\}^{m+k+1}$ for T_φ is defined by $\iota(a_i) = (0, \dots, 0, 1, 0, \dots, 0)$, where the 1 is located at the ith position, and $\iota(\$) = (0, \dots, 0, b_1, \dots, b_k, o_\varphi)$. Both the query and key matrices, Q and K, are zero matrices of dimension $(m+k+1) \times 1$. Note that this configuration ensures that $\langle Q\boldsymbol{x}', K\boldsymbol{x}\rangle = 0$ for all pairs of vectors $\boldsymbol{x}, \boldsymbol{x}'$. The value matrix V is simply the identity matrix of dimensionality $(m+k+1) \times (m+k+1)$. Overall, this ensures that, given a word $w\$$ of length n, that $\boldsymbol{a}_n$, denoting the attention vector computed for \$ is given by $\frac{1}{n}(|w|_{a_1}, \dots, |w|_{a_m}, b_1, \dots, b_k, o_\varphi)$.

Next, we construct the final FNN $\mathcal{N}_\varphi$. The key idea is that $\mathcal{N}_\varphi$, when given the input $(\iota(\$), \frac{1}{n}|w|_{a_1}, \dots, \frac{1}{n}b_k, \frac{1}{n}o_\varphi)$, satisfies $\mathcal{N}_\varphi(\cdot)_1 > 0$ if and only if $(|w|_{a_1}, \dots, |w|_{a_m})$ is a solution to φ. Indeed, the inputs $\iota(\$)$ are not needed, and thus, we assume $\mathcal{N}$ weights these respective inputs with zero. Therefore, we consider only the expression $\mathcal{N}_\varphi(y_1, \dots, y_{m+k+1})$ from here on. We construct $\mathcal{N}_\varphi$ inductively over the structure of φ.

Consider the case $\varphi = c_{j1}x_1 + \cdots + c_{jm}x_m \leq b_j$. Then, $\mathcal{N}_\varphi$ is given by a single node v computing $v(y_1, \dots, y_{m+k}) = \max(0, y_{m+k+1} - (c_{j1}y_1 + \cdots + c_{jm}y_m - y_{m+j}))$. Correctness is straightforward, as this essentially computes $\max(0, \frac{1}{n}(o_\varphi - (c_{j1}|w|_{a_1} + \cdots + c_{jm}|w|_{a_m} - b_j)))$. Given what we observed for o_φ, we have that $\mathcal{N}_\varphi(\cdot) > 0$ if and only if $(|w|_{a_1}, \dots, |w|_{a_m})$ is a solution of $c_{j1}x_1 + \cdots + c_{jm}x_m \leq b_j$. Next, in the case of $\varphi = \neg(c_{j1}x_1 + \cdots + c_{jm}x_m \leq b_j)$ we simply construct $\max(0, c_{j1}y_1 + \cdots + c_{jm}y_m - y_{m+j})$. We remark that in both cases we have that N_φ outputs at least $\frac{1}{n}o_\varphi$ if the condition to be checked is satisfied. Additionally, we adjust N_φ in these base cases such that the output is at most $\frac{1}{n}o_\varphi$, adding a component computing $\max(y_{m+k+1}, y) = \max(0, x - y_{m+k+1}) + y_{m+k+1}$, where x corresponds to the output of N_φ. Thus, we can assume that N_φ outputs exactly $\frac{1}{n}o_\varphi$ if the condition is satisfied and 0 if not.

Next, consider the case $\varphi = \varphi_1 \vee \varphi_2$, assuming that N_{φ_1} and N_{φ_2} are already given. Then, N_φ simply computes the sum of the outputs of N_{φ_1} and N_{φ_2}. Using the same adjustment as above, we assume that N_φ is such that it outputs exactly $\frac{1}{n}o_\varphi$ if $\varphi_1 \vee \varphi_2$ holds. In case that $\varphi = \varphi_1 \wedge \varphi_2$, again assuming that N_{φ_1} and N_{φ_2} are already given, the FNN N_φ computes the maximum of 0 and the sum of the outputs of N_{φ_1} and N_{φ_2} minus y_{m+k+1}. Given that N_{φ_1} and N_{φ_2} each output exactly $\frac{1}{n}o_\varphi$ if φ_i is satisfied, it is straightforward that N_φ also outputs exactly $\frac{1}{n}o_\varphi$ if and only if $\varphi_1 \wedge \varphi_2$ holds. Given this construction, it is evident that T_φ is such that $\Psi^{-1}(\llbracket\varphi\rrbracket) = \{w \in \{a_1, \dots, a_m\}^* \mid T_\varphi \text{ accepts } w\$\}$. □

D The power of NoPE without end marker

We begin with a formal definition. In the case of *no end marker*, we define the language of an AHAT T over Σ as the set of all $w \in \Sigma^+$ with $T(w) = 1$. In the case of no positional encoding, this yields the class $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}$. Further restriction to uniform attention, uniform-or-tieless, or at most ℓ attention layers, are then captured in the classes $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}[\mathsf{U}]$, $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}[\mathsf{UT}]$, and $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}[\leq \ell]$.

D.1 Overview of results

Let us first outline our results on NoPE-AHAT without end marker.

Our proof of Theorem 3 crucially relies on the presence of an end marker. However, even without an end marker and with two attention layers, we can still go beyond powerful formalisms. Recall that $\mathsf{SQRT} = \{w \in \{\mathsf{a}, \mathsf{b}\}^* \mid |w|_{\mathsf{a}} < |w|/\sqrt{2}\}$.

Theorem 19. SQRT *belongs to* $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}[\leq 2]$*, but is (i) not definable in* $\mathsf{LTL}[\mathsf{Count}]$*, (ii) not accepted by a simplified multi-counter machine, and (iv) not semilinear.*

Again, *semilinear sets* are exactly those sets of natural numbers that are definable in existential Presburger arithmetic.

Perhaps surprisingly, in the absence of an end marker, decidability of the emptiness problem for NoPE AHAT turns out to be equivalent to a major open problem in number theory: We show that it is decidable if and only if solvability of Diophantine equations over the rationals is decidable. The latter is a longstanding open problem [37].

Theorem 20. *The emptiness problem for* $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}$ *is equivalent to solvability of Diophantine equations over the rationals.*

Finally, even without an end marker, one-layer NoPE AHAT can still accept languages beyond the complexity class AC^0, and hence also beyond UHAT (even with positional encoding). Let $\mathsf{MAJ} = \{w \in \{\mathtt{a},\mathtt{b}\}^+ \mid |w|_\mathtt{a} > |w|_\mathtt{b}\}$ be the majority language.

Theorem 21. $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}[\leq 1]$ *contains* MAJ*, a non-regular language that does not belong to* AC^0*. In particular, this language is not accepted by a UHAT even with positional encoding.*

The fact that MAJ is not in AC^0 was shown in [13, Thm. 4.3].

D.2 Expressiveness of NoPE AHAT without end marker

We now consider Theorem 19. To recognize SQRT we use that $w \in \mathsf{SQRT}$ is equivalent to $|w|_\mathtt{a}^2/|w|^2 < \frac{1}{2}$. We encode the letters by $\mathtt{a} \mapsto (1,0)$ and $\mathtt{b} \mapsto (0,1)$. In a first attention layer we compute the frequency $\frac{|w|_\mathtt{a}}{|w|}$ of the letter $\mathtt{a}$ in w while we replace each position of letter $\mathtt{b}$ by 0. In a second attention layer, we square the frequency of $\mathtt{a}$ in each position. In the end, we accept if this value is $< \frac{1}{2}$. To this end, we compute $t = 1/2 - |w|_\mathtt{a}^2/|w|^2$ and accept if and only if $t > 0$.

Lemma 22. $\mathsf{SQRT} \in \mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}[\leq 2]$

Proof. Let us now construct an AHAT without end marker and with two layers SQRT. To this end, we view SQRT as

$$\mathsf{SQRT} = \left\{ w \in \{\mathtt{a},\mathtt{b}\}^+ \;\middle|\; \frac{|w|_\mathtt{a}^2}{|w|^2} < \frac{1}{2} \right\}.$$

We encode $\mathtt{a}$ by $\boldsymbol{a} = (1,0) \in \mathbb{R}^2$ and $\mathtt{b}$ by $\boldsymbol{b} = (0,1) \in \mathbb{R}^2$. Suppose we are given as input a non-empty string over $\{\mathtt{a},\mathtt{b}\}$, with $n_\mathtt{a}$ occurrences of $\mathtt{a}$ and $n_\mathtt{b}$ occurrences of $\mathtt{b}$.

Layer 1. Computing frequencies The first layer performs the following replacement:

$$\begin{aligned} \boldsymbol{a} &\rightsquigarrow \tfrac{n_\mathtt{a}}{n_\mathtt{a}+n_\mathtt{b}} \\ \boldsymbol{b} &\rightsquigarrow 0\,. \end{aligned}$$

Thus, we replace (i) every occurrence of $\mathtt{a}$ with its frequency $\frac{n_\mathtt{a}}{n_\mathtt{a}+n_\mathtt{b}}$ and (ii) every occurrence of $\mathtt{b}$ with the value 0. This is done by choosing the matrices K and Q so that $\langle K\boldsymbol{a}, Q\boldsymbol{a}\rangle = \langle K\boldsymbol{a}, Q\boldsymbol{b}\rangle = 1$, whereas $\langle K\boldsymbol{b}, Q\boldsymbol{a}\rangle = 0$ and $\langle K\boldsymbol{b}, Q\boldsymbol{b}\rangle = 1$, which is easy to achieve. Thus, for input positions holding $\mathtt{a}$, we take the average over all input letters, yielding the attention vector

$$\boldsymbol{v} = \tfrac{n_\mathtt{a}}{n_\mathtt{a}+n_\mathtt{b}} \cdot \boldsymbol{a} + \tfrac{n_\mathtt{b}}{n_\mathtt{a}+n_\mathtt{b}} \cdot \boldsymbol{b} = \left(\tfrac{n_\mathtt{a}}{n_\mathtt{a}+n_\mathtt{b}}, \tfrac{n_\mathtt{b}}{n_\mathtt{a}+n_\mathtt{b}}\right).$$

For input positions holding $\mathtt{b}$, we take the average just over the $\mathtt{b}$ positions, yielding the attention vector $\boldsymbol{b}$. Then, we choose a neural net $\mathcal{N}$ so that $\mathcal{N}(\boldsymbol{a},\boldsymbol{v}) = \frac{n_\mathtt{a}}{n_\mathtt{a}+n_\mathtt{b}}$ and $\mathcal{N}(\boldsymbol{b},\boldsymbol{b}) = 0$ which is indeed the replacement above.

Layer 2. Squaring and spreading The second layer computes the value $\frac{n_\mathtt{a}^2}{(n_\mathtt{a}+n_\mathtt{b})^2}$ on all positions. We achieve this as follows. We choose K and Q so that for all positions i and j, the value $\langle K\boldsymbol{v}_i, Q\boldsymbol{v}_j\rangle$ is 1, thus yielding the attention vector with left-most component

$$\tfrac{n_\mathtt{a}}{n_\mathtt{a}+n_\mathtt{b}} \cdot \tfrac{n_\mathtt{a}}{n_\mathtt{a}+n_\mathtt{b}} + \tfrac{n_\mathtt{b}}{n_\mathtt{a}+n_\mathtt{b}} \cdot 0 = \tfrac{n_\mathtt{a}^2}{(n_\mathtt{a}+n_\mathtt{b})^2}$$

We can then set up a neural net $\mathcal{N}$ that outputs $\frac{n_\mathtt{a}^2}{(n_\mathtt{a}+n_\mathtt{b})^2}$ on all positions.

Finally, another neural net computes $t = \frac{1}{2} - \frac{n_{\mathsf{a}}^2}{(n_{\mathsf{a}}+n_{\mathsf{b}})^2}$ in each position. Then we know that $t > 0$ iff $\frac{n_{\mathsf{a}}^2}{(n_{\mathsf{a}}+n_{\mathsf{b}})^2} < \frac{1}{2}$ iff the input word is in SQRT. Note that the case $t = 0$ is impossible since $\frac{|w|_{\mathsf{a}}}{|w|}$ is always a rational number, but $\frac{1}{\sqrt{2}}$ is irrational. Hence, the constructed AHAT accepts exactly the language SQRT. □

Moreover, we show that SQRT is not semilinear. To this end, we argue that the supremum over all ratios of one entry compared to the entry sum in a semilinear set must either be ∞ or a rational number; whereas for SQRT, this supremum is $1/\sqrt{2}$ and hence irrational.

We make use of the concept of *semilinear representations* of a set $S \subseteq \mathbb{N}^d$, meaning S is represented as a $\bigcup_i U_i$ of sets U_i, each generated by a base vector $a_i \in \mathbb{N}^d$ and period vectors $b_{i,j} \in \mathbb{N}^d$. It is known that such semilinear representations generate all semilinear sets (see [14, Thm. 1.3]).

Lemma 23. *The Parikh image of* SQRT *is not semilinear.*

Proof. For every vector $\boldsymbol{x} = (x_1, x_2) \in \mathbb{N}^2 \setminus \{0\}$, define $\rho(x_1, x_2) := \frac{x_1}{x_1+x_2}$ and for a set $S \subseteq \mathbb{N}^2$, define the quantity $\rho(S) = \sup\{\rho(\boldsymbol{x}) \mid \boldsymbol{x} \in S\} \in \mathbb{R} \cup \{\infty\}$. Now observe that if $S \subseteq \mathbb{N} \times \mathbb{N}$ is semilinear, then $\rho(S)$ is ∞ or a rational number: Indeed, take the maximum value of ρ among all base vectors and period vectors in a semilinear representation for S—this rational number is the supremum of all $\rho(\boldsymbol{x})$ with $\boldsymbol{x} \in S$. However, for the Parikh image $\Psi(\mathsf{SQRT})$ of SQRT, we have $\rho(\Psi(\mathsf{SQRT})) = \frac{1}{\sqrt{2}}$, meaning $\Psi(\mathsf{SQRT})$ cannot be semilinear. □

For Theorem 21, we also begin with a sketch. The language MAJ is clearly contained in QFPA, and thus in $\mathsf{NoPE\text{-}AHAT}[\leq 1] = \mathsf{QFPA}$ (Theorem 6). However, the construction in Theorem 6 reveals that if all inequalities in a given QFPA formula are *homogeneous*, i.e. of the form $c_1x_1+\cdots+c_mx_m < 0$, then the language can even be accepted without an end marker. Essentially, the end marker is used to get access to the frequency $\frac{1}{n+1}$, where n is the length of the input word; however homogeneous inequalities can be expressed purely in terms of input letter frequencies (i.e. without $\frac{1}{n+1}$), yielding membership in $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}[\leq 1]$.

We begin by defining *homogeneous* inequalities as those whiche are of the form $c_1x_1+\cdots+c_mx_m < 0$. Let $\mathsf{QFPA}^{\mathsf{hom}}$ denote the languages of QFPA defined by a quantifier-free PA formula where all inequalities are homogeneous. Considering AHAT with a single layer and without a designated end marker, the construction provided in the proof of Theorem 6 in Section C.2 applies directly to show that $\mathsf{QFPA}^{\mathsf{hom}}$ is captured by $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}[\leq 1]$. The key insight is that all constant terms b_j of all atomic subformulas of some $\varphi \in \mathsf{QFPA}^{\mathsf{hom}}$ are zero, which is respected by the embedding ι of T_φ for each input symbol a_i. We thus have:

Proposition 24. $\mathsf{QFPA}^{\mathsf{hom}} \subseteq \mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}[\leq 1]$.

This allows us to show Theorem 21:

Proof of Theorem 21. The language MAJ clearly belongs to $\mathsf{QFPA}^{\mathsf{hom}}$ and thus by Proposition 24, to $\mathsf{NoPE\text{-}AHAT}^{\neg\mathsf{em}}[\leq 1]$. Moreover, MAJ is well-known not to be in AC^0 [13, Thm. 4.3]. Since all languages accepted by UHAT are in AC^0 [19], we also know that MAJ is not accepted by a UHAT. □

D.3 The emptiness problem without an end marker

Let us begin with an intuition on Theorem 20. In the proof of Theorem 1, the presence of the end marker allows us to assume that among the frequencies $f_i = |w|_{\mathsf{a}_i}/(n+1)$—where n is the input length—the frequency for the end marker always holds $1/(n+1)$. This frequency is used to precisely compute the value the given polynomial, up to a factor $1/(n+1)^d$, where d is the polynomial's degree. Without an end marker, we don't have access to $1/(n+1)$. However, we still know that the frequencies will satisfy $f_1 + \cdots + f_m = 1$ and we can ensure $f_1, \ldots, f_m > 0$. By using similar techniques as above, we can again evaluate the given polynomial on the frequencies. This way, we can reduce the problem of deciding whether a given polynomial p has a rational solution $f_1, \ldots, f_m$ such that $f_1, \ldots, f_m > 0$ and $f_1 + \cdots + f_m = 1$. Additionally, we argue algebraically that deciding whether a given polynomial has such a rational solution is as difficult as deciding whether a given polynomial has any rational solution. The converse reduction uses ideas similar to Proposition 10.

In this subsection, we prove Theorem 20.

Reduction from Diophantine to AHATs. Suppose we are given a diophantine equation $p(x_1, \ldots, x_m) = 0$. First, we need another equation $q(y_1, \ldots, y_k) = 0$ such that the initial one has a rational solution the new one has a rational solution, satisfying $y_1, \ldots, y_r > 0$, $y_1 + \ldots + y_r = 1$. Indeed, consider the following "diophantine equation"

$$p\left(\frac{y_1}{z_1} - \frac{u_1}{v_1}, \ldots, \frac{y_m}{z_m} - \frac{u_m}{v_m}\right) = 0, \tag{4}$$

which we transform to the polynomial form via multiplying by sufficiently large degrees of z_i's and v_i's. On the one hand, if this diophantine equation has a rational solution with $z_i \neq, v_i \neq 0$, the original one also has a rational solution. On the other hand, every rational number x can be represented as $x = \frac{y}{z} - \frac{u}{v}$ for some positive y, z, u, v. Hence, if the original equation has a rational solution, the new one has a solution where every variable is strictly positive. Moreover, we can multiply all variables in this solution by the same constant, thus making their sum equal to 1.

Now, given a diophantine equation $q(y_1, \ldots, y_m) = 0$, we construct an AHAT that accepts at least one word if and only if the diophantine equation has a rational solution satisfying $y_1, \ldots, y_m > 0$ and $y_1 + \ldots + y_m = 1$. The input alphabet will be $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$, where m is the number of variables. It is enough to construct an AHAT that, given an input word w with frequences of letters $f_1, \ldots, f_m$, accepts w if and only if $f_1 > 0, \ldots, f_m > 0$, and $q(f_1, \ldots, f_m) = 0$.

For that, we show that for any monomial in $f_1, \ldots, f_m$ there is an AHAT the computes the value of this monomial in every position. The argument is by induction over the degree of the monomial. The induction base is trivial, with 0 layers we can compute 1/ in every position. Now, take non-zero degree monomial M. Let $M = f_k M_1$, where f_k is a variable and M_1 is a monomial of smaller degree. By the induction hypothesis, there exists an AHAT the computes M_1 in every position. Using a ReLU network, we can compute a sequence

$$\alpha_i = \mathbb{I}\{w_i = \sigma_k\} M_1.$$

Indeed, utilizing the fact that the absolute value of M_1 does not exceed 1, we can write:

$$\alpha_i = ReLU(M_1 - \mathbb{I}\{w_i \neq \sigma_k\})$$

Taking the average of α_i, we obtain M_1 times the frequency of the letter σ_k, that is $f_k M_1 = M$, as required.

In this way, we can calculate $f_1 \cdot \ldots \cdot f_m$ and $q^2(f_1, \ldots, f_m)$ in every position. We have to accept the word if the first quantity is strictly positive and the second one is not positive.

To do this, we prove the following lemma: there is an AHAT that, given a number x, bounded by an absolute constant C in the absolute value, output 1 if $x > 0$ and 0 otherwise. We apply it first to $x = f_1 \ldots f_m$ and then to $x = q^2(f_1, \ldots, f_m)$, and accept if and only of the first output minus second output $> 1/2$. We can bound the absolute value of $q^2(f_1, \ldots, f_m)$ computably because this is a fixed polynomial, taken on inputs, not exceeding 1 in the absolute value.

Finally, we prove the lemma. First, having x, we can compute $y = ReLU(x)$, and it remains to distinguish the case y is strictly positive from the case y is 0. For any input letter σ, we can then compute the sequence:

$$\alpha_i = \mathbb{I}\{w_i = \sigma\} y = ReLU(y - C \cdot \mathbb{I}\{w_i \neq \sigma\}).$$

We can then consider an attention which is for position i is equal to α_i. If $y > 0$, this will give us a uniform distribution over positions with σ, and if $y = 0$, this will give us the uniform distribution over all position. Taking the average over the indicator sequence $\mathbb{I}\{w_i = \sigma\}$, we obtain 1 in the first case and f_i in the second case. Doing this for all input letters and summing the results, we obtain some quantity z which is m if $y > 0$ and 1 if $y = 0$, and it remains to output $ReLU(z - (m - 1))$,

Reduction from AHATs to Diophantine. For a given AHAT we construct a disjunction of systems of diophantine equations and inequalities such that, this AHAT accepts a given word $w = w_1 \ldots w_m$ if and only if frequences of letter in this word satisfy one of the systems.

Why is this enough? We can check feasibility of every system separately. Now, how to reduce a system of diophantine equations and inequalities to a single equation? If there are just equalities, this is easy using sum of squares of equations. Now, inequalities can be dealt with thanks to the Lagrange's four-square theorem, implying that every non-negative rational number can be written as the sum of 4 squares of rationals. That is, we replace every inequality $p \geq 0$ into equality $p = x^2 + y^2 + z^2 + w^2$ for fresh variables x, y, z, w. A strict inequality $p > 0$ can be simulated by $(1 + x^2 + y^2 + z^2 + w^2)p = 1 + a^2 + b^2 + c^2 + d^2$.

Next, observe that the value of a token in an AHAT without positional encoding determined by the input letter (there is no way we can distinguish two tokens with the same input letter). Thus, we can define $x_\sigma^{(\ell)}$, the value in a token with input letter $\sigma \in \Sigma$ after ℓ layers. The values, keys, and queries at this level are linear functions of $x_\sigma^{(\ell)}$. The outcome of the layer is determined by finitely many comparisons of expressions of the form $k_{\sigma_1}^{(\ell)} q_{\sigma_2}^{(\ell)}$. Under any fixations of the results of these comparisons, attention vectors $a_\sigma^{(\ell)}$ become polynomials of $x_\sigma^{(\ell)}$ and fractions of the letter. Indeed, for each letter $\sigma \in \Sigma$ there is an argmax set $S \subseteq \Sigma$ such that:

$$a_\sigma^{(\ell)} = \frac{\sum\limits_{\delta \in S} n f_\sigma v_\sigma^{(\ell)}}{\sum\limits_{\delta \in S} n f_\sigma},$$

and n, input lenght, cancels out. Then we pass $a_\sigma^{(\ell)} + x_\sigma^{(\ell)}$ to a fixed neural net $\mathcal{N}$, where under fixing results of comparisions of neurons, everything is linear.

As a result, we partition the whole space of possible values of variables into finitely many systems of polynomial inequalities such that for each system, values of tokens are fixed polynomial expressions. It remains to add to each system a statement that the first coordinate of the last token is positive.