

Expressivity of Transformers: Logic, Circuits, and Formal Languages

Day 2: Transformers and Star-Free Languages

David Chiang (Univ. of Notre Dame, USA)

Jon Rawski (MIT/San Jose State Univ., USA)

Lena Strobl (Umeå University, Sweden)

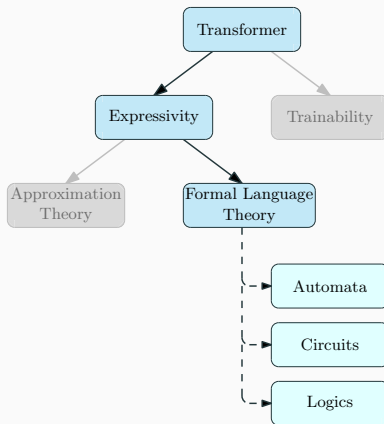
Andy Yang (Univ. of Notre Dame, USA)

30 July 2024



Recap

Recap: Situating ourselves



Recap: Current results

Seemingly contradictory

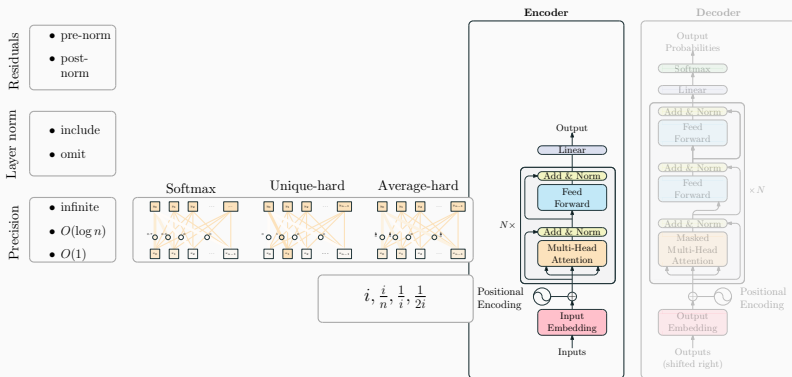
Lower bound	Source	PE	Attention	Notes
⊇ MAJORITY	Pérez et al. 2019	none	average-hard	poly(n) steps
⊇ SHUFFLE-DYCK- k	Bhattachishra et al. 2020a	none	softmax, future mask	
⊇ SSCMs	Bhattachishra et al. 2020a	none	softmax, future mask	
⊇ DYCK- k	Yao et al. 2021	$i/n, i/n^3, n$	softmax & leftmost-hard	
⊇ P	Pérez et al. 2021	$i, 1/i, 1/i^2$	average-hard	
⊇ PARITY	Chiang and Cholak 2022	$i/n, (-1)^i$	softmax	
⊇ FOC[MOD; +]	Chiang et al. 2023	sinusoidal	softmax	
⊇ FO[Mon]	Barceló et al. 2024	arbitrary	leftmost-hard	
⊇ LTL+C[Mon]	Barceló et al. 2024	arbitrary	average-hard	
Upper bound	Source	Precision	Attention	Notes
⊆ PARITY, DYCK-1	Hahn 2020	R	leftmost-hard	$\varepsilon_N > 0$, vanishing KL
⊆ PARITY, DYCK-2	Hahn 2020	R	softmax, future mask	
⊆ AC ⁰	Hao et al. 2022	Q	leftmost-hard	
⊆ TC ⁰	Merrill et al. 2022	F	average-hard	
⊆ FOC[MOD; +]	Chiang et al. 2023	$O(1)$	softmax	
⊆ L-uniform TC ⁰	Merrill & Sabharwal 2023a	$O(\log n)$	softmax	
⊆ FOM[BIT]	Merrill & Sabharwal 2023b	$O(\log n)$	softmax	
⊆ L-uniform TC ⁰	Strobl 2023	F	average-hard	
Equivalent	Source	PE	Attention	Notes
= RE	Pérez et al. 2021	$i, 1/i, 1/i^2$	average-hard	unbounded steps
= FO	Angluin et al. 2023	none	rightmost-hard, strict future mask	
= FO[MOD]	Angluin et al. 2023	sinusoidal	rightmost-hard, strict future mask	
= FO[Mon]	Angluin et al. 2023	arbitrary	rightmost-hard, strict future mask	
= P	Merrill & Sabharwal 2024	none	average-hard, future mask	poly(n) steps

Recap: Current results

The big picture

Lower bound	Source	PE	Attention	Notes
⊇ MAJORITY	Pérez et al. 2019	none	average-hard	poly(n) steps
⊇ SHUFFLE-DYCK- k	Bhattachishra et al. 2020a	none	softmax, future mask	
⊇ SSCMs	Bhattachishra et al. 2020a	none	softmax, future mask	
⊇ DYCK- k	Yao et al. 2021	$i/n, i/n^3, n$	softmax & leftmost-hard	
⊇ P	Pérez et al. 2021	$i, 1/i, 1/i^2$	average-hard	
⊇ PARITY	Chiang and Cholak 2022	$i/n, (-1)^i$	softmax	
⊇ FOC[MOD;+]	Chiang et al. 2023	sinusoidal	softmax	
⊇ FO[Mon]	Barceló et al. 2024	arbitrary	leftmost-hard	
⊇ LTL+C[Mon]	Barceló et al. 2024	arbitrary	average-hard	
Upper bound	Source	Precision	Attention	Notes
⊆ PARITY, DYCK-1	Hahn 2020	R	leftmost-hard	$\varepsilon_N > 0$, vanishing KL
⊆ PARITY, DYCK-2	Hahn 2020	R	softmax, future mask	
⊆ AC ⁰	Hao et al. 2022	Q	leftmost-hard	
⊆ TC ⁰	Merrill et al. 2022	F	average-hard	
⊆ FOC[MOD;+]	Chiang et al. 2023	$O(1)$	softmax	
⊆ L-uniform TC ⁰	Merrill & Sabharwal 2023a	$O(\log n)$	softmax	
⊆ FOM[BIT]	Merrill & Sabharwal 2023b	$O(\log n)$	softmax	
⊆ L-uniform TC ⁰	Strobl 2023	F	average-hard	
Equivalent	Source	PE	Attention	Notes
= RE	Pérez et al. 2021	$i, 1/i, 1/i^2$	average-hard	unbounded steps
= FO	Angluin et al. 2023	none	rightmost-hard, strict future mask	poly(n) steps
= FO[MOD]	Angluin et al. 2023	sinusoidal	rightmost-hard, strict future mask	
= FO[Mon]	Angluin et al. 2023	arbitrary	rightmost-hard, strict future mask	
= P	Merrill & Sabharwal 2024	none	average-hard, future mask	

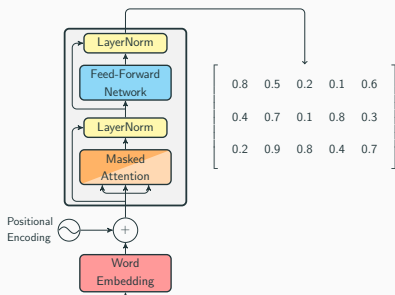
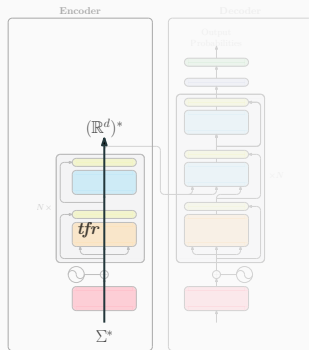
Recap: Lots of assumptions to make



More in depth: Transformer Encoders

A transformer encoder is a function mapping sequences of strings to sequences of real vectors.

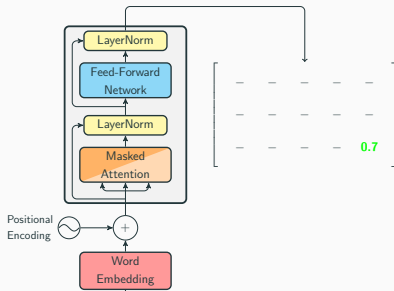
$$tfr: \Sigma^* \xrightarrow{lp} (\mathbb{R}^d)^*$$



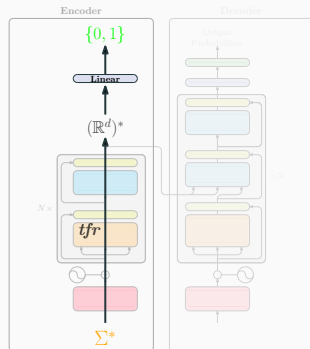
I like to eat croissants

More in depth: Transformer Encoders

Often, we designate the transformer as accepting w if the last value in the last vector $tfr(w)[d, n] \geq 0$ is above some threshold.

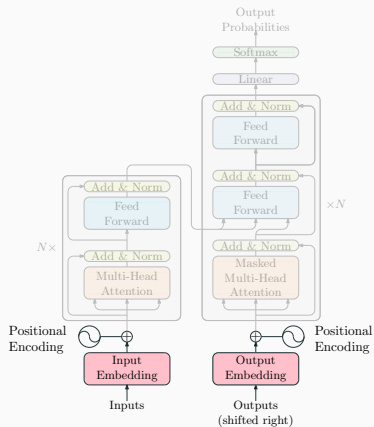


I like to eat croissants



More in depth: Word Embeddings

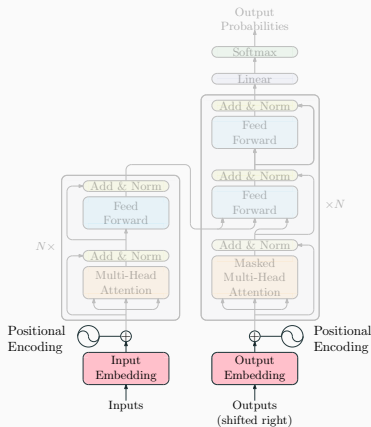
$\begin{bmatrix} 0.8 & 0.5 & 0.2 & 0.1 & 0.6 \\ 0.4 & 0.7 & 0.1 & 0.8 & 0.3 \\ 0.2 & 0.9 & 0.8 & 0.4 & 0.7 \end{bmatrix}$				
↑	↑	↑	↑	↑
I	like	to	eat	croissants



More in depth: Word Embeddings (without PEs)

Without positional embeddings (or masking, to be explained later) the transformer cannot distinguish the order of words

$\begin{bmatrix} 0.8 & 0.5 & 0.2 & 0.1 & 0.6 \\ 0.4 & 0.7 & 0.1 & 0.8 & 0.3 \\ 0.2 & 0.9 & 0.8 & 0.4 & 0.7 \end{bmatrix}$				
↑	↑	↑	↑	↑
croissants	like	to	eat	I



More in depth: Attention layer

Scaled dot-product self-attention with d input/output dimensions and d_{hid} key/value dimensions is a function

$$\text{att}: (\mathbb{R}^d)^* \xrightarrow{\text{lp}} (\mathbb{R}^d)^*$$

$$\mathbf{X} \mapsto \mathbf{Y}$$

$$\mathbf{Q}[i] = \mathbf{W}^{\text{Q}}(\mathbf{X}[i])$$

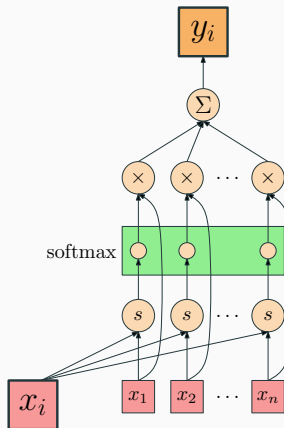
$$\mathbf{K}[j] = \mathbf{W}^{\text{K}}(\mathbf{X}[j])$$

$$\mathbf{V}[j] = \mathbf{W}^{\text{V}}(\mathbf{X}[j])$$

$$s[i, j] = \frac{\mathbf{Q}[i] \cdot \mathbf{K}[j]}{\sqrt{d_{\text{hid}}}}$$

$$\alpha[i, :] = \text{softmax}(s[i, :])$$

$$\mathbf{Y}[i] = \sum_{j \in [n]} \alpha[i, j] \mathbf{V}[j]$$



More in depth: Feed-Forward

A *linear layer* is a function

$$\text{lin}: \mathbb{R}^d \rightarrow \mathbb{R}^{d'} \quad (1)$$

$$\mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b} \quad (2)$$

A *feed-forward neural network* (FFNN) is a function

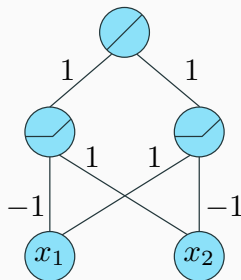
$$\text{ffn}: \mathbb{R}^{d_h} \rightarrow \mathbb{R}^{d_h}$$

$$\mathbf{x} \mapsto \mathbf{y} \text{ where}$$

$$\mathbf{h} = \text{ReLU}(\text{lin}_1(\mathbf{x}))$$

$$\mathbf{y} = \text{lin}_2(\mathbf{h})$$

Note that $\text{ReLU}(x) = \max(x, 0)$



More in depth: Layer composition

A transformer is a composition of attention and feed-forward layers.

$$tfr: \Sigma^* \xrightarrow{lp} (\mathbb{R}^d)^*$$

$$att: (\mathbb{R}^d)^* \xrightarrow{lp} (\mathbb{R}^d)^*$$

$$ffn: \mathbb{R}^d \rightarrow \mathbb{R}^d$$

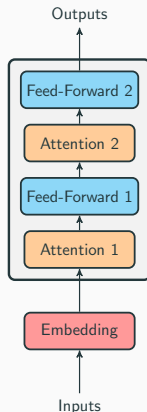
$$WE: \Sigma \rightarrow \mathbb{R}^d$$

Extend WE to sequences, elementwise.

Then:

$$tfr = ffn_2 \circ att_2 \circ ffn_1 \circ att_1 \circ att \circ WE$$

This is the “depth” in “deep learning”.

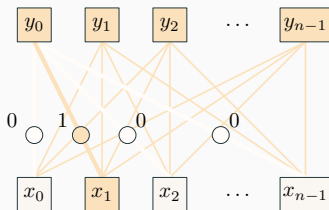


Today's goals and question

Today's Goals

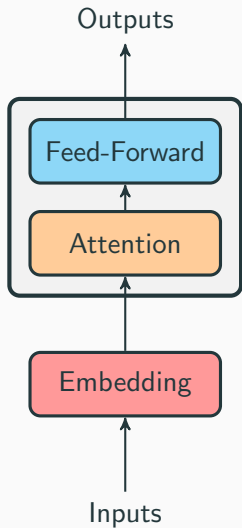
0. **Review** the definition of transformers
1. **Explain** the difference between masked hard-attention transformers and soft attention transformers
2. **Show** the correspondence between masked hard-attention transformers and LTL
3. **Describe** how strict-masking, positional embeddings, and depth contribute to the expressivity of a masked hard-attention transformer

Unique-hard attention transformers correspond closely with LTL

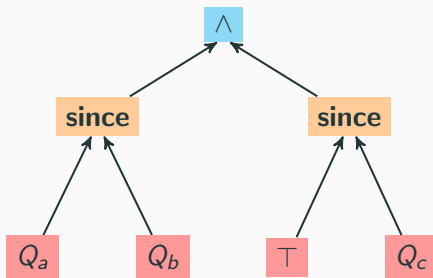


This allows us to bring results from one domain to the other.

LTl Correspondence



$$w \models (Q_a \text{ since } Q_b) \wedge (\top \text{ since } Q_c)$$



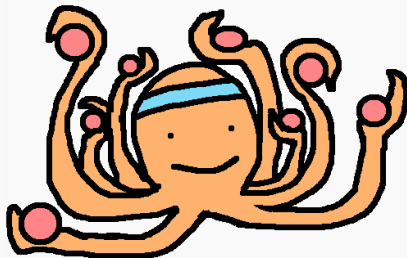
$$w \in \Sigma^*$$

Analyzing masked hard-attention transformers

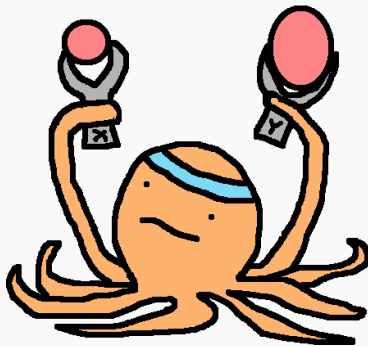
Logic is a powerful tool for describing computation.

- Quantifier depth corresponds to parallel computation time
- Number of variables corresponds to number of processors

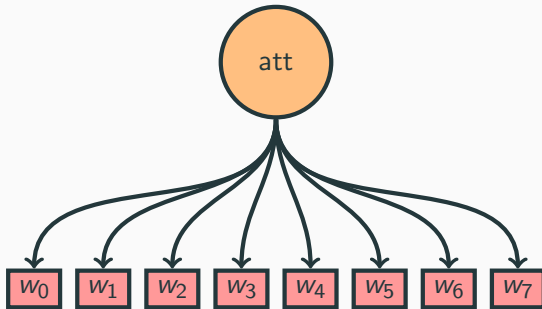
otto $\models \forall x.\text{ball}(x)$



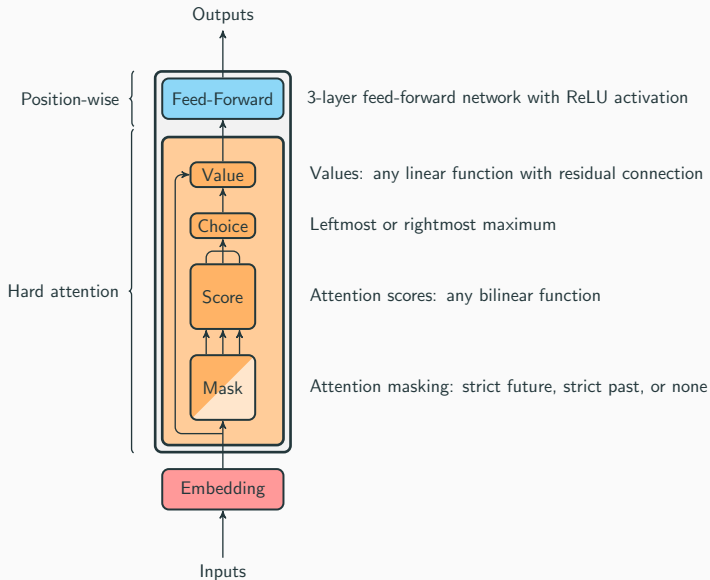
otto $\models \text{size}(x) < \text{size}(y)$



Self-Attention



Masked Hard-Attention Transformers



Definition (Hardmax)

The n -dimensional rightmost hardmax function is the function $\text{rhardmax}: \mathbb{R}^n \rightarrow \mathbb{R}^n$ such that

$$\begin{aligned} [\text{rhardmax}(s)]_i &= \mathbb{I}[i = \max I] \\ I &= \{i \in [n] \mid s_i = \max s\}. \end{aligned}$$

That is, it returns the one-hot vector with a 1 in the *rightmost* coordinate that maximizes s .

Unique Hard Self-Attention (in case needed)

Scaled dot-product self-attention with d input/output dimensions and d_{hid} key/value dimensions is a function

$$\text{att}: (\mathbb{R}^d)^* \xrightarrow{\text{lp}} (\mathbb{R}^d)^*$$

$$\mathbf{X} \mapsto \mathbf{Y}$$

$$\mathbf{Q}[i] = \mathbf{W}^{\mathbf{Q}}(\mathbf{X}[i])$$

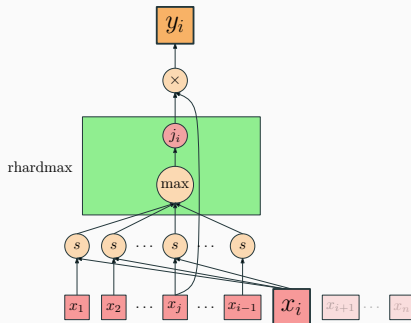
$$\mathbf{K}[j] = \mathbf{W}^{\mathbf{K}}(\mathbf{X}[j])$$

$$\mathbf{V}[j] = \mathbf{W}^{\mathbf{V}}(\mathbf{X}[j])$$

$$s[i, j] = \frac{\mathbf{Q}[i] \cdot \mathbf{K}[j]}{\sqrt{d_{\text{hid}}}}$$

$$\alpha[i, :] = \text{rhardmax}(\mathbf{s}[i, :])$$

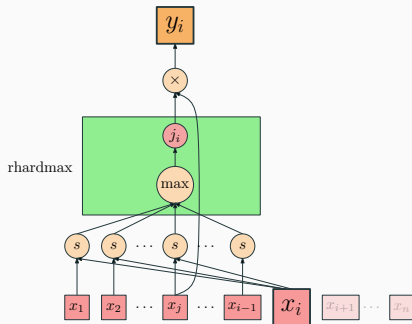
$$\mathbf{Y}[i] = \sum_{j \in [n]} \alpha[i, j] \mathbf{V}[j]$$



Masked Unique-Hard Attention Distilled

For each $\mathbf{X}$, do the following to get $att(\mathbf{X})$:

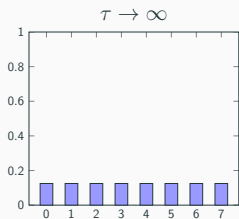
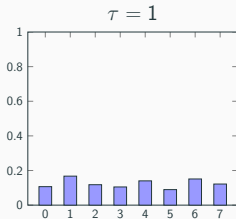
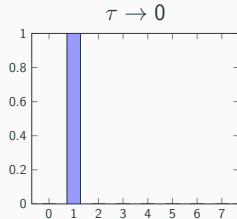
1. If $i = 0$, return $\mathbf{0}$
2. Compute $\mathbf{s}[i, j] = \frac{\mathbf{Q}[i] \cdot \mathbf{K}[j]}{\sqrt{d_{\text{hid}}}}$ for each $j < i$
3. Let $j_i = \text{rhardmax}_j \mathbf{s}[i, j]$
4. Return $\mathbf{V}[j_i]$



Hard Attention

At low temperature, softmax closely approximates rhardmax – selecting a single input. Assuming, there is some tiebreaking.

$$\text{softmax}(x_i, \tau) = \frac{e^{x_i/\tau}}{\sum_{j=0}^n e^{x_j/\tau}}$$



Hard Attention in the Wild

- Real transformers are often observed to focus attention on a very small number of positions [Merrill et al., 2021].
- On Dyck languages, they have been found to learn effectively unique-hard attention in their second layer [Ebrahimi et al., 2020, Fig. 1]. [Chiang and Cholak, 2022], but in practice, transformers cannot learn parity [Bhattamishra et al., 2020]. Unique-hard attention transformers also cannot compute parity [Hahn, 2020], so they are in some sense more realistic.
- Hard attention has occasionally been used in practice in previous research on interpretability [Kinley, 2020] and efficiency [Gupta et al., 2021, Xu et al., 2021].

Goal 1: COMPLETE

We now know the difference between rightmost hard and soft-attention.

What is the right sort of quantification and amount of variables to describe (unique-hard) transformer computation?

A Review of LTL

For input string $\mathbf{w} = \mathbf{w}_1 \cdots \mathbf{w}_n$ and position $i \in [n]$, we define $\mathbf{w}, i \models \phi$ as follows:

$\mathbf{w}, i \models Q_a$ $\mathbf{w}_i = a$

$\mathbf{w}, i \models \phi_1 \vee \phi_2$ $\mathbf{w}, i \models \phi_1$ or $\mathbf{w}, i \models \phi_2$

$\mathbf{w}, i \models \neg \phi_1$ $\mathbf{w}, i \not\models \phi_1$

$\mathbf{w}, i \models \phi_1$ **since** ϕ_2 for some $j < i$, we have $\mathbf{w}, j \models \phi_2$, and
for all k such that $j < k < i$, we have $\mathbf{w}, k \models \phi_1$

For an input string $\mathbf{w} \in \Sigma^+$ of length n we write $\mathbf{w} \models \phi$ if and only if $\mathbf{w}, n \models \phi$.

A Review of LTL

"I am an a"

	a	b	c	a	b	b	b
Q_a	1	0	0	1	0	0	0
Q_b	0	1	0	0	1	1	1
Q_c	0	0	1	0	0	0	0
$Q_a \vee \neg Q_b$	1	0	1	1	0	0	0
Q_b since Q_a	0	1	1	0	1	1	1

$w, 0 \models Q_a$

A Review of LTL

"I am an a or I am not a b"

	1	b	c	a	b	b	b
Q_a	1	0	0	1	0	0	0
Q_b	0	1	0	0	1	1	1
Q_c	0	0	1	0	0	0	0
$Q_a \vee \neg Q_b$	1	0	1	1	0	0	0
Q_b since Q_a	0	1	1	0	1	1	1

$$\mathbf{w}, 3 \models Q_a \vee \neg Q_b$$

A Review of LTL

"I have a left neighbor that is an a and there are only bs between us"

	a	b	c	a	b	b	b
Q_a	1	0	0	1	0	0	0
Q_b	0	1	0	0	1	1	1
Q_c	0	0	1	0	0	0	0
$Q_a \vee \neg Q_b$	1	0	1	1	0	0	0
Q_b since Q_a	0	1	1	0	1	1	1

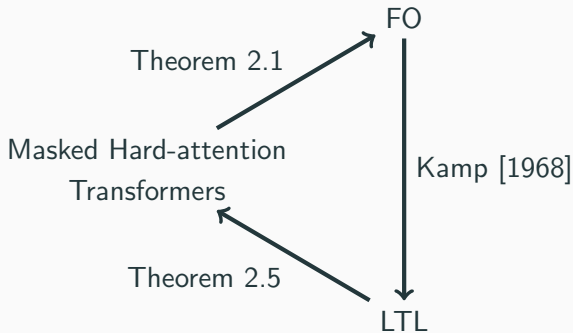
$w, 6 \models Q_b$ since Q_a

Compared to FO:

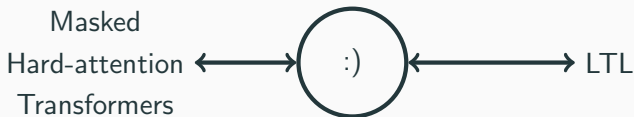
- Restricted variable use
- Restricted quantification

Equivalence

The Main Result

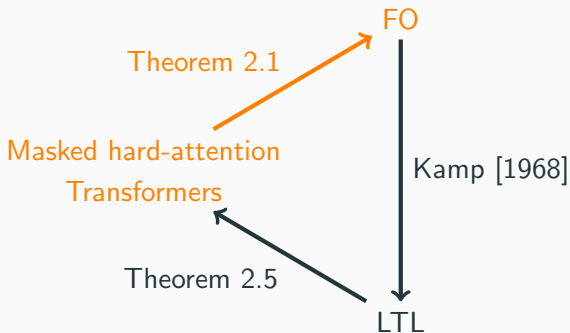


An Alternative Diagram



The direct proof in Yang et al. [2023] has more details. It's more complicated, but the results are important for later.

Masked hard-attention transformers to FO



Lemma

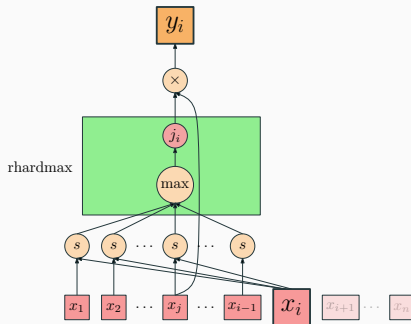
Let T be a unique (leftmost or rightmost) hard attention transformer. There is a finite set $\mathbb{F} \subseteq \mathbb{R}^d$ such that for any input string $\mathbf{w}$, all the activation vectors computed by $T(\mathbf{w})$ belong to $\mathbb{F}$.

Masked Unique-Hard Attention Distilled

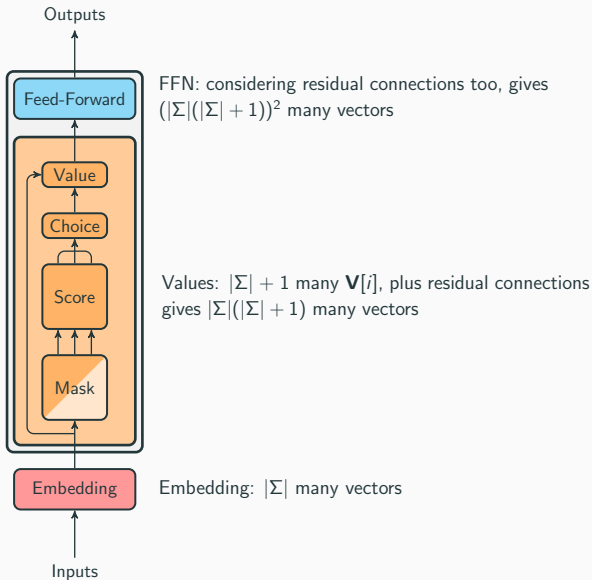
For each $\mathbf{X}$, do the following to get $att(\mathbf{X})$:

1. If $i = 0$, return $\mathbf{0}$
2. Compute $s[i, j] = \frac{Q[i] \cdot K[j]}{\sqrt{d_{\text{hid}}}}$ for each $j < i$
3. Let $j_i = \text{rhardmax}_j s[i, j]$
4. Return $\mathbf{V}[j_i]$

Observe, the output of $att(\mathbf{X})$ will be $\mathbf{V}[j_i]$ for some j_i in $[n]$

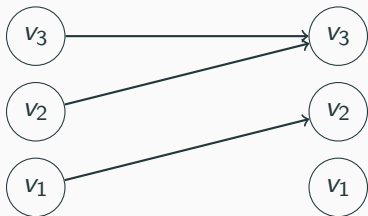


Thinking About Finite Activations



Finite Functions

Let $\mathbb{F} = \{v_1, v_2, v_3\}$. Consider the function $f: \mathbb{F} \rightarrow \mathbb{F}$ defined by $f(v_1) = v_2$, $f(v_2) = v_3$, and $f(v_3) = v_3$.



Let in_{v_i} indicate the input to f is v_i , and out_{v_i} for output. Then f is described by

$$(\text{out}_{v_3} \leftrightarrow (\text{in}_{v_2} \vee \text{in}_{v_3})) \wedge (\text{out}_{v_2} \leftrightarrow \text{in}_{v_1})$$

Consider embedding $\Sigma \rightarrow \mathbb{F}$. For each $\mathbf{v} \in \mathbb{F}$, we have a formula $wemb_{\mathbf{v}}(i)$ that says the embedding of the i -th symbol is $\mathbf{v}$

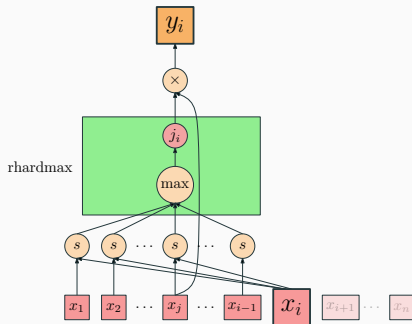
$$wemb_{\mathbf{v}}(i) = \bigvee_{\substack{a \in \Sigma \\ WE(a) = \mathbf{v}}} Q_a(i)$$

Similarly, feedforward networks from $\mathbb{F} \rightarrow \mathbb{F}$ can be described by FO formulas as mentioned before.

Recall, Masked Unique-Hard Attention Distilled

For each $\mathbf{X}$, do the following to get $att(\mathbf{X})$:

1. If $i = 0$, return $\mathbf{0}$
2. Compute $\mathbf{s}[i, j] = \frac{\mathbf{Q}[i] \cdot \mathbf{K}[j]}{\sqrt{d_{\text{hid}}}}$ for each $j < i$
3. Let $j_i = \text{rhardmax}_j \mathbf{s}[i, j]$
4. Return $\mathbf{V}[j_i]$



Recall, Masked Unique-Hard Attention Distilled

For each $\mathbf{X}$, do the following to get $att(\mathbf{X})$:

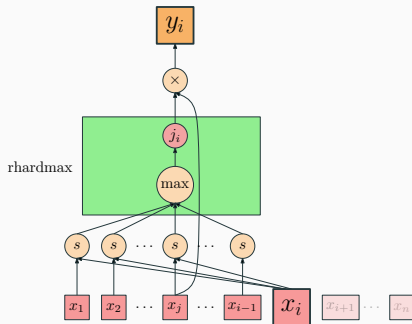
1. If $i = 0$, return $\mathbf{0}$

Can be done in FO similarly to before

2. Compute $\mathbf{s}[i, j] = \frac{\mathbf{Q}[i] \cdot \mathbf{K}[j]}{\sqrt{d_{\text{hid}}}}$ for each $j < i$

Similar

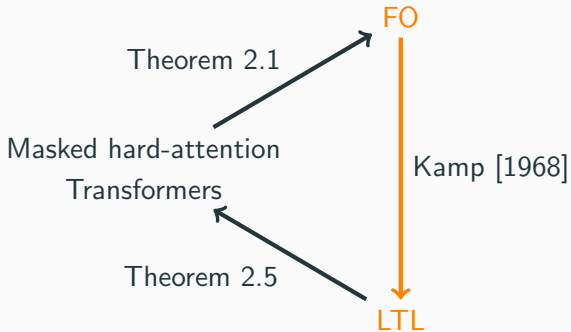
3. Let $j_i = \text{rhardmax}_j \mathbf{s}[i, j]$
4. Return $\mathbf{V}[j_i]$



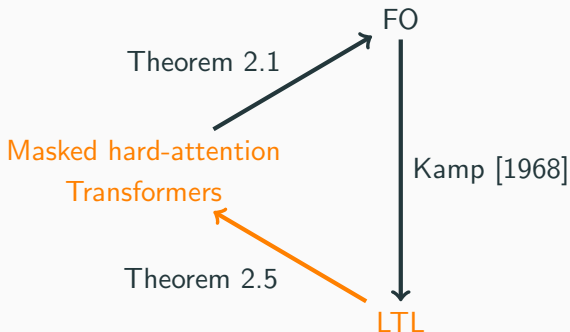
$$\text{compare}(i, j_1, j_2) = \bigvee_{\substack{v_1, v_2 \\ v_1 \leq v_2}} (\text{score}_{v_1}(i, j_1) \wedge \text{score}_{v_2}(i, j_2))$$

$$\begin{aligned} \text{rhardmax}(i, j) = & j < i \\ & \wedge \forall k [k < j \rightarrow \text{compare}(i, k, j)] \\ & \wedge \forall k [j < k \wedge k < i \rightarrow \neg \text{compare}(i, j, k)]. \end{aligned}$$

$$\text{att}_{\mathbf{v}}(i) = \begin{cases} \exists j[\text{rhardmax}(i, j) \wedge \text{value}_{\mathbf{v}}(j)] & \mathbf{v} \neq \mathbf{0} \\ \exists j[\text{rhardmax}(i, j) \wedge \text{value}_{\mathbf{v}}(j)] \vee \neg \exists j[j < i] & \mathbf{v} = \mathbf{0}. \end{cases}$$



LTL to Masked hard-attention transformers



Parallel Composition

Lemma (Parallel composition)

Given transformer encoders without layer normalization

$$tfr_1: \Sigma^* \xrightarrow{\text{lp}} (\mathbb{R}^{d_1})^*$$

$$tfr_2: \Sigma^* \xrightarrow{\text{lp}} (\mathbb{R}^{d_2})^*$$

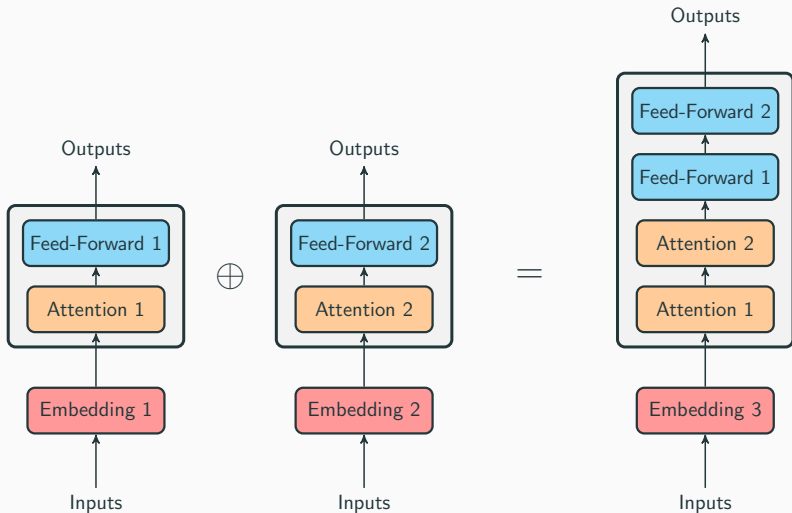
there is a transformer

$$tfr_1 \oplus tfr_2: \Sigma^* \xrightarrow{\text{lp}} (\mathbb{R}^{d_1+d_2})^*$$

such that for all strings $\mathbf{w} = w_0 \cdots w_{n-1} \in \Sigma^$,*

$$(tfr_1 \oplus tfr_2)(\mathbf{w}) = \begin{bmatrix} tfr_1(\mathbf{w})[0] \\ tfr_2(\mathbf{w})[0] \end{bmatrix} \cdots \begin{bmatrix} tfr_1(\mathbf{w})[n-1] \\ tfr_2(\mathbf{w})[n-1] \end{bmatrix}.$$

Parallel Composition Diagram



Theorem

For any LTL formula ϕ , there is a transformer tfr_ϕ and an index k such that

$$tfr_\phi(\mathbf{w})[i, k] = \begin{cases} 1 & \text{if } \mathbf{w}, i \models \phi \\ 0 & \text{otherwise} \end{cases}$$

We show this by induction on the structure of ϕ . Note, we also will notate this as $tfr_\phi(\mathbf{w})[i] = \mathbb{I}[w, i \models \phi]$.

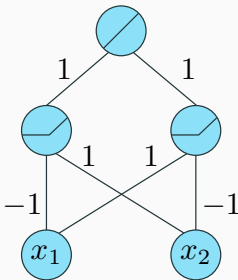
Base Case

Base case $\phi = Q_a$ for some $a \in \Sigma$: Then tfr_ϕ is just an embedding function

$$tfr_\phi(\mathbf{w})[i] = WE(i) = \begin{cases} 1 & \text{if } \mathbf{w}, i \models Q_a \\ 0 & \text{otherwise} \end{cases}. \quad (3)$$

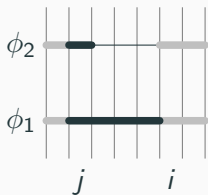
Boolean Functions

Use a feed-forward network to compute the boolean functions. As example, $\text{XOR}(x_1, x_2)$:



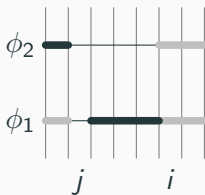
since

Case 1



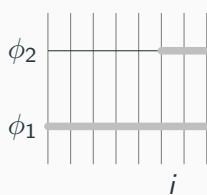
$w, i \models \phi_1$ since ϕ_2

Case 2



$w, i \not\models \phi_1$ since ϕ_2

Case 3



$w, i \not\models \phi_1$ since ϕ_2

1. The rightmost j satisfying ϕ_2 occurs at or beyond the rightmost j satisfying $\neg\phi_1$
2. The rightmost j satisfying ϕ_2 occurs before the rightmost j satisfying $\neg\phi_1$
3. There is no j satisfying ϕ_2

since

1. If $i = 0$, return $\mathbf{0}$
2. Compute $\mathbf{s}[i, j] = (\mathbf{w}, j \models \neg\phi_1 \vee \phi_2)$ for each $j < i$
3. Let $j_i = \text{rhardmax}_j \mathbf{s}[i, j]$
4. Return $\mathbf{V}[j_i] = \mathbf{w}, j_i \models \phi_2$

Goal 2: COMPLETE

We can now show how masked hard-attention transformers and LTL are equivalent.

Corollaries

This close correspondence between MUHATs and LTL allows us to apply results from one domain to the other.

LTL gives us a way to reason precisely about the computational resources a masked hard-attention transformer uses.

What is the difference in strict vs
non-strict masking?

Non-strict since is defined as

$\mathbf{w}, i \models \phi_1$ **since** ϕ_2 for some $j \leq i$, we have $\mathbf{w}, j \models \phi_2$, and
for all k such that $j < k \leq i$, we have $\mathbf{w}, k \models \phi_1$

Non-strict masking is the usual masking scheme where a position can attend to itself.

Stutter-Invariant Star-Free Languages

A language over Σ is called *stutter-invariant* iff for all $u, v \in \Sigma^*$ and $a \in \Sigma$, $uav \in L$ iff $uaav \in L$.

An example of a language that is star-free but not stutter-invariant is $\{a\}$.

Peled and Wilke [1997] prove non-strict LTL recognizes exactly the stutter-invariant star-free languages.

How do positional embeddings affect expressive power?

Positional Embeddings

Sinusoidal position embeddings For any even d , let us define a *rational sinusoidal positional embedding* with d dimensions to be a position embedding

$$\text{PE}_n(i) = \begin{bmatrix} \sin 2\pi f_1 i \\ \cos 2\pi f_1 i \\ \dots \\ \sin 2\pi f_{d/2} i \\ \cos 2\pi f_{d/2} i \end{bmatrix} \quad f_1, \dots, f_{d/2} \in \mathbb{Q}.$$

Admittedly, in the original definition, the f_k were not rational.

Modular Predicates For any $m \in \mathbb{N}$ and $r \in \{0, \dots, m-1\}$, we define the *modular predicate* MOD_m^r as follows:

$$w \models \text{MOD}_m^r(i) \text{ iff } i \bmod m = r$$

Transformers and Regular Languages in AC^0

Masked hard-attention transformers are equivalent to $FO[<, MOD]$, which is equivalent $LTL[MOD]$, which is equivalent to the regular languages in AC^0 .

How does adding more layers affect expressive power?

The *depth* of a masked hard-attention transformer is the number of attention layers it has. The *temporal depth* of an LTL formula is defined inductively as follows:

$$\text{dp}(Q_a) = 0$$

$$\text{dp}(\neg\phi) = \text{dp}(\phi)$$

$$\text{dp}(\phi \wedge \psi) = \max(\text{dp}(\phi), \text{dp}(\psi))$$

$$\text{dp}(\phi \textbf{ since } \psi) = \max(\text{dp}(\phi), \text{dp}(\psi)) + 1$$

Tripping up the Stairs

STAIR_k is the language over $\Sigma = \{a, b, c\}$ of strings which, after deleting c 's, contain a^k as a substring.

For instance, $\top$ **since** $(Q_a \wedge (\neg Q_b$ **since** $(Q_a \wedge (\neg Q_b$ **since** $Q_a))))$ recognizes STAIR_3 .

- $aa \notin \text{STAIR}_3$
- $aaa \in \text{STAIR}_3$
- $acaca \in \text{STAIR}_3$
- $aaaa \in \text{STAIR}_3$
- $acabaca \notin \text{STAIR}_3$

The since Hierarchy

Let LTL_k be the set of languages recognizable by LTL formulas of depth at most k . Then, Etessami and Wilke [2000] show:

$$\begin{aligned} STAIR_k &\in LTL_k \\ STAIR_{k+1} &\notin LTL_k \end{aligned}$$

$$\text{MUHAT}(\blacktriangleright F)_k \subsetneq \text{MUHAT}(\blacktriangleright F)_{2k+1}$$

$$\mathbf{MUHAT}(\blacktriangleright F)_k \subseteq \text{LTL}_{2k} \subsetneq \text{LTL}_{2k+1} \subseteq \mathbf{MUHAT}(\blacktriangleright F)_{2k+1}$$

Goal 3: COMPLETE

We now can describe how different components of masked hard-attention transformers contribute to their expressive power.

0. We **Reviewed** the definition of transformers
1. We **Explained** the difference between masked hard-attention transformers and soft attention transformers
2. We **Showed** the correspondence between masked hard-attention transformers and LTL
3. We **Described** how strict-masking, positional embeddings, and depth contribute to the expressivity of a masked hard-attention transformer

Recap

0. We **Reviewed** the definition of transformers **yes!**
1. We **Explained** the difference between masked hard-attention transformers and soft attention transformers - **lowering the temperature in softmax towards rhardmax**
2. We **Showed** the correspondence between masked hard-attention transformers and LTL - **via simulation in FO, simulation of LTL, and Kamp's Theorem**
3. We **Described** how strict-masking, positional embeddings, and depth contribute to the expressivity of a masked hard-attention transformer - **using interesting results from LTL!**

References i

- Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, 2020. doi:10.18653/v1/2020.emnlp-main.576.
- David Chiang and Peter Cholak. Overcoming a theoretical limitation of self-attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7654–7664, May 2022. doi:10.18653/v1/2022.acl-long.527. URL <https://aclanthology.org/2022.acl-long.527>.
- Javid Ebrahimi, Dhruv Gelda, and Wei Zhang. How can self-attention networks recognize Dyck-n languages? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4301–4306, November 2020. doi:10.18653/v1/2020.findings-emnlp.384.
- Kousha Etessami and Thomas Wilke. An until hierarchy and other applications of an ehrenfeucht–fraissé game for temporal logic. *Information and Computation*, 160(1-2):88–108, 2000. doi:10.1006/inco.1999.2846.
- Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, and Jonathan Berant. Memory-efficient transformers via top-k attention. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pages 39–52, 2021. doi:10.18653/v1/2021.sustainlp-1.5.
- Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020. doi:10.1162/tacl.a.00306. URL <https://aclanthology.org/2020.tacl-1.11>.
- Johan Anthony Willem Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968. URL <https://www.proquest.com/docview/302320357>.
- Jambay Kinley. *Two-Stream Transformer Architecture With Discrete Attention for Better Interpretability and Separation of Model Concerns*. Bachelor’s thesis, Harvard College, 2020. URL <https://dash.harvard.edu/handle/1/37364732>.

- William Merrill, Vivek Ramanujan, Yoav Goldberg, Roy Schwartz, and Noah A. Smith. Effects of parameter norm growth during transformer training: Inductive bias from gradient descent. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1766–1781, 2021. doi:10.18653/v1/2021.emnlp-main.133. URL <https://aclanthology.org/2021.emnlp-main.133>.
- Doron Peled and Thomas Wilke. Stutter-invariant temporal properties are expressible without the next-time operator. *Information Processing Letters*, 63(5):243–246, 1997. doi:10.1016/S0020-0190(97)00133-6.
- Hongfei Xu, Qiuhui Liu, Josef van Genabith, and Deyi Xiong. Learning hard retrieval decoder attention for Transformers. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 779–785, 2021. doi:10.18653/v1/2021.findings-emnlp.67.
- Andy Yang, David Chiang, and Dana Angluin. Masked hard-attention transformers and Boolean RASP recognize exactly the star-free languages, 2023. URL <https://arxiv.org/abs/2310.13897>. arXiv:2310.13897.



© 2024 by the authors (David Chiang, Jon Rawski, Lena Strobl, and Andy J Yang). All slides in this ESSLLI 2024 course series, including this one, are licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). You are free to share, adapt, and build upon this material for any purpose, even commercially, as long as you provide appropriate credit.