Simulating Hard Attention Using Soft Attention

Andy Yang
University of Notre Dame, USA
ayang4@nd.edu

Lena Strobl
Umeå University, Sweden
lena.strobl@umu.se

David Chiang
University of Notre Dame, USA
dchiang@nd.edu

Dana Angluin
Yale University, USA
dana.angluin@yale.edu

Abstract

We study conditions under which transformers using soft attention can simulate hard attention, that is, effectively focus all attention on a subset of positions. First, we examine several subclasses of languages recognized by hard-attention transformers, which can be defined in variants of linear temporal logic. We demonstrate how soft-attention transformers can compute formulas of these logics using unbounded positional embeddings or temperature scaling. Second, we demonstrate how temperature scaling allows softmax transformers to simulate general hard-attention transformers, using a temperature that depends on the minimum gap between the maximum attention scores and other attention scores.

1 Introduction

A central element of transformers (Vaswani et al., 2017) is *attention*, in which each position computes a weighted average of values from all unmasked positions. In standard attention, which we call *soft* attention, the attention weights are computed by the softmax function and cannot be exactly 0 (unless there is masking) or exactly 1 (unless there is only one position). This is appropriate in applications like machine translation (Bahdanau et al., 2015), where attention models the often fuzzy relationship of target words to source words. But there is a tension between soft attention and discrete tasks like integer arithmetic, algebra, logical reasoning, finite automata, Turing machines, and computer programs. Researchers trying to show that transformers can solve such tasks have often turned to other definitions of attention.

Hahn (2020) defined what is now called *unique-hard attention*, in which all attention is on a single position attaining the maximum attention score, and zero attention is paid to all other positions. Yang et al. (2024) proved that unique-hard attention transformers (UHATs) with no position information besides masking recognize exactly the *star-free regular languages*, and equivalences for other variants as well. Although this means that UHATs can solve some interesting problems like the multi-query associative recall task (Friedman et al., 2023), the star-free regular languages are a rather restricted class, so one might expect that UHATs are less expressive than standard soft-attention transformers (SMATs). But, perhaps surprisingly, it has not been shown previously that UHATs can be simulated by SMATs.

Pérez et al. (2019), to prove that transformers with chain-of-thought are Turing-complete, introduced what is now called *average-hard attention* (or *saturated attention*): given a score vector, equal attention is paid to each position in which the maximum attention score is attained, and zero attention is paid to all other positions. Average-hard attention transformers (AHATs) have been constructed to solve a variety of discrete tasks: matching parentheses (Yao et al., 2021), simulating an n-gram model (Svete and Cotterell, 2024), simulating linear temporal logic with counting terms (Barceló et al., 2024), simulating the programming language S-RASP (Strobl et al., 2025), and others. For many of these tasks, no construction using soft attention is known.

In this paper, we contribute new results that help to answer the question: *Under what conditions can soft attention simulate hard attention?* We are interested in simulations of hard attention that are *general*—that is, for a class of hard attention transformers, not just for a particular

Transactions of the Association for Computational Linguistics, vol. 14, pp. 147–166, 2026. https://doi.org/10.1162/TACL.a.597
Action Editor: Daichi Mochihashi. Submission batch: 1/2025; Revision batch: 7/2025; Published 1/2026.

language—and *parameter-uniform*, that is, the number and values of the parameters do not depend on the input length n. This ensures that the same model and parameters can be used for arbitrary n.

However, a result of Hahn (2020, Lemma 5) stands in the way. If a transformer has (1) soft attention, (2) parameters whose number and values do not depend on n, (3) Lipschitz-continuous position-wise operations, and (4) bounded position embeddings (PEs), then a change in one input symbol results in only an $O(1/n)$ change in any output activation. This is in contrast to AHATs, where a change in one input symbol can result in a $\Theta(1)$ change in an output activation.

Why is this property important? For example, Chiang and Cholak (2022) construct a SMAT for PARITY (deciding whether the number of 1's in a string is odd), but as n increases, the gap between the output for acceptance and rejection approaches 0, as it must. However, this is a problem for composability: If one module has discrete output values that are separated by $O(1/n)$, and another module expects discrete inputs that are separated by $\Theta(1)$, then the two modules cannot be composed.

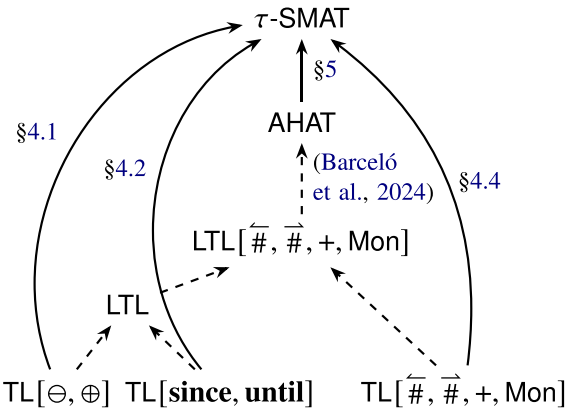
The solutions that we are aware of each circumvent Hahn's lemma by dropping one of the above assumptions:

(1) Temperature scaling, in which attention scores are scaled by a function of n before the softmax. Chiang and Cholak (2022) and Nakanishi (2025) both experiment with $O(1/\log n)$ temperature and find that it can improve length generalization.

(2) Rounding, used by Sanford et al. (2024) to correct approximation errors. They round to $\Theta(\log n)$ bits, which requires a feedforward network with depth or width depending on n (Lemma 27), for a construction that is general, but not parameter-uniform. Similarly, Li et al. (2024) round to constant precision, but the embedding size depends on n.

(3) Layer normalization, as originally defined (Ba et al., 2016), is not Lipschitz-continuous, and can be used to increase sensitivity (Chiang and Cholak, 2022; Yao et al., 2021; Merrill et al., 2024; Yang and Chiang, 2024).

(4) Unbounded position embeddings, whose norm may grow as a function of n.



	Simulatee	Temp. $\tau(n)$	PE
§4.1	TL[⊖, ⊕]	$O(1/n)$ $O(1)$	$O(1)$ $O(n)$
§4.2	TL[**since**, **until**]	$O(1/n)$ $O(1)$	$O(1)$ $O(n)$
§4.4	TL[←#, →#, +, Mon]	$O(1/n)$ $O(1)$	$O(n^2)$ $O(n^3)$
§5	AHAT	See Theorem 26	

Figure 1: Simulation by τ-SMAT. Solid arrows denote results proved in this paper. Key: τ = temperature; PE = position embedding.

Here, we consider general and parameter-uniform simulations of hard attention by soft attention, using unbounded position embeddings or temperature scaling (methods (1) and (4) above). Our results are summarized in Figure 1.

As mentioned above, Barceló et al. (2024) proved that linear temporal logic with counting, or LTL[←#, →#, +, Mon], can be simulated by an AHAT. Thus, LTL[←#, →#, +, Mon] defines a subclass of the languages recognized by AHATs, and various fragments of LTL[←#, →#, +, Mon] define smaller subclasses. Our first set of results (Section 4) examines several such subclasses—among them, the languages recognized by UHATs—and shows that they can be simulated by SMATs using either inverse polynomial temperature or polynomially bounded position embeddings.

Our second result (Section 5) concerns general AHATs and their direct simulation by SMATs. We classify AHATs according to the *gap* $\gamma(n)$ that separates, for inputs of length n, the maximum attention score or scores from all other scores (Edelman et al., 2022). In the literature on

transformer expressivity, constructions of AHATs generally have gap $\gamma(n) \in \Omega(1/n^k)$ for small constants k (Section 6). We prove that any AHAT with gap $\gamma(n)$ can be approximately simulated by a SMAT with the same parameters, using soft attention scaled by a temperature not much lower than $\gamma(n)$.

The implications of our results for the expressivity and learnability of transformers in practice are discussed in Section 7.

2 Preliminaries

2.1 Notation

We write $[n]$ for the set $\{1, \ldots, n\}$. For any Boolean value b, we let $\mathbb{I}[b] = 1$ if b is true and 0 if b is false.

If X and Y are sets, we write X^+ for the set of non-empty sequences over X, and we write $f\colon X^+ \xrightarrow{\text{lp}} Y^+$ to state that f is *length-preserving*, that is, it maps a sequence $(x_1, \ldots, x_n) \in X^+$ to a sequence of the same length $(y_1, \ldots, y_n) \in Y^+$.

If $\mathbf{A}$ is a matrix, we write $\mathbf{A}_{i,*}$ for the i-th row of $\mathbf{A}$ and $\mathbf{A}_{*,j}$ for the j-th column of $\mathbf{A}$.

2.2 Transformers

A *transformer* is a neural network that, in this paper, defines a length-preserving mapping from strings to strings. An input layer maps a string to a sequence of vectors. Then the network processes sequences of vectors through multiple layers, each consisting of a self-attention sublayer followed by a feed-forward sublayer. Finally, an output layer maps the sequence of vectors to a string.

Input Let Σ be an input alphabet, and let $\mathbf{w} = w_1 w_2 \cdots w_n \in \Sigma^+$ be an input sequence of length n. Each token w_i is mapped to a vector $\mathbf{x}_i = \text{WE}(w_i) + \text{PE}_n(i)$, where $\text{WE}\colon \Sigma \to \mathbb{R}^d$ is a *word embedding* and $\text{PE}_n\colon \mathbb{N} \to \mathbb{R}^d$ a *position embedding*. Thus the input sequence is represented as a sequence of vectors $(\mathbf{x}_1, \cdots, \mathbf{x}_n) \in (\mathbb{R}^d)^+$.

Transformer Layers We initialize $\mathbf{h}_i^{(0)} = \mathbf{x}_i$ for all i. For each layer $\ell = 1, \ldots, L$ and position $i = 1, \ldots n$, we compute

$$\mathbf{c}_i^{(\ell)} = \text{SA}^{(\ell)}\left(\mathbf{h}_1^{(\ell-1)}, \ldots, \mathbf{h}_n^{(\ell-1)}\right)_i + \mathbf{h}_i^{(\ell-1)}$$

where $\text{SA}^{(\ell)}$ is the *self-attention* function defined below in Definition 1, with parameters $\mathbf{W}^{(\text{Q},\ell)}$, $\mathbf{W}^{(\text{K},\ell)}$, $\mathbf{W}^{(\text{V},\ell)}$, and weighting function $\mathcal{S}^{(\ell)}$. Then we apply a *feed-forward network* (FFN):

$$\mathbf{h}_i^{(\ell)} = \mathbf{W}_2^{(\ell)}\, \sigma\left(\mathbf{W}_1^{(\ell)}\, \mathbf{c}_i^{(\ell)} + \mathbf{b}_1^{(\ell)}\right) + \mathbf{b}_2^{(\ell)} + \mathbf{c}_i^{(\ell)}$$

where σ is the ReLU activation function, $\mathbf{W}_1^{(\ell)} \in \mathbb{R}^{d_\text{f} \times d}$, $\mathbf{W}_2^{(\ell)} \in \mathbb{R}^{d \times d_\text{f}}$, $\mathbf{b}_1^{(\ell)} \in \mathbb{R}^{d_\text{f}}$, and $\mathbf{b}_2^{(\ell)} \in \mathbb{R}^d$.

The $+\,\mathbf{h}_i^{(\ell-1)}$ and $+\,\mathbf{c}_i^{(\ell)}$ terms above are called *residual connections*. In our constructions, they are useful for making values computed at earlier steps to be available at all later steps.

Transformer layers are usually implemented with *layer normalization*, but we omit this here, as this is a separate method (method (3) in Section 1) for overcoming limitations of soft attention, which may interact in complicated ways with temperature scaling and unbounded PEs.

Output After L layers, the transformer produces an output sequence $(\mathbf{y}_1, \ldots, \mathbf{y}_n) \in (\mathbb{R}^d)^+$, where $\mathbf{y}_i = \mathbf{h}_i^{(L)}$ for all i. To map these vectors to output symbols, we assume, as is common in theoretical research, that each $\mathbf{y}_i$ is exactly the embedding of a symbol in the output alphabet. To accept or reject strings, we look at the output symbol at position n and designate two symbols as accept and reject decisions. Note that this implies a fixed-size gap between the outputs for acceptance and rejection.

Self-Attention A self-attention layer computes weighted sums of *value* vectors at all positions, where the weights are determined by *query* and *key* vectors.

Definition 1 (Self-Attention)**.** A self-attention layer with linear transformations

$$\mathbf{W}^{(\text{Q})}, \mathbf{W}^{(\text{K})}\colon \mathbb{R}^d \to \mathbb{R}^{d_\text{k}}$$
$$\mathbf{W}^{(\text{V})}\colon \mathbb{R}^d \to \mathbb{R}^d$$

and a weighting function

$$\mathcal{S}\colon \mathbb{R}^+ \xrightarrow{\text{lp}} \mathbb{R}^+$$

is a function

$$\text{SA}\colon (\mathbb{R}^d)^+ \xrightarrow{\text{lp}} (\mathbb{R}^d)^+$$
$$\text{SA}\,(\mathbf{h}_1, \ldots, \mathbf{h}_n) = (\mathbf{c}_1, \ldots, \mathbf{c}_n)$$

where, for positions i and j:

$$\mathbf{q}_i = \mathbf{W}^{(\text{Q})}\mathbf{h}_i \qquad \mathbf{k}_j = \mathbf{W}^{(\text{K})}\mathbf{h}_j \qquad \mathbf{v}_j = \mathbf{W}^{(\text{V})}\mathbf{h}_j$$

$$s_{ij} = \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_\text{k}}} \qquad \alpha_{i,*} = \mathcal{S}(s_{i,*}) \qquad \mathbf{c}_i = \sum_{j=1}^{n} \alpha_{ij}\mathbf{v}_j.$$

We call the $\mathbf{q}_i$, $\mathbf{k}_j$, and $\mathbf{v}_j$ the *queries*, *keys*, and *values*, respectively, and the s_{ij}, α_{ij}, and $\mathbf{c}_i$ the *scores*, *weights*, and *outputs*, respectively.

We consider only single-head attention layers for simplicity, as a k-head attention layer can be simulated using k layers of single-head attention (each layer computes the value of one head, and residual connections are used to accumulate the sum of the heads).

Masking In *future-masked attention*, the scores are redefined to

$$s_{ij} = \begin{cases} \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_k}} & j \le i \\ -\infty & \text{otherwise} \end{cases}$$

and we define $\exp(-\infty) = 0$. Similarly, in *past-masked attention*, we have $s_{ij} = -\infty$ iff $j < i$.

Some of our proofs construct transformers that use both future-masked and past-masked attention (also cf. Yao et al., 2021). In practice, mixed attention masks have been employed by applying different masks to different attention heads (Shen et al., 2018) or by applying multiple masks with learnable weights (McDonald and Chiang, 2021; Sible and Chiang, 2024).

Weighting Function The weighting function $\mathcal{S}\colon \mathbb{R}^+ \xrightarrow{\text{lp}} \mathbb{R}^+$ computes the attention weights $\alpha_{i,*}$ based on the attention scores $s_{i,*}$. A common choice is the *softmax* function:

$$[\text{softmax}(s_1, \ldots, s_n)]_j = \frac{e^{s_j}}{\sum_{k=1}^n e^{s_k}}.$$

In hard attention, the attention weights are assigned to focus all attention on the maximum-scoring position or positions.

Definition 2 (Hardmax)**.** The leftmost, rightmost, and average-hardmax functions are defined as:

$$\begin{aligned} I(\mathbf{s}) &= \{i \in [|\mathbf{s}|] \mid s_i = \max \mathbf{s}\} \\ [\text{lhardmax}\,\mathbf{s}]_i &= \mathbb{I}[i = \min I(\mathbf{s})] \\ [\text{rhardmax}\,\mathbf{s}]_i &= \mathbb{I}[i = \max I(\mathbf{s})] \\ [\text{ahardmax}\,\mathbf{s}]_i &= \frac{1}{|I(\mathbf{s})|}\mathbb{I}[i \in I(\mathbf{s})]. \end{aligned}$$

The lhardmax and rhardmax functions return a one-hot vector with a 1 at the position of the leftmost or rightmost maximal element, respectively. The ahardmax function pays equal attention to all the maximal elements.

Temperature Scaling We also consider scaling the attention scores by an inverse *temperature* before applying the softmax.

Definition 3 (Temperature-Dependent Softmax)**.** For a temperature $\tau > 0$, the temperature-dependent softmax function $\text{softmax}_\tau\colon \mathbb{R}^+ \xrightarrow{\text{lp}} \mathbb{R}^+$ is defined as:

$$[\text{softmax}_\tau(s_i, \ldots, s_n)]_j = \frac{e^{s_j/\tau}}{\sum_{k=1}^n e^{s_k/\tau}}.$$

When $\tau = 1$, this reduces to standard softmax.

Definition 4. We categorize transformers by the weighting functions used in their attention layers:

SMAT	uses	softmax
τ-SMAT	uses	softmax_τ
AHAT	uses	ahardmax
UHAT	uses	both lhardmax and rhardmax.

3 Basic Approximations

In this section, we introduce some techniques that will be used throughout the paper. These techniques all serve to bound and correct the difference between hard attention and soft attention.

Definition 5. For any vector of attention scores $\mathbf{s} \in \mathbb{R}^+$, let $s_{\max} = \max_i s_i$. We say that $\mathbf{s}$ is *tieless* if $|\{i \mid s_i = s_{\max}\}| = 1$, and $\mathbf{s}$ has *gap* γ if for all i such that $s_i \neq s_{\max}$, we have $s_i \le s_{\max} - \gamma$.

If $\mathbf{s}$ is tieless, then $\text{lhardmax}\,\mathbf{s} = \text{rhardmax}\,\mathbf{s} = \text{ahardmax}\,\mathbf{s}$, and we simply write $\text{hardmax}\,\mathbf{s}$ for all three.

Edelman et al. (2022) prove a bound on the difference between softmax and ahardmax for attention scores with gap γ. Here, we prove several stronger bounds under stronger assumptions.

3.1 Ziggurat Attention Scores

First, we give a bound for attention scores that are not only tieless with gap γ, but also decrease by at least γ at every position in a ''ziggurat'' pattern (Figure 2).

Lemma 6. *Suppose an attention layer has scores* $\mathbf{s} = (s_1, \ldots, s_n)$*, and there is a* $j^* \in [n]$ *such that for all* $j \in [n]$*,* $s_j \le s_{j^*} - |j - j^*|\gamma$*. Let* $v_1, \ldots, v_n$ *be the attention values, and let* $v_{\max} = \max_j |v_j|$*.*

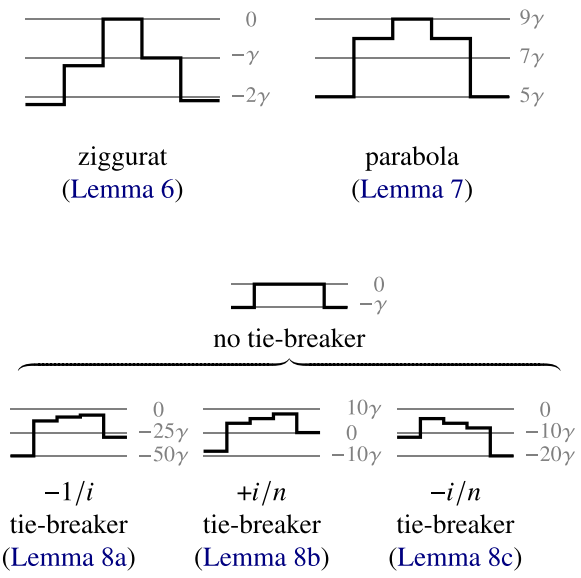


Figure 2: Illustration of the various attention score patterns in Section 3.

Then the difference between the hard and soft attention outputs is

$$\left|\sum_{j=1}^{n}[\text{hardmax } \mathbf{s} - \text{softmax } \mathbf{s}]_j v_j\right| \leq 4e^{-\gamma} v_{\max}.$$

Proof. See Appendix B.1. □

3.2 Table Lookup

Some constructions of hard-attention transformers use a *table lookup* operation (Barceló et al., 2024; Strobl et al., 2025): given $t_i \in [n]$ for $i \in [n]$, we want to retrieve the value v_{t_i}. This is accomplished using a parabola-shaped pattern that peaks at position t_i (Figure 2). Under hard attention, each position i attends solely to position j such that $j = t_i$. To replicate this using soft attention, we must approximate it. When $|v_j|$ is bounded, Lemma 6 suffices to bound the approximation error, but in one case, we have $v_j = j$, so we need the following lemma.

Lemma 7. *Let $\gamma \geq 1$ and let t and n be integers such that $1 \leq t \leq n$. For $j \in [n]$, define $s_j = \gamma(2tj - j^2)$. Then*

$$\left|\sum_{j=1}^{n}[\text{hardmax } \mathbf{s} - \text{softmax } \mathbf{s}]_j j\right| \leq \tfrac{3}{2}e^{-\gamma}.$$

Proof. See Appendix B.2. □

3.3 Tie-Breaking

Next, we show some special cases of Lemma 6, which are useful when using soft attention to simulate unique-hard attention. In unique-hard attention, if two positions are tied for the maximum score, the tie needs to be broken to the left or the right. To simulate this using soft attention, we need to add a *tie-breaking* term, allowing softmax to approximate rhardmax or lhardmax sufficiently closely.

There are three variations, depending on whether the attention to be approximated is rhardmax or lhardmax and on whether the sequence length n is known or not (Figure 2).

Lemma 8. *Let $s_1, \ldots, s_n$ be attention scores with gap at least 1. Let $v_1, \ldots, v_n$ be attention values, and let $v_{\max} = \max_j |v_j|$. Then, for any $\gamma > 0$:*

(a) If $\hat{s}_j = \gamma n^2 \left(s_j - \frac{1}{j}\right)$, *then*

$$\left|\sum_{j=1}^{n}[\text{rhardmax } \mathbf{s} - \text{softmax } \hat{\mathbf{s}}]_j v_j\right| \leq 4e^{-\gamma} v_{\max}.$$

(a) If $\hat{s}_j = 2\gamma n \left(s_j + \frac{j}{2n}\right)$, *then*

$$\left|\sum_{j=1}^{n}[\text{rhardmax } \mathbf{s} - \text{softmax } \hat{\mathbf{s}}]_j v_j\right| \leq 4e^{-\gamma} v_{\max}.$$

(a) If $\hat{s}_j = 2\gamma n \left(s_j - \frac{j}{2n}\right)$, *then*

$$\left|\sum_{j=1}^{n}[\text{lhardmax } \mathbf{s} - \text{softmax } \hat{\mathbf{s}}]_j v_j\right| \leq 4e^{-\gamma} v_{\max}.$$

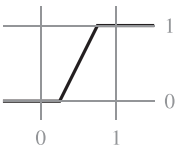
Proof. By Lemma 6; see Appendix B.3. □

3.4 Approximate Booleans

The output of hard attention is often interpreted as a Boolean value: 0 for false and 1 for true. If we approximate hard attention using soft attention, we get an approximate Boolean value. For any $\delta < 1/2$, we treat any number in $(-\infty, \delta]$ as false and any number in $[1 - \delta, \infty)$ as true. For concreteness, we set $\delta = 1/4$. A transformer can convert approximate Booleans into exact ones using a FFN that rounds them to 0 or 1.

Lemma 9. *We can compute the following function with a FFN.*

$$f(x) = \begin{cases} 0 & x \le \frac{1}{4} \\ 2x - \frac{1}{2} & \frac{1}{4} \le x \le \frac{3}{4} \\ 1 & \frac{3}{4} \le x. \end{cases}$$



Proof. Any function that is continuous piecewise affine with a finite number of pieces can be computed by a FFN. Function f can be computed by

$$f(x) = 2\,\mathrm{ReLU}(x - \tfrac{1}{4}) - 2\,\mathrm{ReLU}(x - \tfrac{3}{4}). \qquad \square$$

3.5 First and Last Position

Finally, we show how to approximately mark the first and last positions of the input sequence and then correct the marker to an exact Boolean value.

Proposition 10. *A* SMAT with a PE including $(-1)^i$ can compute the following functions $f\colon \Sigma^+ \xrightarrow{\text{lp}} [0,1]^+$:

(a) $f(\mathbf{w}) = (1, 0, \ldots, 0, 0)$ *using future-masking*

(b) $f(\mathbf{w}) = (0, 0, \ldots, 0, 1)$ *using past-masking*

(c) $f(\mathbf{w}) = (1/1, 1/2, \ldots, 1/(n-1), 1/n)$ *using future-masking*

(d) $f(\mathbf{w}) = (1/n, 1/(n-1), \ldots, 1/2, 1/1)$ *using past-masking*.

Proof. To mark the first position (a), use uniform future-masked attention with values $v_j = (-1)^{j+1}$. If $i = 1$, then all attention is on position 1, and the output value is 1. If $i > 1$, then the attention output is at most $1/3$. So a FFN, similar to the one in Lemma 9, can map all such values to 0. To mark the last position (b), we proceed as above, but using past-masked attention. If $i = n$, then the attention output is 1 or -1; if $i < n$, the attention output is in $[-1/3, 1/3]$. The FFN outputs 1 if the former is true and 0 if the latter is true.

Furthermore, we can compute the quantity $1/i$ (c) using uniform future-masked attention on the output of the above construction, and $1/(n+1-i)$ (d) using past-masking. $\square$

4 Soft Attention, LTL, and Counting

As mentioned above, Barceló et al. (2024) introduced a temporal logic with counting, $\mathsf{LTL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$ (to be defined below), and showed that it can be simulated by AHATs. The languages definable in $\mathsf{LTL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$ therefore form a subclass of those recognized by AHATs that might be more amenable to simulation by SMATs. To show this, we find it useful to factor $\mathsf{LTL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$ into three logics, which account for three different types of hard-attention patterns:

- $\mathsf{TL}[\ominus, \oplus]$, which accounts for immediate predecessor and successor, for example, n-grams (Section 4.1).
- $\mathsf{TL}[\textbf{since}, \textbf{until}]$, which accounts for tie-breaking, for example, flip-flops or induction heads (Section 4.2).
- $\mathsf{TL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$, which accounts for numerical relations, for example, PARITY or MAJORITY (Section 4.4).

In describing the first simulation in Section 4.1, we will make some general remarks that will apply to the other simulations as well.

4.1 Previous and Next

First we show that SMATs can simulate formulas in $\mathsf{TL}[\ominus, \oplus]$, which has just the previous ($\ominus$) and next ($\oplus$) operations.

This logic, like all the logics presented in this section, is a temporal logic, in which each formula ϕ is true or false depending on the position in a string.

Definition 11 ($\mathsf{TL}[\ominus, \oplus]$)**.** Let σ denote symbols of a given alphabet Σ. The syntax of $\mathsf{TL}[\ominus, \oplus]$ is:

$$\phi ::= Q_\sigma \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \ominus\phi \mid \oplus\phi$$

The semantics of $\mathsf{TL}[\ominus, \oplus]$ is given by the relation $\mathbf{w}, i \models \phi$, which means that ϕ is true at position i of $\mathbf{w}$. This relation is defined as follows:

$\mathbf{w}, i \models Q_\sigma$	iff $w_i = \sigma$
$\mathbf{w}, i \models \neg\phi$	iff $\mathbf{w}, i \not\models \phi$
$\mathbf{w}, i \models \phi_1 \wedge \phi_2$	iff $\mathbf{w}, i \models \phi_1$ and $\mathbf{w}, i \models \phi_2$
$\mathbf{w}, i \models \phi_1 \vee \phi_2$	iff $\mathbf{w}, i \models \phi_1$ or $\mathbf{w}, i \models \phi_2$
$\mathbf{w}, i \models \ominus\phi$	iff $\mathbf{w}, (i-1) \models \phi$
$\mathbf{w}, i \models \oplus\phi$	iff $\mathbf{w}, (i+1) \models \phi$

A formula ϕ is *satisfied* by a string $\mathbf{w}$ iff ϕ is true at the last symbol of $\mathbf{w}$, that is, $\mathbf{w}, |\mathbf{w}| \models \phi$.

This logic accounts for the hard-attention patterns used in simulating n-gram models. For instance, the following $\mathsf{TL}[\ominus, \oplus]$ formula detects the 3-gram 101 ending at the current position:

$$\phi_{101} = \ominus \ominus Q_1 \wedge \ominus Q_0 \wedge Q_1$$

Then $10101, 3 \models \phi_{101}$ and $10101, 5 \models \phi_{101}$, but $10101, 4 \not\models \phi_{101}$. Also, the whole string 10101 satisfies ϕ_{101}, that is, $10101 \models \phi_{101}$.

We now show how to simulate $\mathsf{TL}[\ominus, \oplus]$ in a SMAT. There are many conceivable ways to do this. We focus on SMATs that use temperature scaling (and bounded PEs) and those that use unbounded PEs (and no temperature scaling). In both cases, we also consider SMATs that use only future-masking and do not depend on n, suitable for autoregressive language modeling.

Theorem 12. *Any formula ϕ of $\mathsf{TL}[\ominus, \oplus]$ can be simulated by a τ-SMAT with either:*

(a) $\tau(n) = 1/n$ and PEs including i/n and $(-1)^i$

(a) $\tau(n) = 1$ and PEs including i/n, $(-1)^i$, and n.

Proof. Given a formula ϕ, we will construct a transformer that receives as input the string $\mathbf{w}$, and in the final sequence of activation vectors, a designated coordinate gives the sequence of values $\mathbb{I}[\mathbf{w}, i \models \phi]$ for $i \in [|\mathbf{w}|]$.

We construct this transformer by induction on the structure of a formula. Each subformula's truth value is stored in a different component of the activation vectors. For the base case $\phi = Q_\sigma$, we store a 1 in the corresponding component of the word embedding $\mathrm{WE}(\sigma)$, as shown by Yang et al. (2024). For the inductive step, where ϕ is formed out of smaller subformulas ϕ_1 and possibly ϕ_2, we assume that the results of the computation of ϕ_1 and ϕ_2 are available in two components of the activation vector, and we construct a new layer that computes ϕ from ϕ_1 and ϕ_2. The new value is placed in a previously unused coordinate, with all other coordinates zero, and the residual connection supplies all of the previously computed outputs.

Boolean operations can be computed by a FFN as shown by Yang et al. (2024). The remaining cases are the operators $\ominus \phi_1$ and $\oplus \phi_1$. Let $v_1, \ldots, v_n$ be the values of ϕ_1.

In a UHAT, we could compute $\ominus \phi_1$ at positions $i > 1$ by making each position i attend only to position $(i-1)$. If i is odd (and greater than 1), we would make the attention scores greater for even positions, so that rightmost tie-breaking would select position $i - 1$. If i is even, we would make the attention scores greater for odd positions. If $i = 1$, we would output 0. But in a τ-SMAT, we have to approximate this.

Condition (a): We use two future-masked attention layers (or one layer with two heads). In the first attention, the queries and keys are set so that the attention scores are

$$\hat{s}_{ij} = 6\left(\tfrac{1}{2}(-1)^j + \tfrac{j}{2n}\right)$$

which is maximized at the rightmost even position $j \leq i$. Applying Lemma 8b (with $s_{ij} = \frac{1}{2}(-1)^j$), we get

$$\left|\sum_j [\mathrm{rhardmax}\, \mathbf{s} - \mathrm{softmax}\, \hat{\mathbf{s}}]_j v_j\right| \leq 4e^{-3} < \tfrac{1}{4}.$$

Similarly, the second attention attends to the rightmost odd position $j \leq i$, using scores

$$\hat{s}_{ij} = 6\left(\tfrac{1}{2}(-1)^{j+1} + \tfrac{j}{2n}\right).$$

The FFN, at each position i, tests whether i is odd or even; if odd, it uses the output of the first attention, and if even, it uses the output of the second attention. Position $i = 1$ may be set to 0 with a FFN using Proposition 10a.

Similarly, to compute $\oplus \phi_1$, we use two past-masked attention layers with scores as above, but with the tie-breaking terms $+j/(2n)$ changed to $-j/(2n)$, and bound the error using Lemma 8c. Position $i = n$ may be set to 0 with a FFN using Proposition 10b.

Condition (b): Same as condition (a), but scale the query vectors by a factor of n. □

With future-masking, we don't assume that the total length n is known, so we make the temperature depend on the current position i.

Theorem 13. *Any formula ϕ of $\mathsf{TL}[\ominus]$* can be simulated by a future-masked τ-SMAT with:

(a) $\tau(i) = 1/i^2$ and PEs containing $(-1)^i$

(b) $\tau(i) = 1$ and PEs containing $(-1)^i$ and i^2.

Proof. Same as the proof of Theorem 12, but use Lemma 8a. The quantity $1/i$ required by Lemma 8a is supplied by Proposition 10c. □

4.2 Since and Until

Next, we show that SMATs can compute the values of **since** and **until**, which we now define.

Definition 14 (TL[**since**, **until**])**.** Let $\sigma \in \Sigma$ denote symbols of a given alphabet Σ. The temporal logic TL[**since**, **until**] has syntax

$$\phi ::= Q_\sigma \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \\ \mid \phi_1 \textbf{ since } \phi_2 \mid \phi_1 \textbf{ until } \phi_2$$

The semantics of Boolean expressions is as in Definition 11. The intuitive meaning of ϕ_1 **since** ϕ_2 is ''since the last time ϕ_2 was true, ϕ_1 has been true,'' and ϕ_1 **until** ϕ_2 means ''ϕ_1 will be true until the next time ϕ_2 is true.'' More formally:

$\mathbf{w}, i \models \phi_1$ **since** ϕ_2	iff there is $j \le i$ s.t. $\mathbf{w}, j \models \phi_2$ and for all k s.t. $j \le k \le i$ we have $\mathbf{w}, k \models \phi_1$
$\mathbf{w}, i \models \phi_1$ **until** ϕ_2	iff there is $j \ge i$ s.t. $\mathbf{w}, j \models \phi_2$ and for all k s.t. $i \le k \le j$ we have $\mathbf{w}, k \models \phi_1$.

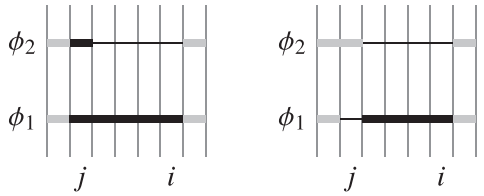
Theorem 15. *Any formula ϕ of* TL[**since**, **until**] *can be simulated by a τ-*SMAT with either:

(a) $\tau(n) = 1/n$ *and PEs including* i/n

(b) $\tau(n) = 1$ *and PEs including* i/n *and* n.

Proof. By induction on the structure of ϕ. The Q_σ and Boolean operation cases are as in Theorem 12.

The case $\phi = \phi_1$ **since** ϕ_2 is intuitively computed as follows. We look at positions $j = i, i-1, i-2$, and so on. As long as ϕ_1 is true and ϕ_2 is false, we keep looking to the left. But we stop when we encounter one of these situations (black means true, blank means false, and gray means either):



$\mathbf{w}, i \models \phi_1$ **since** ϕ_2 $\quad$ $\mathbf{w}, i \not\models \phi_1$ **since** ϕ_2

On the left, j is the rightmost position satisfying ϕ_2, and ϕ_1 is true at positions j to i, so $w, i \models \phi_1$ **since** ϕ_2. On the right, the rightmost position satisfying ϕ_2 is at or to the left of j, but ϕ_1 is false at j, so $w, i \not\models \phi_1$ **since** ϕ_2. If we never encounter either case, there is no j satisfying ϕ_2, so $w, i \not\models \phi_1$ **since** ϕ_2.

Thus, in a UHAT (Yang et al., 2024), we could attend to the rightmost position j maximizing $\mathbb{I}[\mathbf{w}, j \models \neg\phi_1 \vee \phi_2]$ and return the value of $\phi_1 \wedge \phi_2$ at j. But in a τ-SMAT, we have to approximate this.

Under condition (a), add a future-masked self-attention layer with:

$$\hat{s}_{ij} = 6\left(\mathbb{I}[\mathbf{w}, j \models \neg\phi_1 \vee \phi_2] + \tfrac{j}{2n}\right) \\ v_j = \left[\mathbb{I}[\mathbf{w}, j \models \phi_1 \wedge \phi_2]\right].$$

Using temperature $\tau \le 1/n$ satisfies the requirements of Lemma 8b (with $s_{ij} = \mathbb{I}[\mathbf{w}, j \models \neg\phi_1 \vee \phi_2]$), so we have

$$\left|\sum_j [\text{rhardmax } \mathbf{s} - \text{softmax } \hat{\mathbf{s}}]_j v_j\right| \le 4e^{-3} < \tfrac{1}{4}.$$

This yields approximate Boolean values for ϕ_1 **since** ϕ_2, which are rounded to 0 or 1 using a FFN (Lemma 9).

The simulation of ϕ_1 **until** ϕ_2 is symmetric, using the tiebreaker of $-j/(2n)$ and Lemma 8c. Under condition (b), we scale the queries $\mathbf{q}_i$ by a factor of n. □

For decoder-only models (with obligatory future-masking) we also describe a simulation of TL[**since**] that does not depend on n.

Theorem 16. *Any formula ϕ of* TL[**since**] *can be simulated by a future-masked τ-*SMAT with PEs including $1/i$ and either:

(a) $\tau(i) = 1/i^2$ *and PEs including* $1/i$

(b) $\tau(i) = 1$ *and PEs including* $1/i$ *and* i^2.

Proof. To simulate **since**, use a future-masked attention layer with:

$$\hat{s}_{ij} = 3\left(\mathbb{I}[\mathbf{w}, j \models\neq \phi_1 \vee \phi_2] - \tfrac{1}{j}\right) \\ v_j = \left[\mathbb{I}[\mathbf{w}, j \models \phi_1 \wedge \phi_2]\right].$$

Using either a temperature of $1/i^2$ or multiplying $\mathbf{q}_i$ by i^2 satisfies the requirements of Lemma 8b (with $s_{ij} = \mathbb{I}[\mathbf{w}, j \models \neg\phi_1 \vee \phi_2]$). The result then follows after rounding the approximate Boolean values to 0 or 1. □

4.3 LTL

Combining $\mathsf{TL}[\ominus, \oplus]$ and $\mathsf{TL}[\textbf{since}, \textbf{until}]$ we get $\mathsf{TL}[\ominus, \oplus, \textbf{since}, \textbf{until}] = \mathsf{LTL}$. Masked UHATs recognize exactly those formal languages definable by LTL (Yang et al., 2024). It follows from the above that LTL, and therefore any masked UHAT, can be simulated by a τ-SMAT.

Corollary 17. *Any formula ϕ of LTL can be simulated by a τ-SMAT with either:*

(a) $\tau(n) = 1/n$ and PEs including i/n and $(-1)^i$

(b) $\tau(n) = 1$ and PEs including i/n, $(-1)^i$, and n.

Proof. By Theorem 12 and 15. □

We can also simulate the past fragment of LTL (with only **since** and $\ominus$) using only future-masking, suitable for a decoder-only transformer.

Corollary 18. *Any formula of $\mathsf{TL}[\ominus, \textbf{since}]$ can be simulated by a future-masked τ-SMAT with*

(a) $\tau(i) = 1/i^2$ and PEs including $(-1)^i$

(b) $\tau(i) = 1$ and PEs including $(-1)^i$ and i^2.

Proof. By Theorem 13 and 16. The quantity $1/i$ required by Theorem 16 is supplied by Proposition 10c. □

We recall that Hahn (2020, Lemma 5) places a bound $O(1/n)$ on the gap between the output for acceptance or rejection. For a fixed-size gap, some activation in the transformer must be scaled up by a factor of n. The constructions in Theorem 12, Theorem 15, and Corollary 17 use $\tau(n) = 1/n$ or scale queries by n, and are optimal in the sense that we should not expect the factor of n to be asymptotically any smaller.

4.4 Counting

As the third step, we show that τ-SMAT can simulate a logic with counting terms and numerical predicates that can be applied to both positions and counts (Barceló et al., 2024).

Definition 19. A *unary numerical predicate* is a family of functions $\theta = (\theta_n)_{n>0}$ where $\theta_n \colon \{0, 1, \ldots, n\} \to \{0, 1\}$. We write Mon for the set of all unary numerical predicates. The logic $\mathsf{TL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$ has syntax

$$\begin{aligned} \phi &::= Q_\sigma \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \\ &\quad \mid \theta \mid \theta(\kappa) \mid t_1 < t_2 \\ t &::= \kappa \mid t_1 + t_2 \mid t_1 - t_2 \mid 1 \\ \kappa &::= \overleftarrow{\#}[\phi] \mid \overrightarrow{\#}[\phi] \end{aligned}$$

Formulas are interpreted as follows:

$$\begin{aligned} \mathbf{w}, i &\models \theta && \text{iff } \theta_{|w|}(i) \\ \mathbf{w}, i &\models \theta(\kappa) && \text{iff } \theta_{|w|}(\kappa^{\mathbf{w},i}) \\ \mathbf{w}, i &\models t_1 < t_2 && \text{iff } t_1^{(\mathbf{w},i)} < t_2^{(\mathbf{w},i)}. \end{aligned}$$

Terms are interpreted as integers:

$$\begin{aligned} \overrightarrow{\#}[\phi]^{(\mathbf{w},i)} &= |\{j \le i \mid \mathbf{w}, j \models \phi\}| \\ \overleftarrow{\#}[\phi]^{(\mathbf{w},i)} &= |\{j \ge i \mid \mathbf{w}, j \models \phi\}| \\ (t_1 + t_2)^{(\mathbf{w},i)} &= t_1^{(\mathbf{w},i)} + t_2^{(\mathbf{w},i)} \\ (t_1 - t_2)^{(\mathbf{w},i)} &= t_1^{(\mathbf{w},i)} - t_2^{(\mathbf{w},i)} \\ 1^{(\mathbf{w},i)} &= 1. \end{aligned}$$

For example, PARITY is the set of binary strings with an odd number of 1s. Using the numerical predicate ODD, where $\mathsf{ODD}_n(i)$ is true iff i is odd, we can define this using the following $\mathsf{TL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$ formula:

$$\phi_{\mathsf{PARITY}} = \mathsf{ODD}\left(\overleftarrow{\#}[Q_1]\right).$$

When evaluated at the last position, $\overleftarrow{\#}[Q_1]$ is the number of occurrences of 1 in the whole string, and ϕ_{PARITY} tests whether this number is odd.

We show how SMATs can compute formulas of $\mathsf{TL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$.

Theorem 20. *Any formula ϕ of $\mathsf{TL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$ can be simulated by a τ-SMAT with either:*

(a) $\tau(n) = 1/n$ and PEs including $1/i$, $1/(n - i + 1)$, i, i^2, and all numerical predicates in ϕ

(b) $\tau(n) = 1$ and PEs including $1/i$, $1/(n - i + 1)$, i, in, $i^2 n$, and all numerical predicates in ϕ.

Proof. We show condition (a) in detail and remark briefly at the end how to modify the proof for condition (b).

The proof is again by induction on the structure of ϕ. The cases we need to consider are:

- $\phi = Q_\sigma$ or ϕ is a Boolean operation: As in Theorem 12.
- $\phi = \theta$: We store $\theta_n(i)$ in the corresponding component of the position embedding $\mathrm{PE}_n(i)$.
- $\phi = (\overleftarrow{\#}[\phi_1] > 0)$ or $(\overrightarrow{\#}[\phi_1] > 0)$: This case is proved below, both as a warm-up for the other cases as well as a subroutine of the other cases.
- $\phi = (t_1 < t_2)$: Proved below.
- $\phi = \theta(\overleftarrow{\#}[\phi_1])$ or $\theta(\overrightarrow{\#}[\phi_1])$: Proved below.

Case $\phi = (\overleftarrow{\#}[\phi_1] > 0)$: We can use an FFN to test whether ϕ_1 is true at the current position i, and if so, we know that $\overleftarrow{\#}[\phi_1] > 0$, so we return 1. Otherwise, we know that $\overleftarrow{\#}[\phi_1] < i$, as we assume below.

The essential idea is simply to use uniform attention to perform the counting. However, attention does not count; it averages. So the challenge is to turn averages into counts. The construction proceeds in three steps. First, we compute $\overleftarrow{\#}[\phi_1]/i$ exactly. Second, we scale up to $\overleftarrow{\#}[\phi_1]$, but only approximately. Third, we test whether the count is greater than 0.

Step 1: Average. The first step of computing $\overleftarrow{\#}[\phi_1]/i$ can be done using masked uniform attention (Yang and Chiang, 2024).

Step 2: Scale. For the second step, we approximately retrieve $\big(\overleftarrow{\#}[\phi_1] + 1\big)$ using a table-lookup attention layer with

$$\mathbf{q}_i = 3 \begin{bmatrix} \frac{\overleftarrow{\#}[Q_1]}{i} + \frac{1}{i} \\ \frac{1}{i} \end{bmatrix} \quad \mathbf{k}_j = \begin{bmatrix} 2j \\ -j^2 \end{bmatrix} \quad v_j = \begin{bmatrix} j \end{bmatrix}.$$

Then $s_{ij} = 3\big(2\big(\overleftarrow{\#}[\phi_1] + 1\big)j - j^2\big)/i$, which is uniquely maximized at $j = \overleftarrow{\#}[\phi_1] + 1$. We apply Lemma 7 with temperature $\tau = 1/n$ and gap $\gamma = 3/(i\tau)$ to obtain

$$c_i = \sum_j [\mathrm{softmax}_\tau(s_{i1}, \ldots, s_{ii})]_j v_j$$

$$\left| c_i - \big(\overleftarrow{\#}[\phi_1] + 1\big) \right| \le 4e^{-3n/i} \le 4e^{-3} < \frac{1}{4}.$$

Thus, the output value c_i is within $\overleftarrow{\#}[\phi_1] + 1 \pm 1/4$.

Step 3: Compare. The third step is to apply the FFN of Lemma 9 to $(c_i - 1)$, which will output 0 if $\overleftarrow{\#}[\phi_1] = 0$ and 1 if $\overleftarrow{\#}[\phi_1] > 0$.

Case $\phi = (t_1 < t_2)$: If ϕ is a comparison of count terms, it can be rewritten in the following form. Let $\overleftarrow{K}$ and $\overrightarrow{K}$ be disjoint sets of indices, and let C and λ_k (for $k \in \overleftarrow{K} \cup \overrightarrow{K}$) be integers:

$$\phi = \left(\sum_{k \in \overleftarrow{K}} \lambda_k \overleftarrow{\#}[\phi_k] + \sum_{k \in \overrightarrow{K}} \lambda_k \overrightarrow{\#}[\phi_k] > C \right).$$

Let $\Lambda = \sum_{k \in \overleftarrow{K} \cup \overrightarrow{K}} |\lambda_k|$. If $\Lambda = 0$, the formula is constant, so we assume without loss of generality that $\Lambda \ge 1$.

For each $k \in \overleftarrow{K}$, we need to approximate $\overleftarrow{\#}[\phi_k]$. (Terms $\overrightarrow{\#}[\phi_k]$ for $k \in \overrightarrow{K}$ are similar and described briefly afterwards.) We can use the construction in the previous case to test whether $\overleftarrow{\#}[\phi_k] = 0$; if not, then we approximate $\overleftarrow{\#}[\phi_k]$ as follows, assuming that $\overleftarrow{\#}[\phi_k] > 0$.

Step 1: Average. The first step of computing $\overleftarrow{\#}[\phi_k]/i$ for each k is the same as in the previous case.

Step 2: Scale. For the second step, we approximately retrieve $\overleftarrow{\#}[\phi_k]$ using a table-lookup attention layer with

$$\mathbf{q}_i = 3\Lambda \begin{bmatrix} \frac{\overleftarrow{\#}[\phi_k]}{i} \\ \frac{1}{i} \end{bmatrix} \quad \mathbf{k}_j = \begin{bmatrix} 2j \\ -j^2 \end{bmatrix} \quad v_j = \begin{bmatrix} j \end{bmatrix}.$$

Then $s_{ij} = 3\Lambda(2\overleftarrow{\#}[\phi_k]j - j^2)/i$, which is uniquely maximized at $j = \overleftarrow{\#}[\phi_k]$. We apply Lemma 7 with temperature $\tau = 1/n$ and gap $\gamma = 3\Lambda/(i\tau)$ to obtain an approximate count

$$c_k(i) = \sum_j [\underset{\tau}{\mathrm{softmax}}(s_{i1}, \ldots, s_{ii})]_j v_j$$

that is within $\overleftarrow{\#}[\phi_k] \pm 1/(4\Lambda)$, which is close enough to $\overleftarrow{\#}[\phi_k]$ for the comparison step.

Computing a right-counting term $\overrightarrow{\#}[\phi_k]$ requires only a slight modification of the second step. The first step gives $\overrightarrow{\#}[\phi_k]/(n-i+1)$. The second step requires the quantity $1/(n-i+1)$. Since $n - i + 1 \leq n$, the bound of $1/(4\Lambda)$ is derived in the same way.

Step 3: Compare. The third step is to compute the linear combination using a FFN:

$$H(i) = \sum_{k \in \overleftarrow{K} \cup \overrightarrow{K}} \lambda_k c_k(i).$$

The error in $H(i)$ is

$$\left| H(i) - \left(\sum_{k \in \overleftarrow{K}} \lambda_k \overleftarrow{\#}[\phi_k] + \sum_{k \in \overrightarrow{K}} \lambda_k \overrightarrow{\#}[\phi_k] \right) \right| \leq \left| \sum_{k \in \overleftarrow{K} \cup \overrightarrow{K}} \lambda_k \frac{1}{4\Lambda} \right| \leq \frac{1}{4}.$$

If $H(i) - C \geq 3/4$ then ϕ is true; if $H(i) - C \leq 1/4$ then ϕ is false. So we can compute the correct Boolean value using Lemma 9.

Case $\phi = \theta(\overrightarrow{\#}[\phi_1])$: If ϕ is a numerical predicate θ applied to a count, we test whether $\overrightarrow{\#}[\phi_1] = 0$ as above; if so, return $\theta(0)$. Otherwise, we perform the first step as above. We perform the second step using scores $s_j = 3(2\overrightarrow{\#}[\phi_1]j - j^2)/i$, temperature $\tau = 1/n$, and values $v_j = \theta_n(j)$. Since $|\theta_n(j)| \leq 1$, we can use Lemma 6 with $\gamma = 3/(i\tau)$ to ensure that the attention output approximates $\theta_n(\overrightarrow{\#}[\phi_1])$ with error at most $4e^{-\gamma} \leq 1/4$, which we can correct using a FFN (Lemma 9).

Case $\phi = \theta(\overrightarrow{\#}[\phi])$: This is just the mirror image of the previous case.

This completes the proof for condition (a). The proof for condition (b) is identical, except that in the second step, we scale $\mathbf{k}_j$ by n. □

Theorem 21. *Any formula ϕ of $\mathsf{TL}[\overleftarrow{\#}, +, \mathsf{Mon}]$ can be simulated by a future-masked τ-SMAT with $\tau(i) = 1/i$ and PEs including $1/i$, i, i^2, and all numerical predicates in ϕ.*

Proof. Same as Theorem 20a. □

4.5 $\mathsf{LTL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$

The logic $\mathsf{LTL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$ includes all previously discussed temporal operators ($\ominus, \oplus$, **since**, **until**), counting terms $\overleftarrow{\#}, \overrightarrow{\#}$, linear inequalities of counting terms, and all unary numerical predicates. It is exactly the logic $\mathsf{LTL}(\mathbf{C}, +)$ introduced by Barceló et al. (2024).

For example, the following formula of $\mathsf{LTL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$ recognizes PARITY without using numerical predicates:

$$\phi_{\mathsf{PARITY}} = \left(\overleftarrow{\#} \left[\overleftarrow{\#}[\ominus Q_1] = \overrightarrow{\#}[Q_1] \right] = 0 \right).$$

The total number of 1's is odd iff there is no position i where the number of 1's strictly to the left of i is equal to the number of 1's at or to the right of i. Above, the subformula $\ominus Q_1$ tests whether there is a 1 immediately to the left, so $\overleftarrow{\#}[\ominus Q_1]$ counts the number of 1's strictly to the left.

Barceló et al. (2024) simulate this logic using AHATs, but here we give conditions under which this logic can be simulated using τ-SMATs.

Corollary 22. *Any formula ϕ of $\mathsf{LTL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$ can be simulated by a τ-SMAT with PEs including all numerical predicates in ϕ and either:*

(a) $\tau(n) = 1/n$ and PEs including $(-1)^i$, i/n, i, and i^2

(b) $\tau(n) = 1$ and PEs including $(-1)^i$, i/n, i, n, in, and $i^2 n$.

Proof. By Corollary 17 and Theorem 20. The quantities $1/i$ and $1/(n-i+1)$ required by Theorem 20 are supplied by Proposition 10cd. □

Corollary 23. *Any formula ϕ of $\mathsf{TL}[\ominus, \textbf{since}, \overleftarrow{\#}, +, \mathsf{Mon}]$ can be simulated by a future-masked τ-SMAT with $\tau(i) = 1/i^2$ and PEs including $(-1)^i$, i, i^2, and all numerical predicates in ϕ.*

Proof. By Corollary 18 and Theorem 21. The quantity $1/i$ required by Theorem 21 is supplied by Proposition 10c. □

5 Average-Hard Attention

The simulations of temporal logics in the preceding section compute exact Boolean values for every subformula, because a FFN can round an approximate Boolean to an exact Boolean. Counting terms are approximated sufficiently well that comparisons between them, and applications of numerical predicates to them, produce approximate Booleans. But for the case of simulating an AHAT, there is no similar option for eliminating approximation error, because in general the values in the computation are not confined to a finite set.

There is a simple argument that if we take an AHAT T and replace its ahardmax weighting function with softmax$_\tau$ to produce a τ-SMAT $\hat{T}$, then for any input sequence, as τ goes to zero, the limit of the sequence of activation vectors of the final layer of $\hat{T}$ equals the sequence of activation vectors of the final layer of T. This is because a transformer is the composition of continuous functions, and the softmax$_\tau$ function approaches the ahardmax function as τ goes to zero. This argument is likely the basis of many researchers' confidence that SMATs can approximate AHATs; however, it does not by itself bound the rate of convergence.

Here, we bound the rate of convergence for AHATs in terms of the gap separating the maximal attention score or scores from other scores. We say that an attention layer has *gap* $\gamma(n)$ iff for every input with length n and every position i, the attention scores at i (that is, $s_{i,*}$) have gap $\gamma(n)$ (Definition 5). An AHAT has gap $\gamma(n)$ if all of its attention layers have gap $\gamma(n)$. Existing uses of average-hard attention in the literature generally have gap $\Omega(1/n^k)$ for small constants k (see Section 6).

Proposition 24. *Every* AHAT *has a strictly positive gap function.*

Proof. For any n, because there are only finitely many possible input strings of length n, we may take $\gamma(n)$ to be the minimum absolute difference, over all input strings, attention layers, and positions i, between the maximal attention score $s_{\max} = \max_j s_{ij}$ and any other attention score $s_{ij'} < s_{\max}$. □

Let T be an AHAT with gap $\gamma(n)$. We transform T into a τ-SMAT $\hat{T}$ that has the same parameters as T but uses temperature-scaled softmax attention in place of average-hard attention. By choosing a temperature function $\tau(n)$ depending on $\gamma(n)$, we show that for every input, the output of $\hat{T}$ closely approximates the output of T.

We define the function $x_{\max}(n)$ to be the maximum of 1 and the absolute value of any entry in $\mathbf{x}_i$ over all inputs of length n and positions i, where $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is the initial activation sequence for the input. This is well defined because there are finitely many different inputs of length n (similar to the proof of Proposition 24).

First, we prove the following lemma bounding the error introduced by a single transformer layer.

Lemma 25. *Let T be an* AHAT *with gap $\gamma(n)$, and let $\hat{T}$ be the corresponding τ-*SMAT *with temperature $\tau(n) \leq 1$. There exists a constant $K \geq 1$ such that for all positive integers n and all sufficiently small $\epsilon > 0$, we have the following, for each layer ℓ. For any $(\mathbf{g}_1, \ldots, \mathbf{g}_n)$ and $(\hat{\mathbf{g}}_1, \ldots, \hat{\mathbf{g}}_n)$ such that $\|\mathbf{g}_i - \hat{\mathbf{g}}_i\|_1 \leq \epsilon$ for all i, let $(\mathbf{h}_1, \ldots, \mathbf{h}_n)$ be the output of layer ℓ of T on input $(\mathbf{g}_1, \ldots, \mathbf{g}_n)$, and let $(\hat{\mathbf{h}}_1, \ldots, \hat{\mathbf{h}}_n)$ be the output of layer ℓ of $\hat{T}$ on input $(\hat{\mathbf{g}}_1, \ldots, \hat{\mathbf{g}}_n)$. Then, for all i,*

$$\left\|\mathbf{h}_i - \hat{\mathbf{h}}_i\right\|_1 \leq K x_{\max}(n) \left(n e^{-\frac{\gamma(n)}{\tau(n)}} + \frac{x_{\max}(n)}{\tau(n)} \epsilon \right).$$

Proof. See Appendix C.3. □

The first term inside the parentheses bounds the error due to approximating ahardmax by softmax, while the second term bounds the error propagated from the layer input to the layer output.

By iterating this lemma over the layers of an AHAT, we prove the following theorem.y

Theorem 26. *Let T be an* AHAT *with gap $\gamma(n)$. For any function $\epsilon(n) \in \Omega(1/\mathsf{poly}(n))$, there exists a temperature function $\tau(n)$ with*

$$\frac{1}{\tau(n)} \in O\left(\frac{1}{\gamma(n)} \log \frac{n x_{\max}(n)}{\gamma(n)} \right)$$

*such that the τ-*SMAT *$\hat{T}$ corresponding to T approximates T in the following sense. For every input, if $(\mathbf{y}_1, \ldots, \mathbf{y}_n)$ is the output of T and $(\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_n)$ is the output of $\hat{T}$, then*

$$\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_1 \leq \epsilon(n) \qquad \text{for all } i \in [n].$$

In particular, if $\gamma(n) = 1/n^k$ and $x_{\max}(n) \in n^{O(1)}$, we have $1/\tau(n) \in O(n^k \log n)$. Note that the constant factor in the bound hides quantities not depending on n, including an exponential dependence on the depth L.

Proof. See Appendix C.4. □

6 Some Applications

Below, we consider some discrete tasks that have been solved using UHATs or AHATs.

Induction Heads. In the *basic induction head task* (Elhage et al., 2021), the input is a string $w_1 \cdots w_n$, and the goal is to find the rightmost $i < n$ such that $w_i = w_n$ and output w_{i+1}. Sanford et al. (2024) consider the more general *k-hop induction head task*, which iterates the basic induction head task a constant k number of times.

The basic induction head task can be defined exactly in LTL (Yang et al., 2024), and the k-hop induction head task can be computed by iterating the basic induction head task k times, so Corollary 17 gives a τ-SMAT for the k-hop induction head task with $\tau(n) = 1/n$. Previous solutions for these tasks include the following.

Liu et al. (2023a) show that the *flip-flop* language, a particular instance of the basic induction head task, can be recognized exactly by an AHAT with gap $\gamma(n) = 1/n$ and $x_{\max}(n) \in O(1)$. They also give a SMAT in which queries are scaled by $O(n \log n)$, which is equivalent to a temperature of $\Omega(1/(n \log n))$, a little lower than our $\tau(n) = 1/n$ solution.

Sanford et al. (2024) give an AHAT for the k-hop induction head task with gap $\gamma(n) \in \Omega(1/n^2)$ and show how to simulate it with soft attention. As discussed in Section 1, this simulation is general but not parameter-uniform.

S-RASP Simulation. Strobl et al. (2025) consider transductions (functions from finite strings to finite strings) and introduce the language S-RASP as a concise way to define them. They show that any transduction expressible in S-RASP can be computed by an AHAT with gap $\gamma(n) \in \Omega(1/n^3)$ and $x_{\max}(n) \in O(1)$. Thus, by Theorem 26, every transduction that is expressible in S-RASP can be exactly computed by a τ-SMAT with $\tau(n) \in \Omega(1/(n^3 \log n))$.

DYCK-k Membership. The language DYCK-k consists of balanced sequences of brackets of k types. Yao et al. (2021) give a transformer for DYCK-k using selective layer norm and both soft attention and average-hard attention. Their algorithm can be expressed in S-RASP (Appendix D), so Theorem 26 gives a τ-SMAT with $\tau(n) \in \Omega(1/(n^3 \log n))$. However, we conjecture that DYCK-2 cannot be defined in $\mathsf{LTL}[\overleftarrow{\#}, \overrightarrow{\#}, +, \mathsf{Mon}]$.

7 Conclusion

Under what conditions can soft attention simulate hard attention? The answer to this question turns out to be rather intricate. As argued in the introduction, any such simulation must give up one of the assumptions of Hahn (2020) by adding temperature scaling, rounding, non-Lipschitz layer normalization, or unbounded position embeddings; or give up a fixed-size gap between acceptance and rejection. We have demonstrated how to use inverse-temperature scaling and unbounded position embeddings to simulate various subclasses of average-hard attention transformers, and inverse-temperature scaling to simulate general average-hard attention transformers. Future research could explore the other interventions, as well as how all these interventions interact in practice with training dynamics.

Acknowledgments

We thank Clayton Sanford for his generous assistance in understanding certain aspects of his paper (Sanford et al., 2024); Bingbin Liu and her co-authors for their beautiful paper (Liu et al., 2023b), which helped spark the current work; and Anej Svete for discussion about simulation of n-gram models (Svete and Cotterell, 2024). Finally, we thank the action editor for their time and anonymous reviewers for their helpful comments, one of which provoked a substantial improvement of Theorem 26. This material is based upon work supported by the National Science Foundation under Grant Nos. 2502292 and 2236418.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. In *Proceedings of the NIPS 2016 Deep Learning Symposium*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations (ICLR)*.

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

David Chiang. 2025. Transformers in uniform TC^0. *Transactions on Machine Learning Research*.

David Chiang and Peter Cholak. 2022. Overcoming a theoretical limitation of self-attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7654–7664. https://doi.org/10.18653/v1/2022.acl-long.527

Benjamin L. Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. 2022. Inductive biases and variable creation in self-attention mechanisms. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5793–5831.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. Transformer Circuits Thread.

Dan Friedman, Alexander Wettig, and Danqi Chen. 2023. Learning Transformer programs. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.

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

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024. Chain of thought empowers transformers to solve inherently serial problems. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023a. Exposing attention glitches with flip-flop language modeling. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, pages 25549–25583.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023b. Transformers learn shortcuts to automata. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*.

Colin McDonald and David Chiang. 2021. Syntax-based attention masking for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 47–52, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-srw.7

William Merrill, Jackson Petty, and Ashish Sabharwal. 2024. The illusion of state in state-space models. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Ken M. Nakanishi. 2025. Scalable-softmax is superior for attention. arXiv:2501.19399,

Jorge Pérez, Javier Marinkovic, and Pablo Barceló. 2019. On the Turing completeness of modern neural network architectures. In *Proceedings of the Seventh International Conference on Learning Representations (ICLR)*.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. 2024. Transformers, parallel computation, and logarithmic depth. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. DiSAN: Directional self-attention network for RNN/CNN-free language understanding. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v32i1.11941

Kenneth J. Sible and David Chiang. 2024. Improving rare word translation with dictionaries and attention masking. In *Proceedings of the 16th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 225–235.

Lena Strobl, Dana Angluin, David Chiang, Jonathan Rawski, and Ashish Sabharwal. 2025. Transformers as transducers. *Transactions of the Association for Computational Linguistics*, 13:200–219. https://doi.org/10.1162/tacl_a_00736

Anej Svete and Ryan Cotterell. 2024. Transformers can represent n-gram language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, pages 6841–6874. https://doi.org/10.18653/v1/2024.naacl-long.381

Matus Telgarsky. 2015. Representation benefits of deep feedforward networks. arXiv:1509.08101.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*.

Andy Yang and David Chiang. 2024. Counting like transformers: Compiling temporal counting logic into softmax transformers. In *Proceedings of the Conference on Language Modeling (CoLM)*.

Andy Yang, David Chiang, and Dana Angluin. 2024. Masked hard-attention transformers recognize exactly the star-free languages. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. 2021. Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 3770–3785. https://doi.org/10.18653/v1/2021.acl-long.292

A MLP Rounding

Lemma 27. *Let p and p' be positive integers with $p' > p$. Any ReLU FFN to round p'-bit numbers to p-bit numbers with ℓ layers and width m satisfies $(2m)^\ell \geq 2^p$.*

Proof. Telgarsky (2015, Lemma 2.1) proves that if a ReLU FFN of depth ℓ and width m computes a function $f : \mathbb{R} \to \mathbb{R}$ then f is continuous piecewise affine with at most $(2m)^\ell$ pieces. So we need to show that any piecewise affine function rounding a p'-bit number to a p-bit number requires at least 2^p pieces.

Let r be the function that rounds a p'-bit number to a p-bit number. The domain of r is the set of $2^{p'}$ real numbers exactly representable with p' bits, and the range of r is the set of 2^p real numbers exactly representable with p bits. (We assume that each p-bit number represents a different real value.) For each y in the range of r, let $m(y)$ be the least x in the domain of r such that $r(x) = y$.

Suppose $f : \mathbb{R} \to \mathbb{R}$ is a piecewise affine function that agrees with r on its domain. If $y_1 < y_2$ are two consecutive elements of the range of r, then $x_1 = m(y_1)$ and $x_2 = m(y_2)$ must be in different pieces of f. This is because there is at least one point x in the domain of r with $x_1 < x < x_2$, and if x_1 and x_2 were in the same piece, then the value of $f(x)$ would fall between y_1 and y_2, which are consecutive elements of the range of r, a contradiction. Thus f must have at least 2^p pieces, and the claim follows. □

Because Sanford et al. (2024) consider precision $p = \Theta(\log n)$ and $p' > p$, this implies that a constant-depth ReLU MLP to round p'-bit numbers to p-bit numbers must have width at least n^ϵ for some $\epsilon > 0$.

B Approximation Bounds

B.1 Proof of Lemma 6

Let

$$a = e^{s_{j^*}} \qquad b = \sum_{j \neq j^*} e^{s_j} \leq \frac{2e^{-\gamma}}{1 - e^{-\gamma}} a.$$

Then

$$\begin{aligned} &\|\text{hardmax } \mathbf{s} - \text{softmax } \mathbf{s}\|_1 \\ &\quad = \left(1 - \frac{a}{a+b}\right) + \frac{b}{a+b} = 2\left(1 - \frac{a}{a+b}\right) \end{aligned}$$

$$\leq 2\left(1 - \frac{a}{a + \frac{2e^{-\gamma}}{1-e^{-\gamma}}a}\right) = 2\left(1 - \frac{1}{1 + \frac{2e^{-\gamma}}{1-e^{-\gamma}}}\right)$$
$$= \frac{4e^{-\gamma}}{1+e^{-\gamma}} \leq 4e^{-\gamma}.$$

This allows us to bound the difference between hard and soft attention:

$$\left|\sum_j [\text{hardmax } \mathbf{s}]_j \, v_j - \sum_j [\text{softmax } \mathbf{s}]_j \, v_j\right|$$
$$\leq \|\text{hardmax } \mathbf{s} - \text{softmax } \mathbf{s}\|_1 \, v_{\max}$$
$$\leq 4e^{-\gamma} v_{\max}.$$

B.2 Table Lookup (Proof of Lemma 7)

Observe that $\sum_j [\text{hardmax } \mathbf{s}]_j j = t$, and let $\alpha = \text{softmax } \mathbf{s}$, so we are trying to bound the difference $|t - \sum_j \alpha_j j|$. Subtracting the constant term γt^2 from each s_j does not affect the α_j, so we have:

$$\alpha_j = \frac{e^{-\gamma(j-t)^2}}{Z} \qquad Z = \sum_{k=1}^{n} e^{-\gamma(k-t)^2} \geq 1.$$

Then

$$\left|t - \sum_{j=1}^{n} \alpha_j j\right|$$
$$= \left|\sum_{j=1}^{n} \alpha_j (t-j)\right|$$
$$= \left|\sum_{j=1}^{t-1} \alpha_j (j-t) + \sum_{j=t+1}^{n} \alpha_j (j-t)\right|$$
$$\leq \sum_{j=t+1}^{n} \alpha_j (j-t) = \frac{1}{Z} \sum_{j=t+1}^{n} e^{-\gamma(j-t)^2}(j-t)$$
$$\leq \sum_{j=t+1}^{n} e^{-\gamma(j-t)^2}(j-t) = \sum_{k=1}^{n-t} e^{-\gamma k^2} k$$
$$\leq \sum_{k=1}^{\infty} e^{-\gamma k^2} k = e^{-\gamma} + \underbrace{\sum_{k=2}^{\infty} e^{-\gamma k^2} k}_{(*)}.$$

Consider the function $f(x) = e^{-\gamma x^2} x$, which is non-negative for $x \geq 0$ and decreasing for $x \geq 1$. So the term $(*)$ is a lower Riemann sum of f:

$$\sum_{k=2}^{\infty} e^{-\gamma k^2} k \leq \int_1^{\infty} e^{-\gamma x^2} x \, dx = \frac{1}{2\gamma} e^{-\gamma} \leq \frac{1}{2} e^{-\gamma}.$$

B.3 Tie-Breaking (Proof of Theorem 8)

(a) Let j^* be the rightmost position $j^* \leq n$ maximizing s_{j^*}. Consider any other position j.

If $s_j = s_{j^*}$, then $j \leq j^*$, and

$$\hat{s}_{j^*} - \hat{s}_j = \gamma n^2 \left(-\frac{1}{j^*} + \frac{1}{j}\right)$$
$$= \gamma n^2 \frac{j^* - j}{j^* j}$$
$$\geq \gamma |j^* - j|.$$

If $s_j \leq s_{j^*} - 1$, then

$$\hat{s}_{j^*} - \hat{s}_j \geq \gamma n^2 \left(1 - \frac{1}{j^*} + \frac{1}{j}\right)$$
$$\geq \gamma n^2 \left(1 - \frac{1}{1} + \frac{1}{n}\right)$$
$$= \gamma n \geq \gamma |j^* - j|.$$

So we can use Lemma 6 with a gap of γ:

$$\left|\sum_j [\text{rhardmax } \mathbf{s} - \text{softmax } \hat{\mathbf{s}}]_j v_j\right| \leq 4e^{-\gamma} v_{\max}.$$

(b) Let j^* be the rightmost position $j^* \leq n$ maximizing s_{j^*}. Consider any other position j.

If $s_j = s_{j^*}$, then $j \leq j^*$, and

$$\hat{s}_{j^*} - \hat{s}_j = 2\gamma n \left(\frac{j^*}{2n} - \frac{j}{2n}\right)$$
$$= \gamma |j^* - j|.$$

If $s_j \leq s_{j^*} - 1$, then

$$\hat{s}_{j^*} - \hat{s}_j \geq 2\gamma n \left(1 + \frac{j^*}{2n} - \frac{j}{2n}\right)$$
$$\geq 2\gamma n \left(1 + \frac{1}{2n} - \frac{n}{2n}\right)$$
$$= \gamma(1+n) \geq \gamma |j^* - j|.$$

So we can use Lemma 6:

$$\left|\sum_j [\text{rhardmax } \mathbf{s} - \text{softmax } \hat{\mathbf{s}}]_j v_j\right| \leq 4e^{-\gamma} v_{\max}.$$

(c) The leftmost case is symmetric.

C Proofs for Average-Hard Attention

To reduce complication, we assume that the transformers we consider have $d = d_{\text{k}} = d_{\text{f}}$. We

begin with a lemma bounding the effect of an affine transformation on the norm of a vector.

Lemma 28. *Let* $\mathbf{W} \in \mathbb{R}^{d \times d}$ *and* $\mathbf{b}, \mathbf{x} \in \mathbb{R}^d$. *Then* $\|\mathbf{W}\mathbf{x} + \mathbf{b}\|_1 \leq d w_{\max} \|\mathbf{x}\|_1 + d b_{\max}$, *where* $w_{\max}$ *and* $b_{\max}$ *are the maximum absolute values of any entry in* $\mathbf{W}$ *and* $\mathbf{b}$, *respectively*.

Proof.

$$\begin{aligned}\|\mathbf{W}\mathbf{x} + \mathbf{b}\|_1 &\leq \sum_{k=1}^{d} |\mathbf{W}_{k,*} \cdot \mathbf{x}| + \|\mathbf{b}\|_1 \\ &\leq \sum_{k=1}^{d} w_{\max} \|\mathbf{x}\|_1 + d b_{\max} \\ &= d w_{\max} \|\mathbf{x}\|_1 + d b_{\max}. \qquad \square\end{aligned}$$

For the rest of this section, as defined in Section 5, T is an AHAT with L layers and gap function $\gamma(n)$, and $\hat{T}$ is the τ-SMAT with the same parameters. Let $p_{\max}$ be the maximum of 1 and the absolute value of any parameter occurring in T. Over all inputs of length n, let $x_{\max}(n)$ be the maximum of 1 and the absolute value of any entry in any of the $\mathbf{x}_i$.

First, we bound the activations of T.

Lemma 29. *Let* $K = 2(d^2 p_{\max}^2 + 1)(d p_{\max} + 1)$. *For any* $\ell \in \{0, \ldots, L\}$, *let* $(\mathbf{h}_1, \ldots, \mathbf{h}_n)$ *be the sequence of activations after layer* ℓ *for some input of length* n. *Then for all* $i \in [n]$, $\|\mathbf{h}_i\|_1 \leq K^\ell d x_{\max}(n) \leq K^L d x_{\max}(n)$.

Proof. By induction on ℓ. Fix an input of length n. For $\ell = 0$, we have $\mathbf{h}_i = \mathbf{x}_i$ for the initial activation vector $\mathbf{x}_i$, and $\|\mathbf{x}_i\|_1 \leq d x_{\max}(n)$.

For $\ell > 0$, let $(\mathbf{g}_1, \ldots, \mathbf{g}_n)$ be the sequence of inputs to layer ℓ. By the induction hypothesis, for all $i \in [n]$, $\|\mathbf{g}_i\|_1 \leq K^{\ell-1} d x_{\max}(n)$. For $i \in [n]$, the attention layer computes

$$\mathbf{c}_i = \mathbf{g}_i + \sum_{j=1}^{n} \alpha_{ij} \mathbf{W}^{(\mathrm{V})} \mathbf{g}_j.$$

By Lemma 28, $\left\|\mathbf{W}^{(\mathrm{V})} \mathbf{g}_j\right\| \leq d p_{\max} \left\|\mathbf{g}_j\right\|$ and thus $\|\mathbf{c}_i\|_1 \leq (d p_{\max} + 1) K^{\ell-1} d x_{\max}(n)$.

Let $\mathbf{W}_1$, $\mathbf{b}_1$, $\mathbf{W}_2$, $\mathbf{b}_2$ be the parameters of the feedforward layer. We have

$$\mathbf{h}_i = \mathbf{c}_i + \mathbf{W}_2 \operatorname{ReLU}(\mathbf{W}_1 \mathbf{c}_i + \mathbf{b}_1) + \mathbf{b}_2.$$

Because the ReLU operation does not increase the norm of a vector, it suffices to bound, using Lemma 28,

$$\begin{aligned}\|\mathbf{h}_i\|_1 &\leq \|\mathbf{c}_i + \mathbf{W}_2(\mathbf{W}_1 \mathbf{c}_i + \mathbf{b}_1) + \mathbf{b}_2\|_1 \\ &\leq (d^2 p_{\max}^2 + 1) \|\mathbf{c}_i\|_1 + d^2 p_{\max}^2 + d p_{\max}.\end{aligned}$$

Substituting in the bound for $\|\mathbf{c}_i\|_1$ and observing that $(d^2 p_{\max}^2 + 1)(d p_{\max} + 1) \geq d^2 p_{\max}^2 + d p_{\max}$ and $x_{\max}(n) \geq 1$, we conclude that

$$\begin{aligned}\|\mathbf{h}_i\|_1 &\leq 2(d^2 p_{\max}^2 + 1)(d p_{\max} + 1) K^{\ell-1} d x_{\max}(n) \\ &= K^\ell d x_{\max}(n). \qquad \square\end{aligned}$$

C.1 FFNN Layers

A FFNN layer computes the same function in T and $\hat{T}$. It may amplify error in the input values, but does not introduce new error.

Lemma 30. *If for a FFNN layer of* T *we have two sequences of input vectors* $(\mathbf{c}_1, \ldots, \mathbf{c}_n)$ *and* $(\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_n)$ *such that for all* i, $\|\mathbf{c}_i - \hat{\mathbf{c}}_i\|_1 \leq \epsilon$, *then for the sequences of output vectors* $(\mathbf{h}_1, \ldots, \mathbf{h}_n)$ *of this layer of* T *and* $(\hat{\mathbf{h}}_1, \ldots, \hat{\mathbf{h}}_n)$ *of the same layer of* $\hat{T}$, *we have for all* i, $\left\|\mathbf{h}_i - \hat{\mathbf{h}}_i\right\|_1 \leq (d^2 p_{\max}^2 + 1)\epsilon$.

Proof. Let $\mathbf{W}_1$, $\mathbf{b}_1$, $\mathbf{W}_2$, $\mathbf{b}_2$ be the parameters of the FFNN. We have $|\mathrm{ReLU}(x) - \mathrm{ReLU}(y)| \leq |x - y|$ for all real numbers x, y, so the ReLU operation does not increase error. Thus it suffices to bound the increase in error from the two affine transformations and the residual connection.

For $i \in [n]$, $\|(\mathbf{W}_1 \mathbf{c}_i + \mathbf{b}_1) - (\mathbf{W}_1 \hat{\mathbf{c}}_i + \mathbf{b}_1)\|_1 = \|\mathbf{W}_1(\mathbf{c}_i - \hat{\mathbf{c}}_i)\|_1 \leq d p_{\max} \epsilon$, by Lemma 28.

The second affine transformation with parameters $\mathbf{W}_2$ and $\mathbf{b}_2$ at worst multiplies the error by $d p_{\max}$ again, and the residual connection adds at most the input error, so the final error is bounded by $(d^2 p_{\max}^2 + 1)\epsilon$. $\square$

C.2 Self-attention layers

A self-attention layer may not only amplify error in the input values, but may also introduce new error because of the difference between average-hard attention and temperature scaled softmax attention. Consider a self-attention layer of T with parameter matrices $\mathbf{W}^{(\mathrm{Q})}$, $\mathbf{W}^{(\mathrm{K})}$, and $\mathbf{W}^{(\mathrm{V})}$ in $\mathbb{R}^{d \times d}$. We first bound the error in computing attention scores $s_{i,j}$.

Lemma 31. *Let $(\mathbf{h}_1, \ldots, \mathbf{h}_n)$ be a sequence of input vectors to this attention layer for T and $(\hat{\mathbf{h}}_1, \ldots, \hat{\mathbf{h}}_n)$ a sequence of input vectors to the same attention layer for $\hat{T}$, where for all i, $\|\mathbf{h}_i - \hat{\mathbf{h}}_i\|_1 \leq \epsilon$ for some $\epsilon \leq 1$. There exists a constant $K \geq 1$ depending on d and $p_{\max}$ such that for the attention scores $s_{i,j}$ we have for all i, j,*

$$|s_{i,j} - \hat{s}_{i,j}| \leq K h_{\max} \epsilon$$

where $h_{\max}$ is the maximum of 1 and the absolute value of any entry in any $\mathbf{h}_j$.

Proof. The error in the query and key vectors at positions i and j, respectively, are, by Lemma 28,

$$\|\mathbf{q}_i - \hat{\mathbf{q}}_i\|_1 = \left\|\mathbf{W}^{(\mathrm{Q})}\mathbf{h}_i - \mathbf{W}^{(\mathrm{Q})}\hat{\mathbf{h}}_i\right\| \leq d p_{\max} \epsilon$$
$$\|\mathbf{k}_j - \hat{\mathbf{k}}_j\|_1 = \left\|\mathbf{W}^{(\mathrm{K})}\mathbf{h}_j - \mathbf{W}^{(\mathrm{K})}\hat{\mathbf{h}}_j\right\| \leq d p_{\max} \epsilon.$$

Then the error in the attention scores is

$$\begin{aligned}
&|s_{i,j} - \hat{s}_{i,j}| \\
&\quad = \frac{1}{\sqrt{d}} \left| \mathbf{q}_i \cdot \mathbf{k}_j - \hat{\mathbf{q}}_i \cdot \hat{\mathbf{k}}_j \right| \\
&\quad = \frac{1}{\sqrt{d}} \big| \mathbf{q}_i \cdot \mathbf{k}_j - (\mathbf{q}_i + (\hat{\mathbf{q}}_i - \mathbf{q}_i)) \\
&\qquad\qquad \cdot (\mathbf{k}_j + (\hat{\mathbf{k}}_j - \mathbf{k}_j)) \big| \\
&\quad = \frac{1}{\sqrt{d}} \big| - \mathbf{q}_i \cdot (\hat{\mathbf{k}}_j - \mathbf{k}_j) - (\hat{\mathbf{q}}_i - \mathbf{q}_i) \cdot \mathbf{k}_j \\
&\qquad\qquad - (\hat{\mathbf{q}}_i - \mathbf{q}_i) \cdot (\hat{\mathbf{k}}_j - \mathbf{k}_j) \big|
\end{aligned}$$

and observing that $|\mathbf{a} \cdot \mathbf{b}| \leq \|\mathbf{a}\|_1 \|\mathbf{b}\|_1$ and $\epsilon \leq 1 \leq h_{\max}$, we get

$$\begin{aligned}
&|s_{i,j} - \hat{s}_{i,j}| \\
&\quad \leq \frac{1}{\sqrt{d}} (2 d^2 p_{\max}^2 h_{\max} \epsilon + d^2 p_{\max}^2 \epsilon^2) \\
&\quad \leq 3 d^{3/2} p_{\max}^2 h_{\max} \epsilon. \qquad \square
\end{aligned}$$

We now turn to bounding the error in the resulting attention weights $\alpha_{i,j}$.

Lemma 32. *Let $\tau > 0$. Let $\mathbf{s} = (s_1, \ldots, s_n)$ be a sequence of scores for T with gap γ, and let $\hat{\mathbf{s}} = (\hat{s}_1, \ldots \hat{s}_n)$ be a sequence of scores for $\hat{T}$, respectively, such that for all i, $|s_i - \hat{s}_i| \leq \epsilon$. Then*

$$\|\mathrm{ahardmax}\, \mathbf{s} - \mathrm{softmax}_\tau\, \hat{\mathbf{s}}\|_1 \leq 2ne^{-\gamma/\tau} + 4\epsilon/\tau.$$

Proof. There are two sources of error, the softmax and the approximation of $\mathbf{s}$:

$$\begin{aligned}
&\|\mathrm{ahardmax}\, \mathbf{s} - \mathrm{softmax}_\tau\, \hat{\mathbf{s}}\|_1 \\
&\quad \leq \|\mathrm{ahardmax}\, \mathbf{s} - \mathrm{softmax}_\tau\, \mathbf{s}\|_1 \\
&\qquad + \|\mathrm{softmax}_\tau\, \mathbf{s} - \mathrm{softmax}_\tau\, \hat{\mathbf{s}}\|_1 .
\end{aligned}$$

First, by Lemma B.7 of Edelman et al. (2022),

$$\|\mathrm{ahardmax}\, \mathbf{s} - \mathrm{softmax}\, \mathbf{s}\|_1 \leq 2ne^{-\gamma/\tau}.$$

Second (cf. Theorem 14f of Chiang (2025)), for all $i \in [n]$,

$$\begin{aligned}
[\mathrm{softmax}_\tau\, \hat{\mathbf{s}}]_i &= \frac{e^{\hat{s}_i/\tau}}{\sum_j e^{\hat{s}_j/\tau}} \\
&\leq \frac{e^{(s_i+\epsilon)/\tau}}{\sum_j e^{(s_j-\epsilon)/\tau}} = \frac{e^{(s_i+2\epsilon)/\tau}}{\sum_j e^{s_j/\tau}} \\
&= e^{2\epsilon/\tau} [\mathrm{softmax}_\tau\, \mathbf{s}]_i.
\end{aligned}$$

If $\epsilon \leq \tau/2$, then

$$[\mathrm{softmax}_\tau\, \hat{\mathbf{s}}]_i \leq (1 + 4\epsilon/\tau)[\mathrm{softmax}_\tau\, \mathbf{s}]_i$$

and by similar reasoning,

$$[\mathrm{softmax}_\tau\, \hat{\mathbf{s}}]_i \geq (1 - 4\epsilon/\tau)[\mathrm{softmax}_\tau\, \mathbf{s}]_i.$$

Then

$$\begin{aligned}
&\|\mathrm{softmax}_\tau\, \mathbf{s} - \mathrm{softmax}_\tau\, \hat{\mathbf{s}}\|_1 \\
&\quad \leq \sum_i 4\epsilon/\tau [\mathrm{softmax}_\tau\, \mathbf{s}]_i = 4\epsilon/\tau.
\end{aligned}$$

On the other hand, if $\epsilon > \tau/2$, then $4\epsilon/\tau > 2$, and $\|\mathrm{softmax}_\tau\, \mathbf{s} - \mathrm{softmax}_\tau\, \hat{\mathbf{s}}\|_1$ cannot exceed 2. $\square$

The next stage of an attention layer averages the vectors $\mathbf{W}^{(\mathrm{V})}\mathbf{h}_j$ weighted by the attention weights $\alpha_{i,j}$ and adds the residuals. The following bounds the error in the result in terms of the errors in the approximations of $\alpha_{i,j}$ and $\mathbf{h}_j$.

Lemma 33. *Let $(\mathbf{h}_1, \ldots, \mathbf{h}_n)$ and $(\hat{\mathbf{h}}_1, \ldots, \hat{\mathbf{h}}_n)$ be sequences of elements of $\mathbb{R}^d$ such that $\|\mathbf{h}_i - \hat{\mathbf{h}}_i\|_1 \leq \epsilon$ for all i. For each i, let $(\alpha_{i,1}, \ldots, \alpha_{i,n})$ and $(\hat{\alpha}_{i,1}, \ldots, \hat{\alpha}_{i,n})$ be sequences of real numbers in $[0, 1]$ that each sum to 1 such that $\|\alpha_{i,*} - \hat{\alpha}_{i,*}\|_1 \leq \epsilon_1$. Let $\mathbf{c}_i = \sum_{j=1}^n \alpha_{i,j} \mathbf{W}^{(\mathrm{V})}\mathbf{h}_\mathbf{j} + \mathbf{h}_\mathbf{i}$ and $\hat{\mathbf{c}}_i = \sum_{j=1}^n \hat{\alpha}_{i,j} \mathbf{W}^{(\mathrm{V})}\hat{\mathbf{h}}_\mathbf{j} + \hat{\mathbf{h}}_\mathbf{i}$. Then for all i,*

$$\|\mathbf{c}_i - \hat{\mathbf{c}}_i\|_1 \leq d p_{\max}(d h_{\max} \epsilon_1 + \epsilon) + \epsilon$$

where $h_{\max}$ is the maximum absolute value of any entry in any $\mathbf{h}_j$.

Proof. Fix position i. Then

$$\mathbf{c}_i - \hat{\mathbf{c}}_i = \mathbf{W}^{(\mathrm{V})} \sum_{j=1}^{n} (\alpha_{i,j}\mathbf{h}_j - \hat{\alpha}_{i,j}\hat{\mathbf{h}}_j) + \mathbf{h}_i - \hat{\mathbf{h}}_i.$$

Fix position j. Then

$$\begin{aligned}
&\left\|\alpha_{i,j}\mathbf{h}_j - \hat{\alpha}_{i,j}\hat{\mathbf{h}}_j\right\|_1 \\
&\quad = \left\|\alpha_{i,j}\mathbf{h}_j - \hat{\alpha}_{i,j}\mathbf{h}_j + \hat{\alpha}_{i,j}\mathbf{h}_j - \hat{\alpha}_{i,j}\hat{\mathbf{h}}_j\right\|_1 \\
&\quad \le |\alpha_{i,j} - \hat{\alpha}_{i,j}| \left\|\mathbf{h}_j\right\|_1 + \hat{\alpha}_{i,j}\left\|\mathbf{h}_j - \hat{\mathbf{h}}_j\right\|_1 \\
&\quad \le |\alpha_{i,j} - \hat{\alpha}_{i,j}| \left\|\mathbf{h}_j\right\|_1 + \hat{\alpha}_{i,j}\epsilon.
\end{aligned}$$

This can be used to show

$$\begin{aligned}
&\left\| \sum_{j=1}^{n} (\alpha_{i,j}\mathbf{h}_j - \hat{\alpha}_{i,j}\hat{\mathbf{h}}_j) \right\| \\
&\quad \le \sum_{j=1}^{n} \left\|\alpha_{i,j}\mathbf{h}_j - \hat{\alpha}_{i,j}\hat{\mathbf{h}}_j\right\|_1 \\
&\quad \le \sum_{j=1}^{n} \left(|\alpha_{i,j} - \hat{\alpha}_{i,j}| \left\|\mathbf{h}_j\right\|_1 + \hat{\alpha}_{i,j}\epsilon \right) \\
&\quad \le d h_{\max} \epsilon_1 + \epsilon.
\end{aligned}$$

Finally, the multiplication by $\mathbf{W}^{(\mathrm{V})}$ multiplies this bound by at most $d p_{\max}$, and the residual connection adds at most another ϵ. □

C.3 Proof of Lemma 25

Proof. Let $\mathbf{g}_i$, $\hat{\mathbf{g}}_i$, $\mathbf{h}_i$, and $\hat{\mathbf{h}}_i$ for $i \in [n]$ be as in the statement of the lemma. Assume that for all $i \in [n]$, $\|\mathbf{g}_i - \hat{\mathbf{g}}_i\|_1 \le \epsilon \le 1$.

By Lemmas 29 and 31, there exist constants $K_1, K_2 \ge 1$ (not depending on ℓ) such that for $i, j \in [n]$,

$$\begin{aligned}
|s_{i,j} - \hat{s}_{i,j}| &\le K_1 h_{\max}(n)\epsilon \\
&\le K_2 x_{\max}(n)\epsilon.
\end{aligned}$$

By Lemma 32, for all $i \in [n]$,

$$\|\alpha_{i,*} - \hat{\alpha}_{i,*}\|_1 \le 2ne^{-\frac{\gamma(n)}{\tau(n)}} + \frac{4K_2 x_{\max}(n)\epsilon}{\tau(n)}.$$

By Lemmas 29 and 33, there exist constants $K_3, K_4, K_5 \ge 1$ (not depending on ℓ) such that for all $i \in [n]$,

$$\begin{aligned}
&\|\mathbf{c}_i - \hat{\mathbf{c}}_i\|_1 \\
&\le K_3 x_{\max} \left(ne^{-\frac{\gamma(n)}{\tau(n)}} + \frac{x_{\max}(n)\epsilon}{\tau(n)} \right) + K_4 h_{\max} \\
&\le K_5 x_{\max} \left(ne^{-\frac{\gamma(n)}{\tau(n)}} + \frac{x_{\max}(n)\epsilon}{\tau(n)} \right).
\end{aligned}$$

Finally, by Lemma 30, there exists a constant $K \ge 1$ (not depending on ℓ) such that

$$\left\|\mathbf{h}_i - \hat{\mathbf{h}}_i\right\| \le K x_{\max}(n) \left(ne^{-\frac{\gamma(n)}{\tau(n)}} + \frac{x_{\max}(n)}{\tau(n)}\epsilon \right).$$

□

C.4 Proof of Theorem 26

Proof. Fix an input length n. For brevity, we write $\min(\gamma(n), 1)$, $\max(x_{\max}(n), 1)$, and $\min(\tau(n), 1)$ as γ, $x_{\max}$, and τ, respectively.

Fix an input sequence of vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. Let us write the bound from Lemma 25 as

$$\begin{aligned}
\epsilon_\ell &= \left\|\mathbf{h}_i - \hat{\mathbf{h}}_i\right\| \le a + r\epsilon_{\ell-1} \\
a &= K x_{\max} n e^{\gamma/\tau} \\
r &= K x_{\max}^2 / \tau.
\end{aligned}$$

Since the error ϵ_ℓ at each layer is bounded by an affine function of the error at the layer below, the final error is the sum of a geometric series:

$$\begin{aligned}
\epsilon_L = \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_1 &= a + ar + \cdots + ar^{L-1} \\
&\le L a r^{L-1} \\
&= LKnx_{\max} e^{-\gamma/\tau} \left(\frac{Kx_{\max}^2}{\tau} \right)^{L-1}.
\end{aligned}$$

Taking logs of both sides and using the inequality $\log r \le \log r_0 + \frac{1}{r_0}(r - r_0)$ based on the first-order Taylor approximation, with r as above and $r_0 = 2K(L-1)x_{\max}^2/\gamma$, we get

$$\begin{aligned}
\log \epsilon_L &= \log \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_1 \\
&\le \log Knx_{\max} - \frac{\gamma}{\tau} + (L-1)\log r \\
&\le \log Knx_{\max} - \frac{\gamma}{\tau} + (L-1)\left(\log r_0 + \frac{r - r_0}{r_0} \right) \\
&\le \log Knx_{\max} - \frac{\gamma}{2\tau} + (L-1)(\log r_0 - 1).
\end{aligned}$$

We want ϵ_L to be bounded by the given error function $\epsilon(n)$. If α and β are such that $\epsilon(n) \leq \alpha/n^\beta$, then we need

$$\log Knx_{\max} - \frac{\gamma}{2\tau} + (L-1)(\log r_0 - 1) \leq \log \alpha/n^\beta$$

which is obtained with the temperature bound

$$\frac{1}{\tau} \geq \frac{2}{\gamma}\left(\log \frac{Kn^{\beta+1}x_{\max}}{\alpha} + (L-1)(\log r_0 - 1)\right) \in O\left(\frac{1}{\gamma}\log\frac{nx_{\max}}{\gamma}\right). \qquad \square$$

D S-RASP Program for DYCK-k

The following S-RASP program (Strobl et al., 2025) returns, for each prefix of the input (in), whether the prefix belongs to DYCK-k, the language of balanced strings of brackets with k types of brackets. We assume that $\texttt{left}(\sigma)$ returns 1 if σ is a left bracket and 0 otherwise, $\texttt{right}(\sigma)$ returns 1 if σ is a right bracket and 0 otherwise, and $\texttt{mismatch}(\sigma, \tau)$ returns $\top$ if σ and τ are not left and right brackets of the same type.

$$\begin{aligned}
\texttt{sleft}(i) &= \textbf{psum}_j\,[j \leq i]\ \texttt{left}(\texttt{in}(i)) \\
\texttt{sright}(i) &= \textbf{psum}_j\,[j \leq i]\ \texttt{right}(\texttt{in}(i)) \\
\texttt{er1}(i) &= \texttt{sright}(i) > \texttt{sleft}(i) \\
\texttt{diff}(i) &= \texttt{sleft}(i) - \texttt{sright}(i) \\
\texttt{d}(i) &= \texttt{diff}(i) + \texttt{right}(\texttt{in}(i)) \\
\texttt{check}(i) &= \blacktriangleright_j\,[j < i, \texttt{d}(j) = \texttt{d}(i)]\ \texttt{in}(j) : \text{`?'} \\
\texttt{er2}(i) &= (\texttt{right}(\texttt{in}(i)) = 1) \wedge \\
&\qquad \texttt{mismatch}(\texttt{check}(i), \texttt{in}(i)) \\
\texttt{okprefix}(i) &= \blacktriangleright_j\,[j \leq i, \texttt{er1}(j) \vee \texttt{er2}(j)]\ \bot : \top \\
\texttt{out}(i) &= \texttt{okprefix}(i) \wedge (\texttt{diff}(i) = 0)
\end{aligned}$$

$\texttt{sleft}(i)$ and $\texttt{sright}(i)$ are the numbers of left and right (resp.) brackets of any type through position i. An error ($\texttt{er1}(i)$) is recorded wherever there are more right than left brackets through position i. $\texttt{d}(i)$ is the depth of the bracket at i. $\texttt{check}(i)$ records the nearest bracket strictly to the left that has the same depth. An error ($\texttt{er2}(i)$) is recorded wherever there is a right bracket and $\texttt{check}(i)$ is not a left bracket of the same type. Finally, the output $\texttt{out}(i)$ for i is $\top$ iff the prefix through i has no errors ($\texttt{okprefix}(i)$) and contains an equal number of left and right brackets.