Transformer Expressivity and Formal Logic

David Chiang
Joint work with Dana Angluin (Yale), Peter Cholak, Anand Pillay, and Andy Yang (Notre Dame)

UNIVERSITY OF NOTRE DAME

Neural networks and formal languages

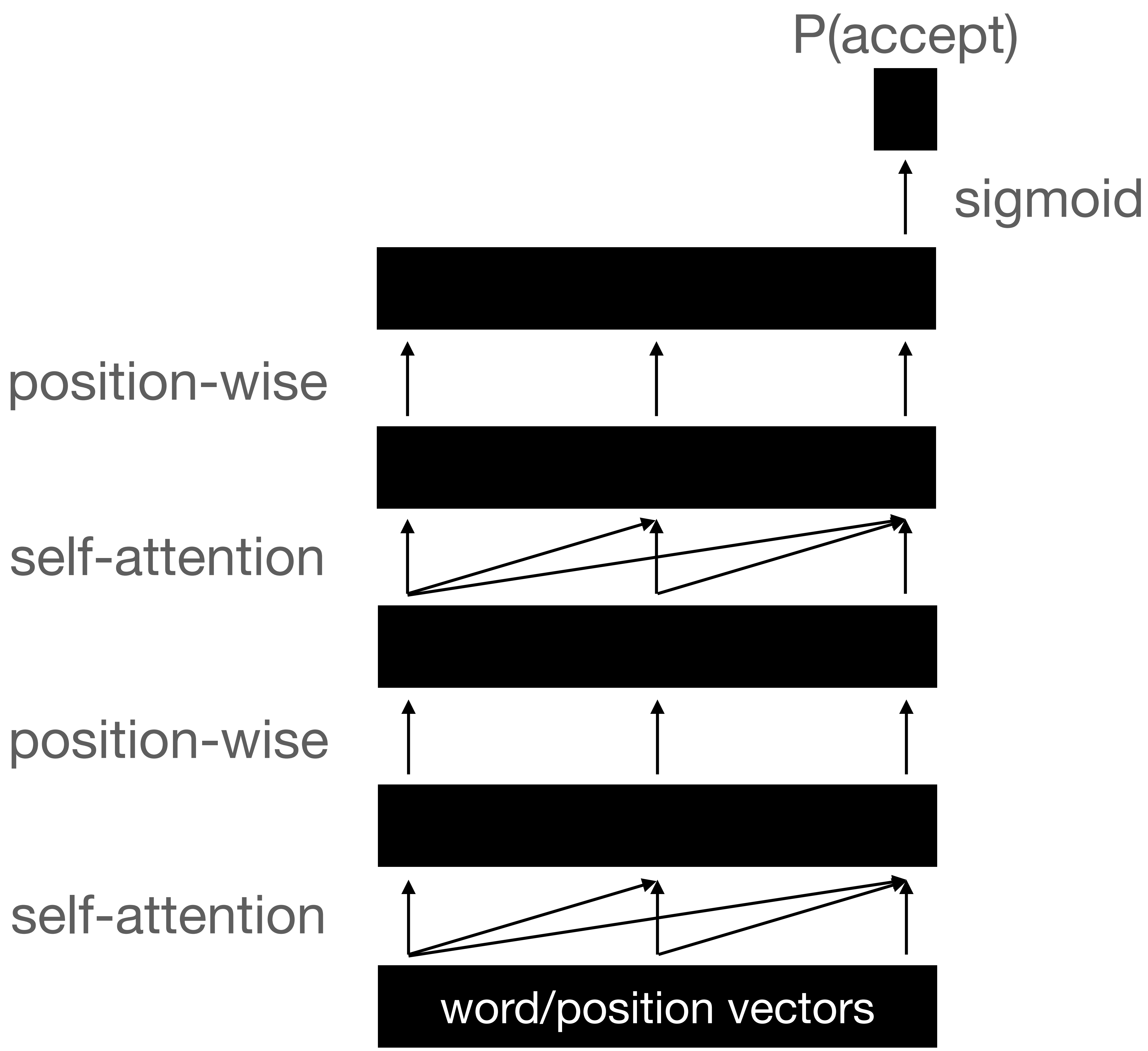
NN theory

expressivity | optimization | generalization

what formal languages can be described by transformers?

[1] Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? A survey. *Transactions of the Association for Computational Linguistics*, 2024.

Neural networks and formal languages



Neural networks and formal languages

$$(0 \land \neg 1) \lor (\neg 0 \land 1)$$

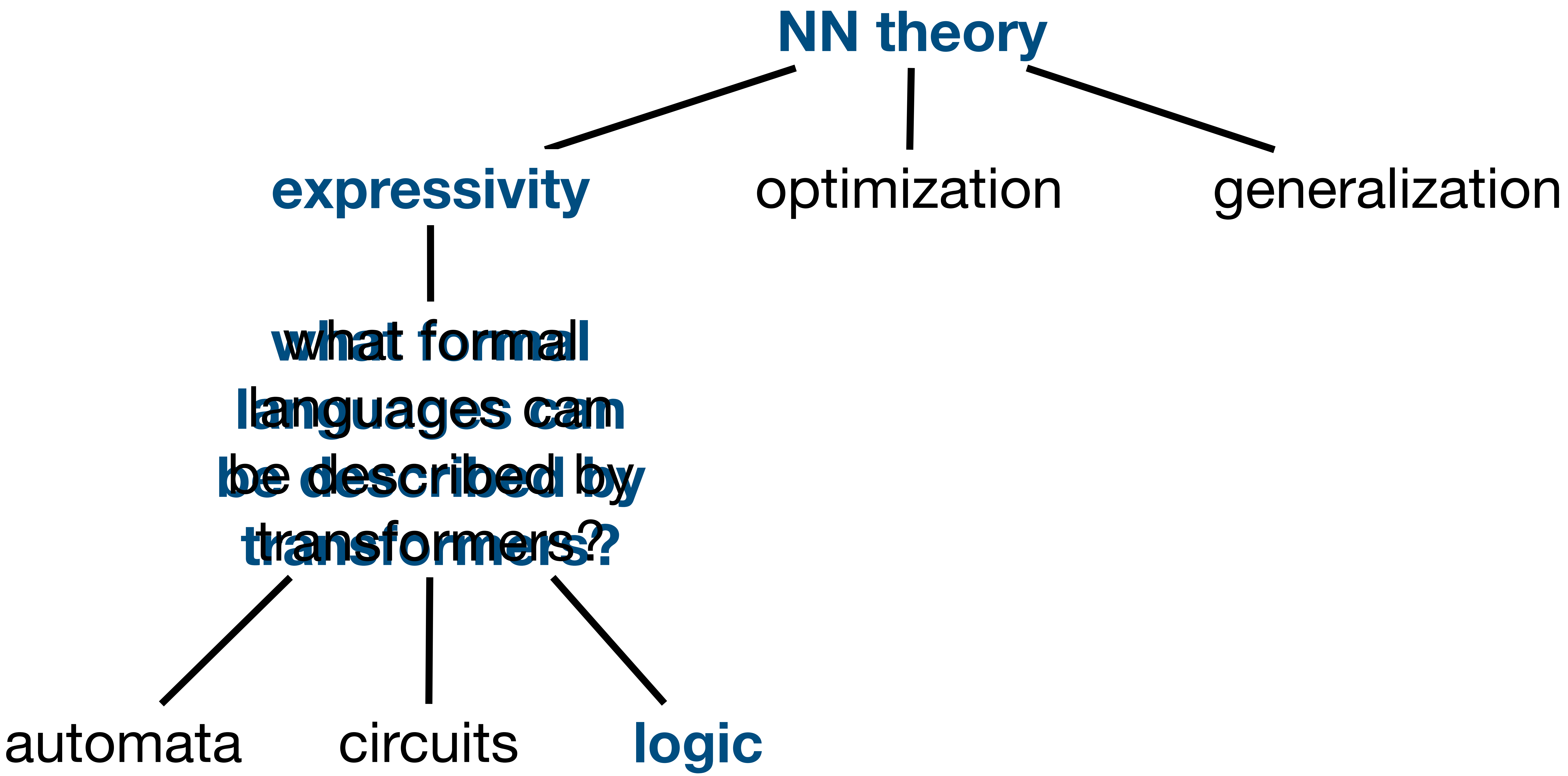
- Syntax: Testing whether a formula is well-formed (≈ 1-Dyck) *is* recognizable by a transformer
- Semantics: Testing whether a formula is true (= Boolean formula value problem or BFVP) is *not* recognizable by a transformer (assuming O(poly(n)) bits of precision and $TC^0 \neq NC^1$)

Neural networks and formal languages

- Weekly online seminar
- Organized by Lena Strobl (Umea) and Andy Yang (Notre Dame)

https://flann.super.site

Neural networks and formal languages



[1] Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? A survey. *Transactions of the Association for Computational Linguistics*, 2024.

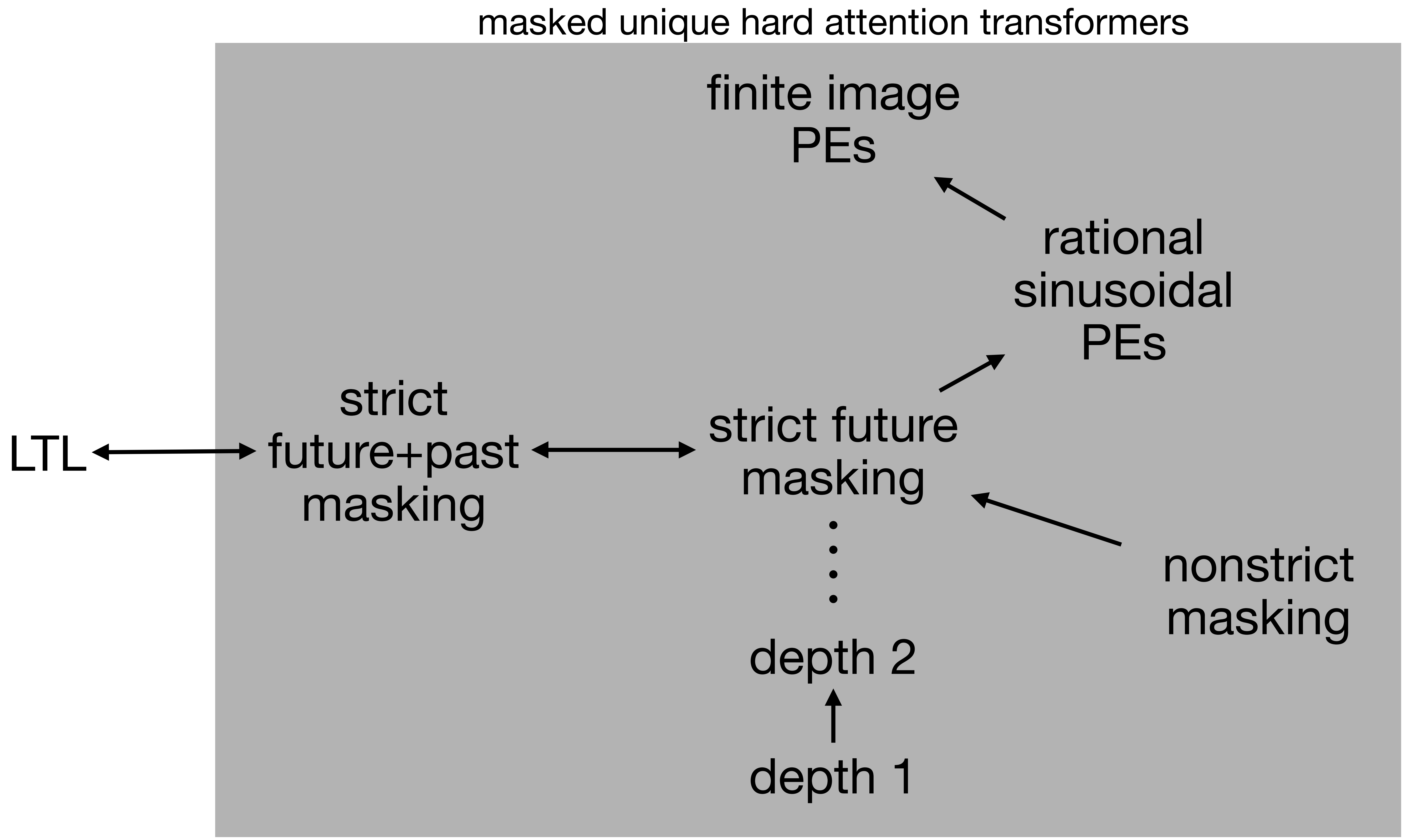
Neural networks and formal languages

- Transformers and logic (finite model theory, descriptive complexity theory)
  - Many connections with circuit complexity but don’t have to worry about uniformity
  - Fine control over computational resources (available predicates, quantifiers, number of variables, etc.)

Overview

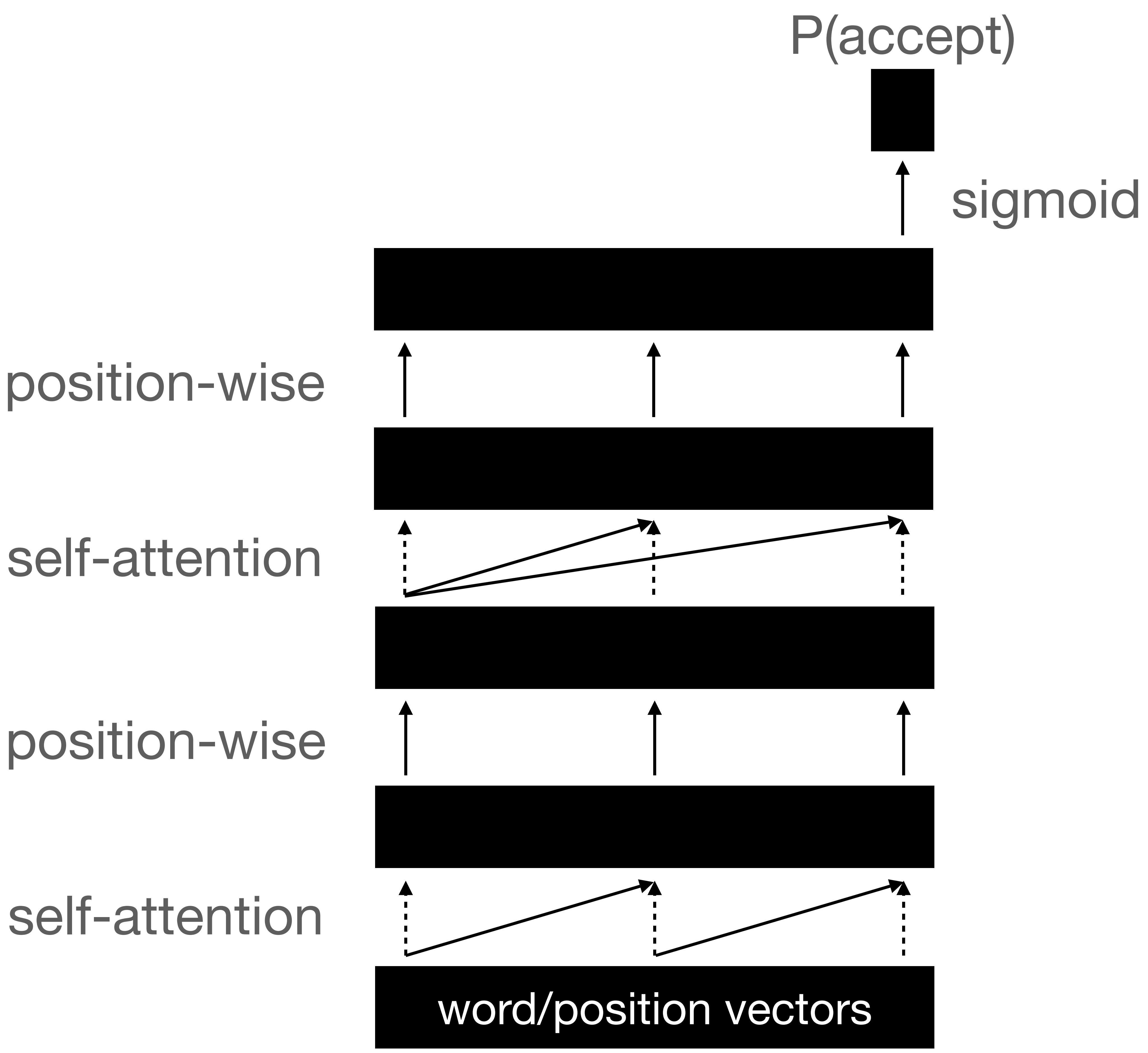
- First-order logic
  - Masked unique-hard attention transformers = first-order logic
  - Many related results, e.g., adding layers always adds expressivity
- Counting logics
  - Lower bounds: a temporal logic with counting ≤ transformers
  - Upper bounds: transformers ≤ a first-order logic with counting

First-order logic



[2] Andy Yang, David Chiang, and Dana Angluin. Masked hard-attention transformers recognize exactly the star-free languages. To appear, NeurIPS 2024.

Masked unique hard attention transformers (MUHATs)



- Unique hard attention
  - All attention is on the highest-scoring position
  - In case of a tie, choose rightmost
- Strict future masking
  - Each position can only attend to a previous (not same) position
  - First position attends nowhere

First-order logic (FO[<])

for strings

$$\forall x\,.\,Q_0(x)$$

"every position has a `0`"

`000000`: true

`010101`: false

`000111`: false

$$\forall x\,.\,\forall y\,.\,(Q_0(x) \wedge Q_1(y)) \rightarrow x < y$$

"every `0` comes before every 1"

`000000`: true

`010101`: false

`000111`: true

Linear temporal logic (LTL)

$$\mathbf{G}^{-1}\, Q_0$$

"it has always been `0`"

`000000`: true

`010101`: false

`000111`: false

$$(Q_1\ \mathbf{S}\ (\mathbf{G}^{-1}\, Q_0)) \wedge Q_1$$

"it was `1` since it had always been `0`, and it is now `1`"

`000000`: true

`010101`: false

`000111`: true

Linear temporal logic (LTL)

- $\phi \ \mathbf{S} \ \psi$ ("ϕ since ψ") is strict: ϕ doesn't have to be true currently
- Similarly, $\phi \ \mathbf{U} \ \psi$ ("ϕ until ψ")
- $\mathbf{G}^{-1} \ \phi$ ("always has been ϕ") can be defined in terms of $\mathbf{S}$

Star-free languages

(aa)*, PARITY, MAJORITY

$\not\Cap$

star-free regular expressions $\xleftrightarrow{\text{Schützenberger, 1965}}$ counter-free automata $\xleftrightarrow{\text{McNaughton and Papert, 1971}}$ FO[<] $\xleftrightarrow{\text{Kamp, 1968}}$ LTL

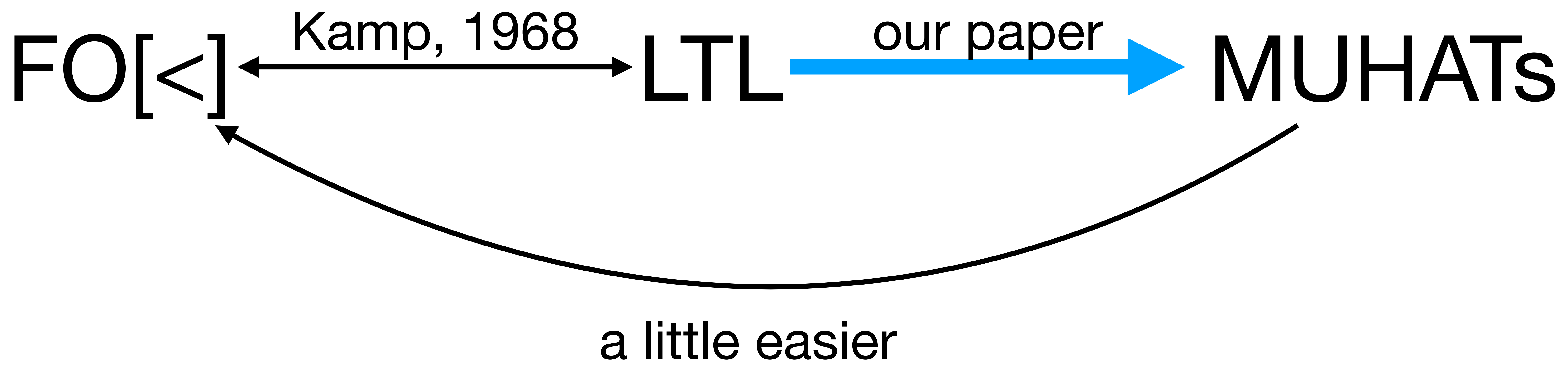
$\Cup$

(ab)*

Main result

$$
\begin{tikzcd}
\mathrm{FO}[<] \arrow[r, leftrightarrow, "\text{Kamp, 1968}"] & \mathrm{LTL} \arrow[r, leftrightarrow, "\text{our paper}"] & \mathrm{MUHATs} \arrow[ll, bend left, "\text{a little easier}"]
\end{tikzcd}
$$

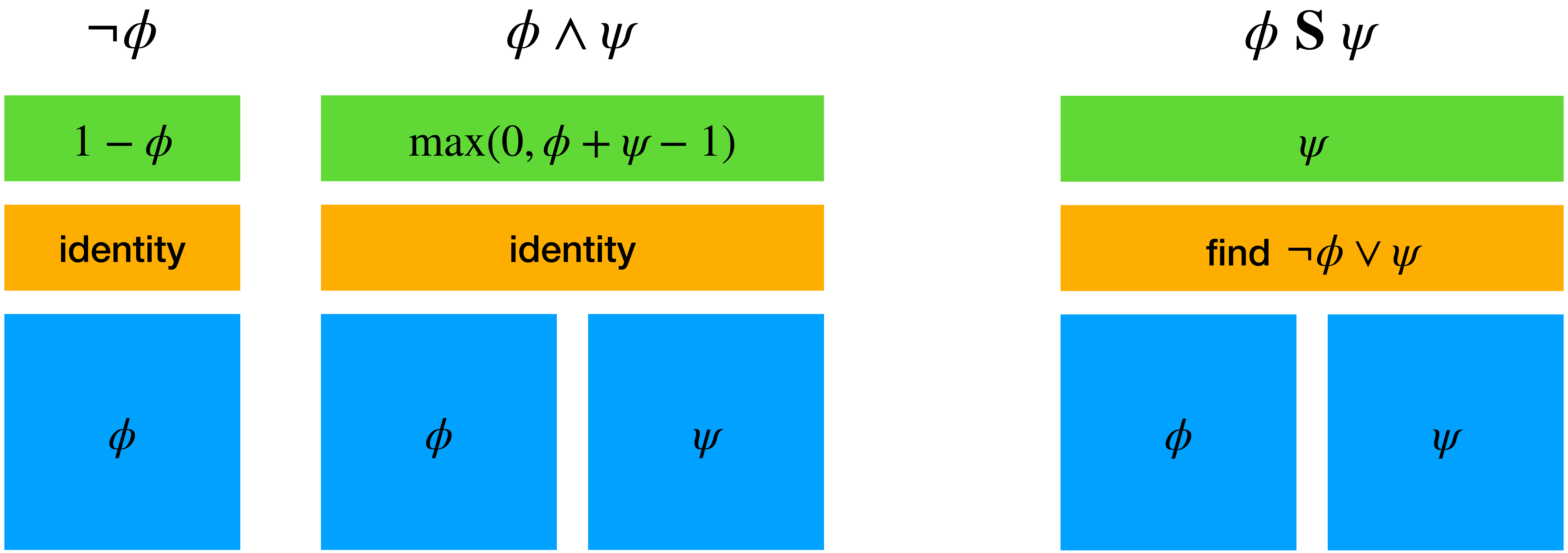
Main result



LTL to MUHATs

connectives become FFNs

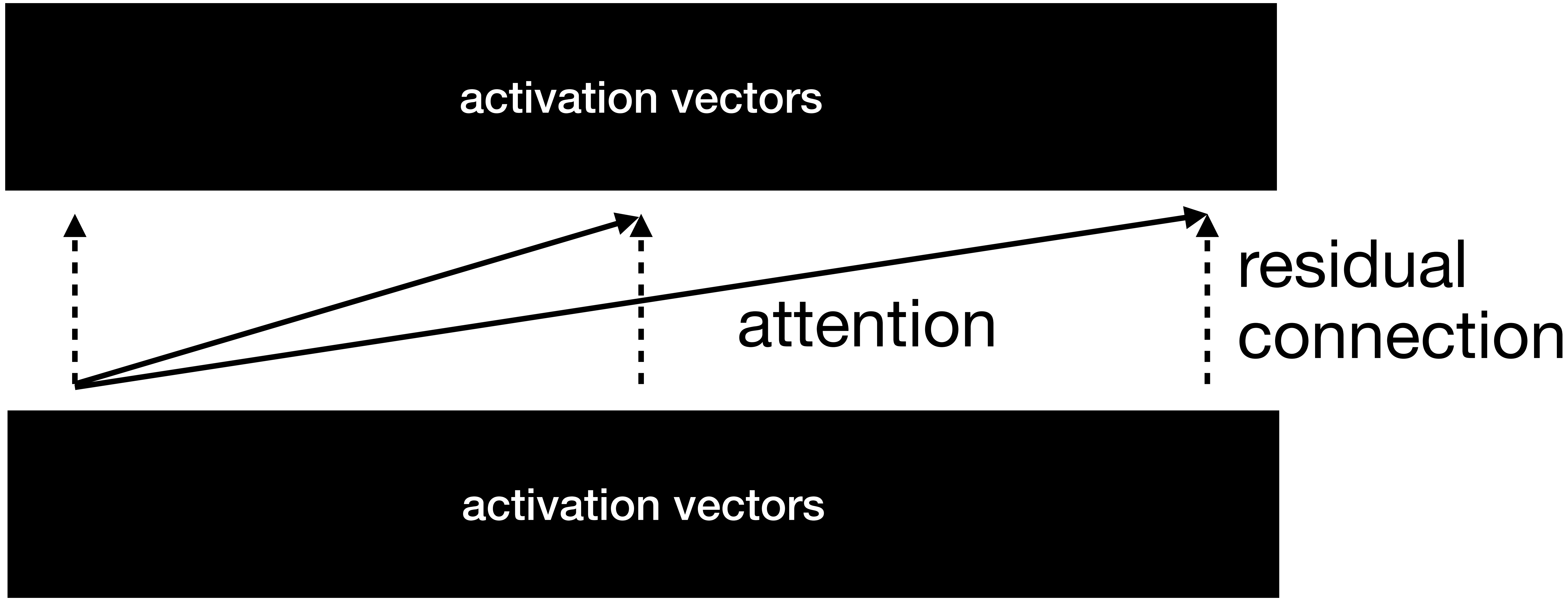
temporal operators use self-attention



Main result

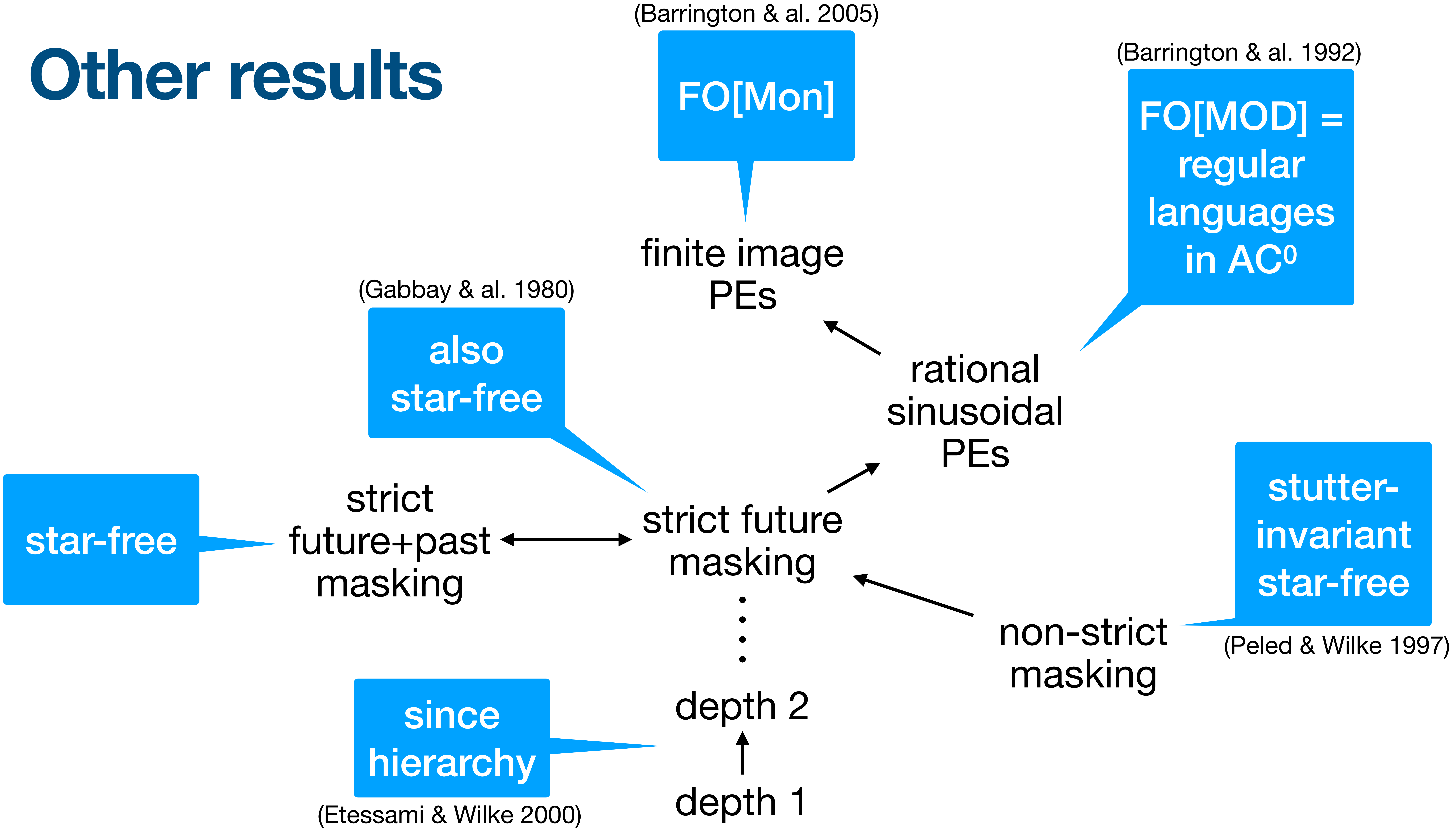
$$\begin{tikzcd} \mathrm{FO}[<] \arrow[r, leftrightarrow, "\text{Kamp, 1968}"] & \mathrm{LTL} \arrow[r, leftrightarrow, "\text{our paper}"] & \mathrm{MUHATs} \arrow[ll, bend left, "\text{a little easier}"] \end{tikzcd}$$

MUHATs to FO[<]



- Inputs are word embeddings
- Each position depends on at most two other positions
- There is a *finite* set of possible activation vectors

Other results



Counting

First-order logic with counting

$$\#x[Q_1(x)] > \#x[Q_0(x)]$$

"there are more 1s than 0s"

`000000`: false

`010100`: false

`101101`: true

$$\#x[Q_0(x)] = \#x[Q_1(x)] + \#x[Q_1(x)]$$

"there are twice as many 0s as 1s"

`000000`: false

`010100`: true

`101101`: false

Temporal logic with counting

$$\#Q_1 > \#Q_0$$

"there are more 1s than `0`s"

`000000`: false

`010100`: false

`101101`: true

$$\#Q_0 = \#Q_1 + \#Q_1$$

"there are twice as many `0`s as 1s"

`000000`: false

`010100`: true

`101101`: false

[3] Pablo Barcelo, Alexander Kozachinskiy, Anthony Widjaja Lin, and Vladimir Podolskii. Logical languages accepted by transformer encoders with hard attention. ICLR 2023.

Temporal logic with counting

1-DYCK: Matched and balanced parentheses

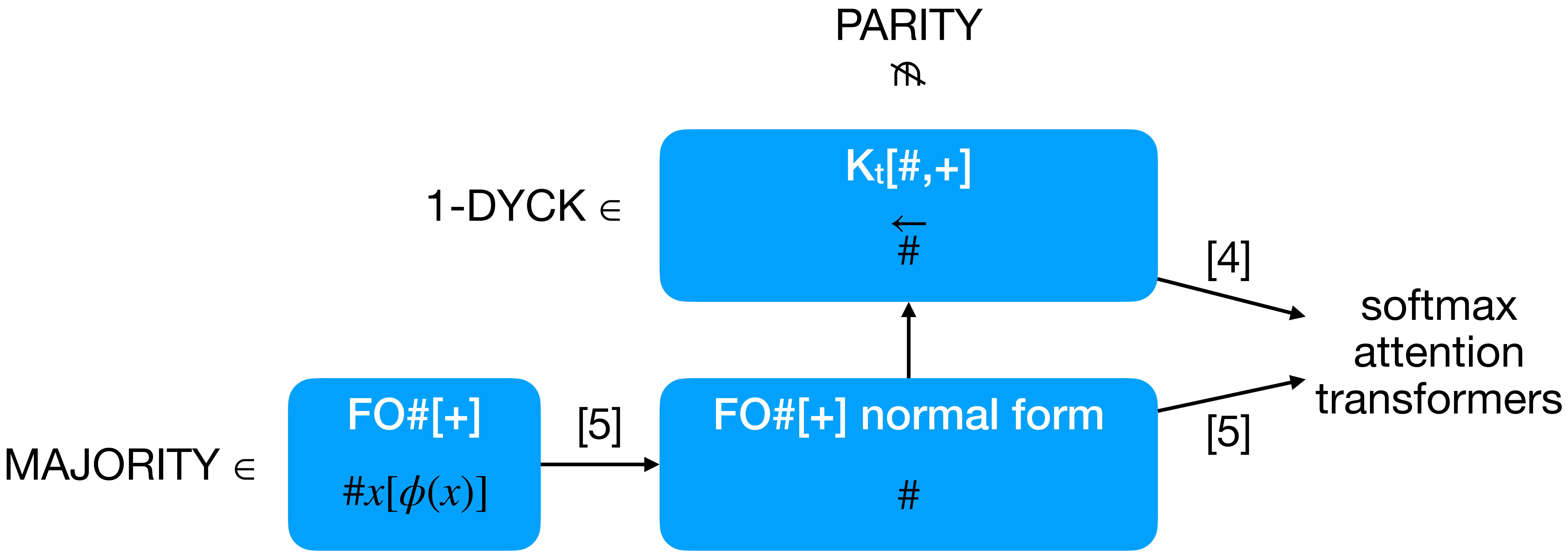
$$\overleftarrow{\#}\left[\overleftarrow{\#}Q_{(} < \overleftarrow{\#}Q_{)}\right] = 0 \ \wedge \ \overleftarrow{\#}Q_{(} = \overleftarrow{\#}Q_{)}$$

At no time have more left parentheses than right parentheses been seen

The number of left and right parentheses is equal

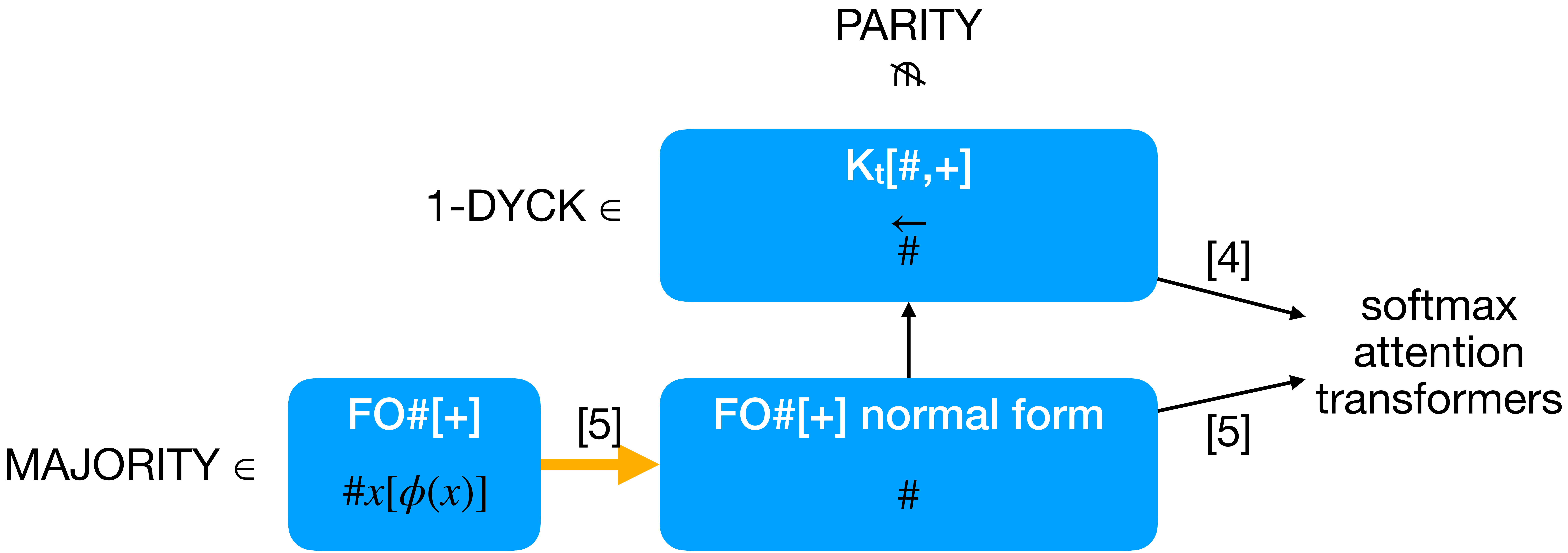
[4] Yang and Chiang. Counting like transformers: Compiling temporal counting logic into softmax transformers. CoLM 2024.

Lower bounds



[4] Yang and Chiang. Counting like transformers: Compiling temporal counting logic into softmax transformers. CoLM 2024.
[5] Chiang et al. Tighter bounds on the expressivity of transformer encoders. ICML 2023.

Lower bounds



[4] Yang and Chiang. Counting like transformers: Compiling temporal counting logic into softmax transformers. CoLM 2024.
[5] Chiang et al. Tighter bounds on the expressivity of transformer encoders. ICML 2023.

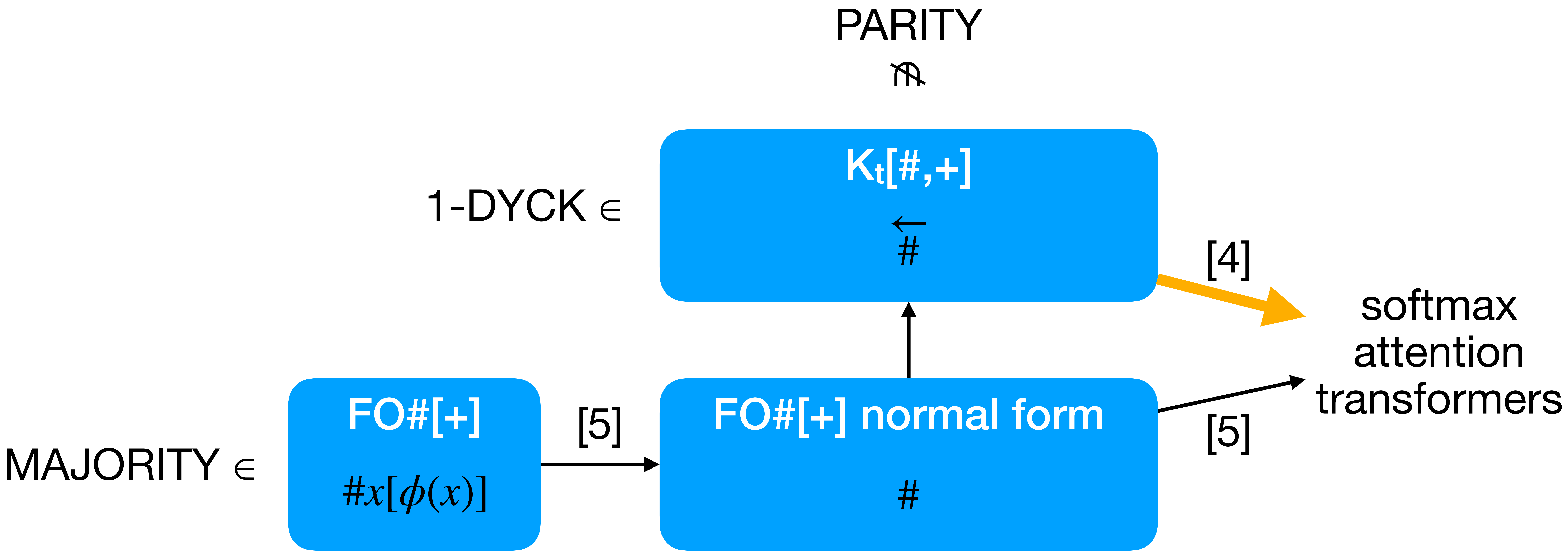
Lower bounds

One-variable normal form for FO#[+]

- FO#[+] only has unary (monadic) predicates ($Q_a(x)$; could add others)
- Adapt normal form for *monadic first-order logic* that has only one variable

$$
\begin{aligned}
&\#x[\#y[P(x) \wedge Q(y)]] \\
&\quad = \#x[P(x) \wedge \#y[Q(y)]] \\
&\quad = \#x[P(x)] \wedge \#y[Q(y)] \\
&\quad = \#x[P(x)] \wedge \#x[Q(x)]
\end{aligned}
$$

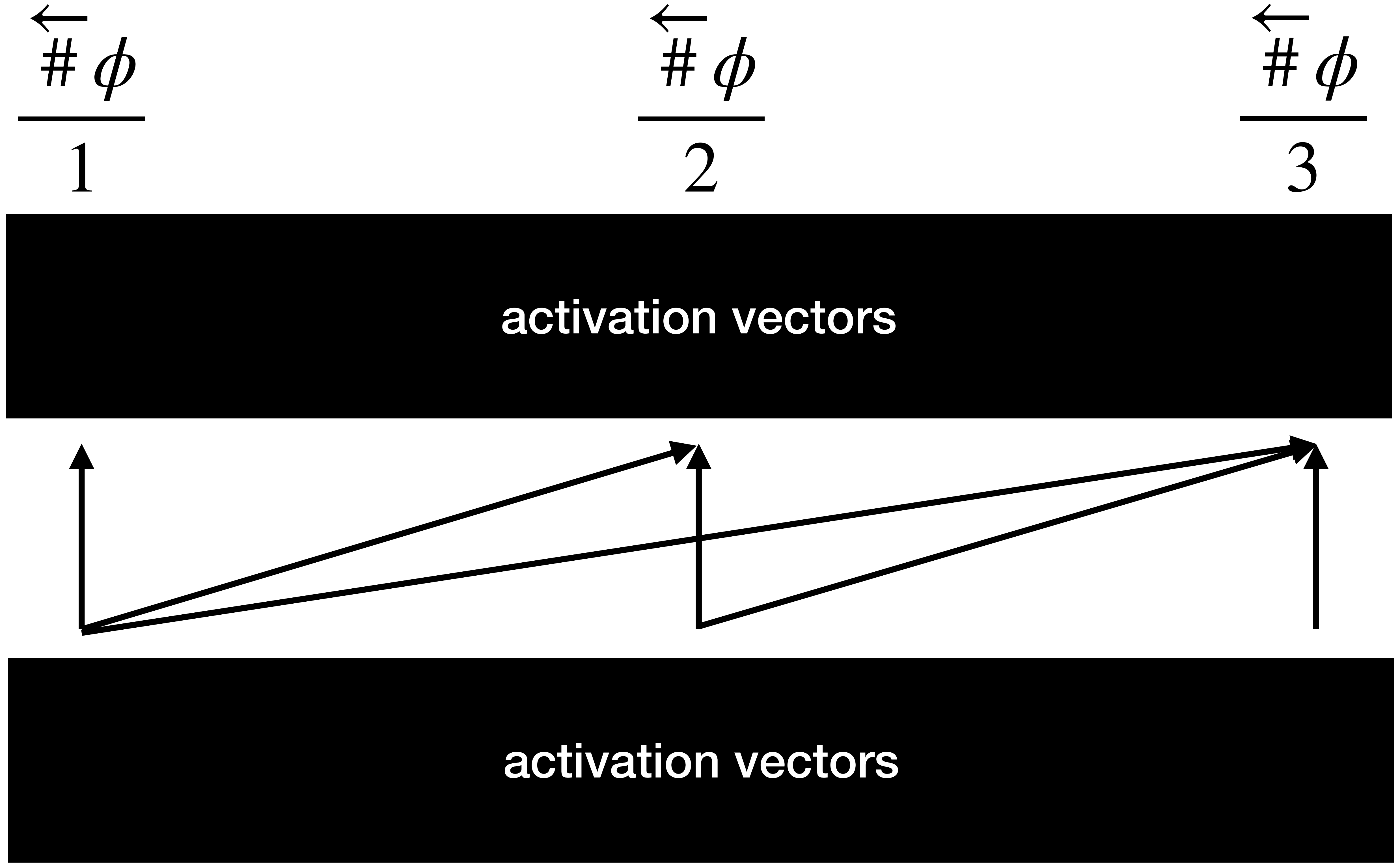
Lower bounds



[4] Yang and Chiang. Counting like transformers: Compiling temporal counting logic into softmax transformers. CoLM 2024.
[5] Chiang et al. Tighter bounds on the expressivity of transformer encoders. ICML 2023.

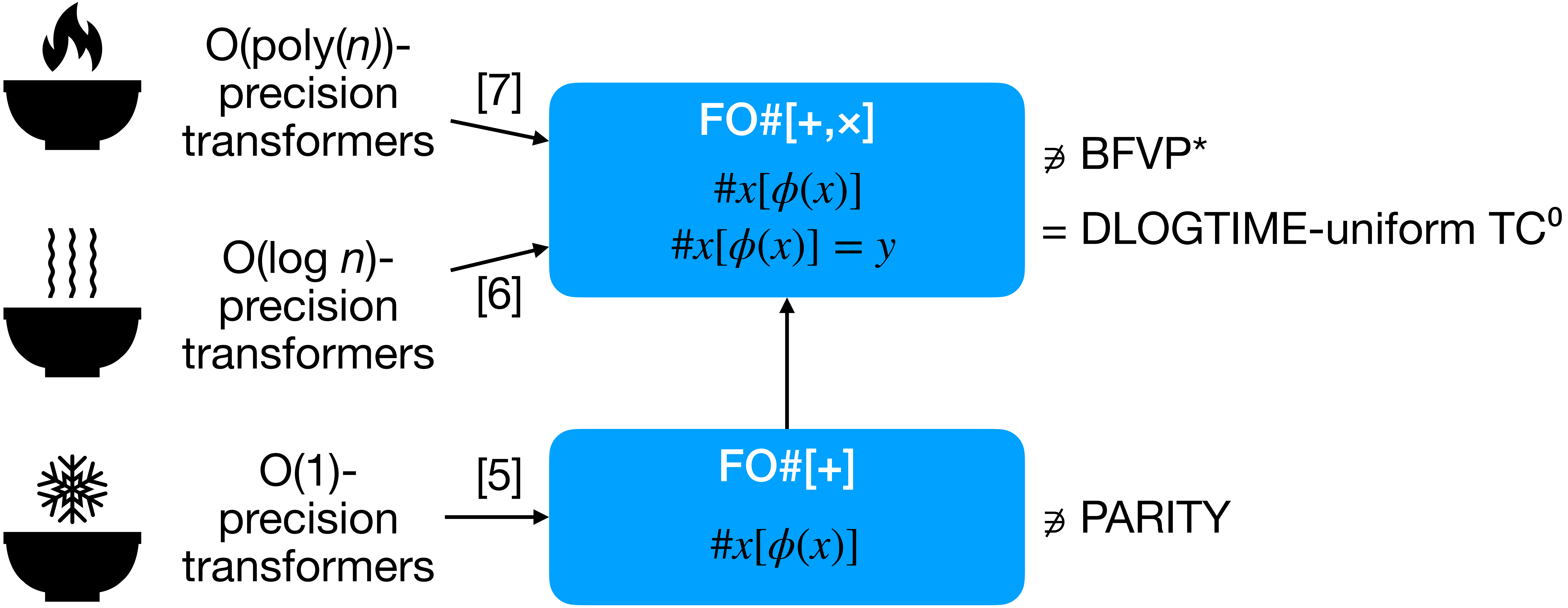
Lower bounds

The tricky part



- Attention doesn't count; it averages
- Often need to multiply by *n* (total length) or *i* (current position)
- Fancy position embeddings
- Layer normalization tricks

Upper bounds



[5] Chiang et al. Tighter bounds on the expressivity of transformer encoders. ICML 2023.
[6] Merrill and Sabharwal. A logic for expressing log-precision transformers. ICLR 2023.
[7] Chiang. Transformers in DLOGTIME-uniform TC^0. arXiv:2409.13629, 2024.

*Assuming $TC^0 \neq NC^1$.

Upper bounds

A restricted version of FO#[+,×]?

- FO#[+,×] is very expressive
  - a.k.a. FOM, FOM[BIT], DLOGTIME-uniform TC^0
  - see esp. Hesse et al., 2002
- Some possible restrictions:
  - Two variables
  - Separate sorts for counts and positions
  - Some restriction on multiplication

Wrap-Up

Conclusions

- Logic is a great tool for studying transformer expressivity
- We've worked out a full-orbed correspondence between *unique hard* attention transformers and FO[<]/LTL
- We're just getting started with *softmax* attention transformers and counting logics

$\wedge$ $\vee$ $\neg$

$\#$ $\overleftarrow{\#}$ $\overrightarrow{\#}$

S U

$\forall$ $\exists$