ICLR NEURAL INFORMATION PROCESSING SYSTEMS

Carnegie Mellon University Penn University of Pennsylvania Microsoft

Thinking Fast with Transformers

Algorithmic Reasoning with Shortcuts

Bingbin Liu
CMU

Jordan T. Ash
MSR NYC

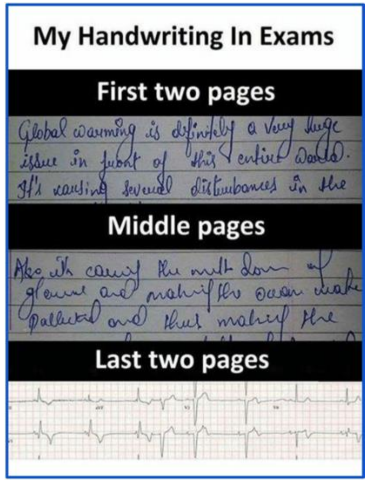
Surbhi Goel
UPenn

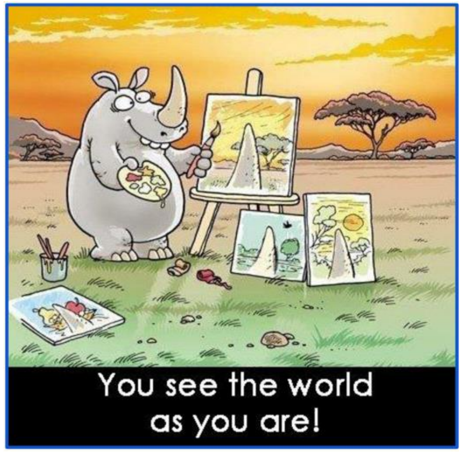
Akshay Krishnamurthy
MSR NYC

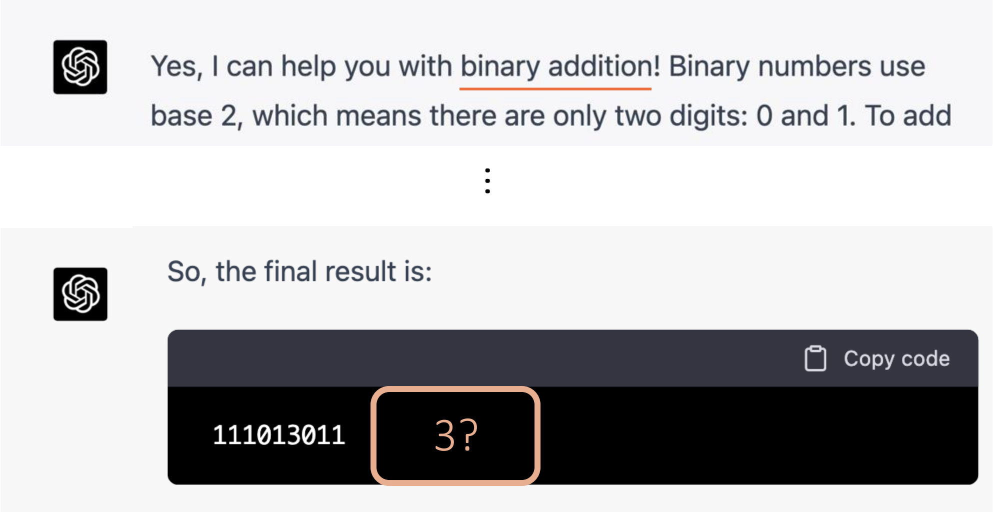
Cyril Zhang
MSR NYC

Reasoning in language models

Capabilities and limitations?







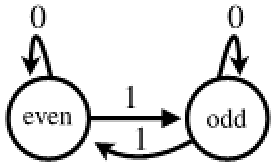
Powerful and practically useful.

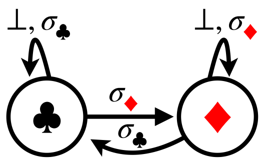
Unreliable with incorrect answers.

Agenda

Q: How can parallel models such as Transformers model sequential reasoning?

Formalizing with automata:





- Capabilities in theory (representational)
- Solutions found in practice (optimization, generalization)

TL;DR: Transformers reason with shallow solutions, with computational advantages but statistical issues.

Formalizing reasoning

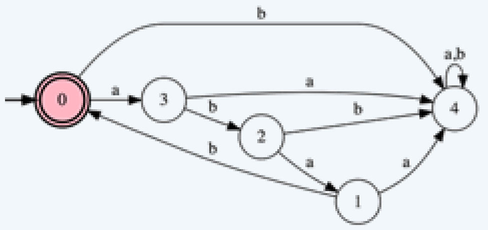
Parse, Infer,

Retrieve, Search

……

→ execute symbolic computations* →

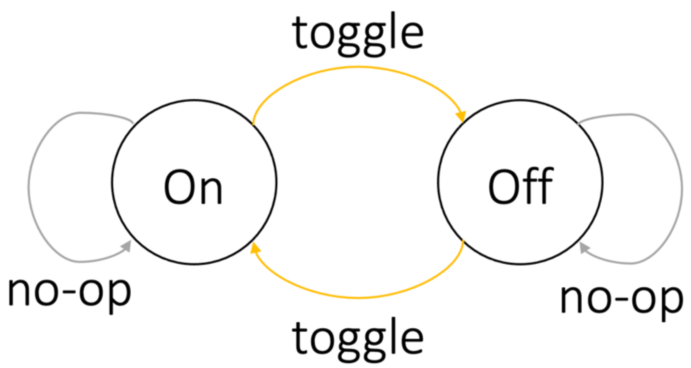
Transitions of discrete state machines



parity

```
An on-off switch is off.
(actions: toggle or not)
Now the switch is ?.
```

[Han 20, Anil et al. 22]



Formalizing reasoning

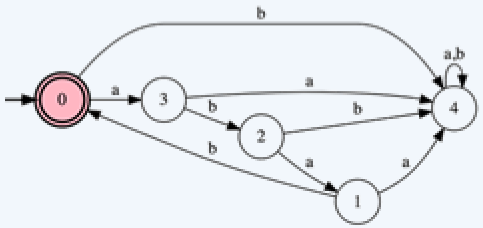
Parse, Infer,

Retrieve, Search

......

→ execute symbolic computations* →

Transitions of **finite state** machines



An on-off switch is off.

(actions: toggle or not)

Now the switch is ?.

Parity

[Han 20, Anil et al. 22]

```
  00010011
+ 10100110
----------
  10111001
```

Addition

[Nogueira et al. 21]

example@cmu.edu

@([a-zA-Z0-9_.+-]+)\.[a-zA-Z0-9_.+-]

Regular expressions

[Bhattamishra et al. 20]

```
<div>
    <div>
        <div>
            <ul>
                <li></li>
                <li></li>
                <li></li>
                <li></li>
            </ul>
        </div>
    </div>
</div>
```

Bounded nested brackets

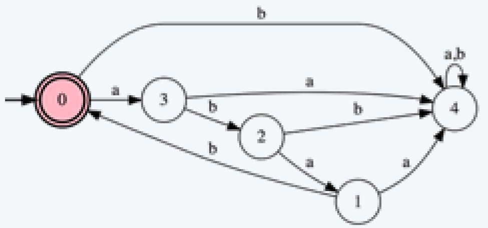
[Yao et al. 21]

Formalizing reasoning

Wide ranges of reasoning tasks

finite-state automata ↔ regular languages

Transitions of
finite state machines




```
An on-off switch is off.
(actions: toggle or not)
Now the switch is ?.
```

Parity
[Han 20, Anil et al. 22]

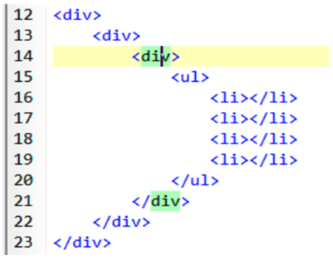
```
  00010011
+ 10100110
  --------
  10111001
```

Addition
[Nogueira et al. 21]

example@cmu.edu

@([a-zA-Z0-9_.+-]+)\.[a-zA-Z0-9_.+-]

Regular expressions
[Bhattamishra et al. 20]



Bounded nested brackets
[Yao et al. 21]

Formalizing reasoning with automata

$\mathcal{A} = (Q, \Sigma, \delta)$

states **inputs** **transitions**

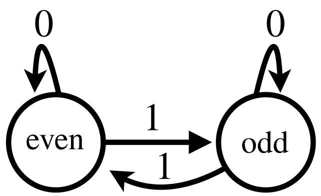
$q_t = \delta(q_{t-1}, \sigma_t)$

(Q is finite)

parity counter

$Q = \{\text{even}, \text{odd}\}$

$\Sigma = \{0, 1\}$

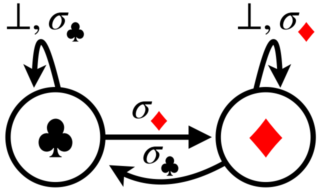


1-bit memory unit

$Q = \{\clubsuit, \diamondsuit\}$

$\Sigma = \{\sigma_\clubsuit, \sigma_\diamondsuit, \bot\}$

(no-op)



(will reappear later)

Task: modeling the dynamics of $\mathcal{A}$.

Task: Simulating automata

$\mathcal{A} = (Q, \Sigma, \delta)$

states, inputs, transitions

Simulating $\mathcal{A}$: learn a *seq2seq function* for sequence length T.

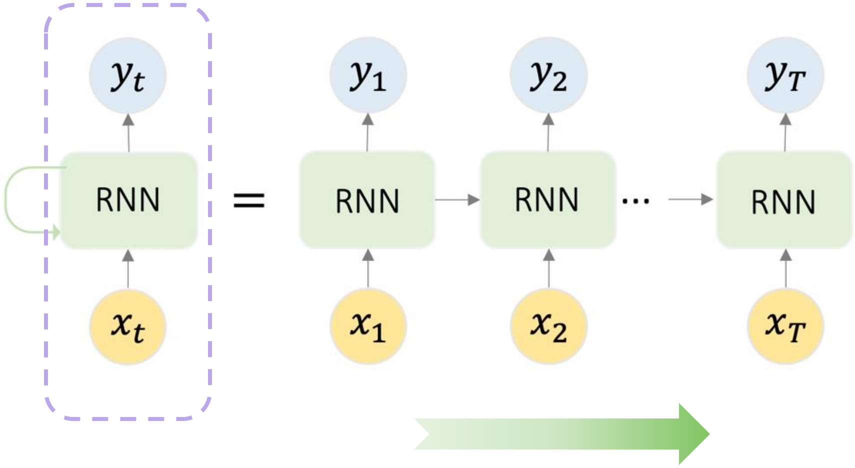
- Input = $\sigma_1, \sigma_2, \cdots, \sigma_T \in \Sigma$ (alphabet), output = $q_1, q_2, \cdots, q_T \in Q$ (states).

Architecture choices

RNN

sequential across positions

Natural for $q_t = \delta(q_{t-1}, \sigma_t)$

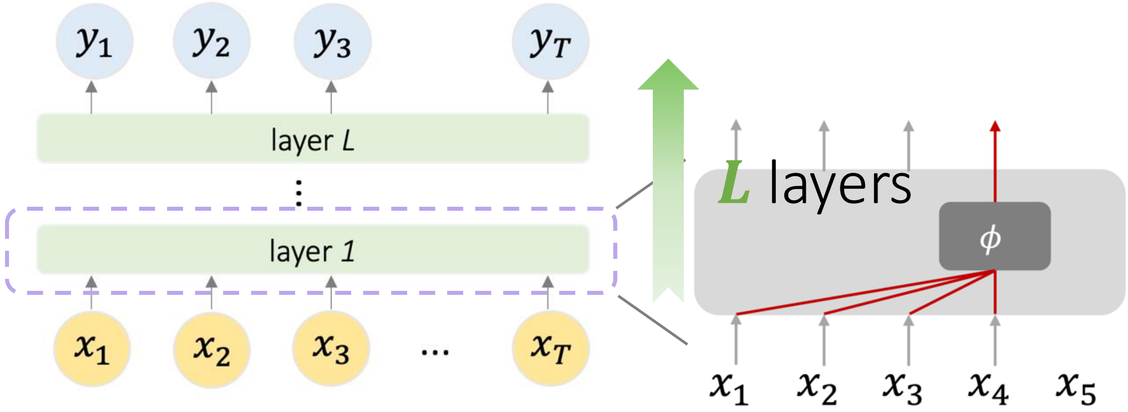


T positions

Transformer

parallel across positions

sequential across layers



Typically $L \ll T$.

Task: Simulating automata

$\mathcal{A} = (Q, \Sigma, \delta)$

states, inputs, transitions

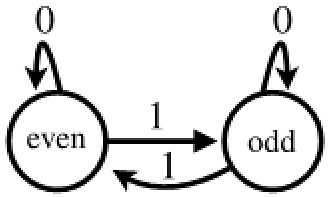
Simulating $\mathcal{A}$: learn a *seq2seq function* for sequence length T.

- Input = $\sigma_1, \sigma_2, \cdots, \sigma_T \in \Sigma$ (alphabet), output = $q_1, q_2, \cdots, q_T \in Q$ (states).

Note: more than 1 way to simulate $\mathcal{A}$.

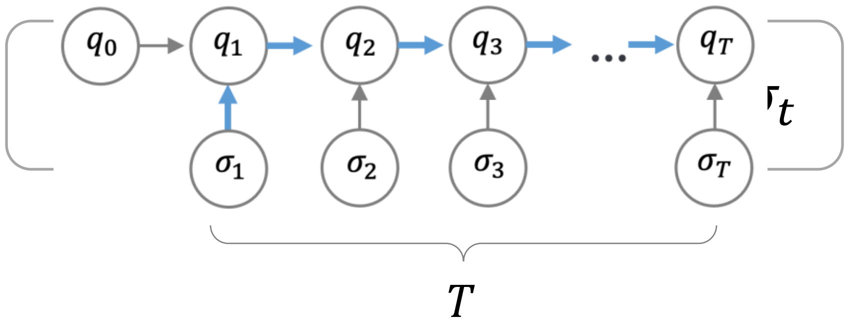
Shortcut

$o(T)$ *# sequential steps*



parity counter

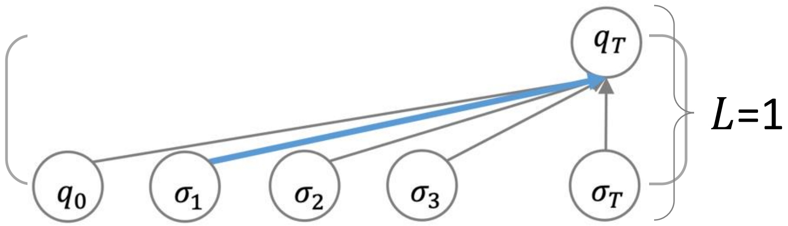
Iterative solution



"RNN solutions"

❌ Not shortcut

Parallel solution



"Transformer solutions"

✅ Shortcut

Transformers learn shortcut to automata

Q: How parallel models such as Transformers perform sequential reasoning?

Theoretically: shortcut solutions

- How short can the shortcuts be?
  - Measured by network depth.
- What structure/properties are needed?
  - Tools: group theory, Krohn-Rhodes.

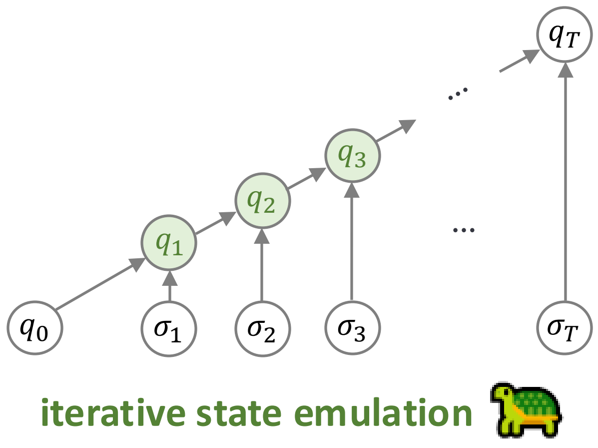
Empirically:

- Can shortcuts be found?
  - Is theory predictive?
- What are the empirical solutions?
  - Same as the constructions?
  - Properties?

Solutions of Reasoning

$$\mathcal{A} = (Q, \Sigma, \delta),$$
$$q_t = \delta(q_{t-1}, \sigma_t).$$

# steps = T definition of δ	# steps = $O(\log T)$	# steps = $O_{|Q|}(1)$



(*shortcuts*)

represented by RNNs

represented by Transformers

$O(\log T)$ steps

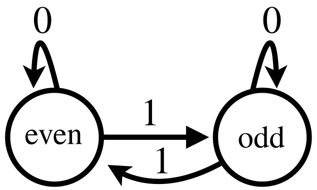
$$\mathcal{A} = (Q, \Sigma, \delta),$$
$$q_t = \delta(q_{t-1}, \sigma_t).$$

Goal: compute $q_t = \left(\delta(\cdot, \sigma_t) \circ \cdots \circ \delta(\cdot, \sigma_1)\right)(q_0),\ t \in [T].$

$\delta(\cdot, \sigma)\colon Q \to Q$

function ⟷ matrix

composition ⟷ multiplication



$Q = \{\text{even}, \text{odd}\}$
$\Sigma = \{0, 1\}$

parity counter

$$\delta(\cdot, 0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$\delta(\cdot, 1) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$q_t = \left(\delta(\cdot, \sigma_t) \circ \cdots \quad \circ\, \delta(\cdot, \sigma_1)\right) q_0$$

$$e_{q_t} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} e_{q_0}$$

$O(\log T)$ steps

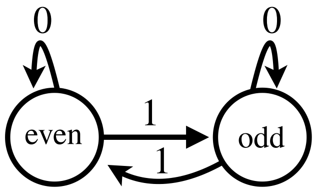
$$\mathcal{A} = (Q, \Sigma, \delta),$$
$$q_t = \delta(q_{t-1}, \sigma_t).$$

Goal: compute $q_t = \big(\delta(\cdot, \sigma_t) \circ \cdots \circ \delta(\cdot, \sigma_1)\big)(q_0),\ t \in [T].$

$\delta(\cdot, \sigma)\colon Q \to Q$

function ⟷ matrix

composition ⟷ multiplication



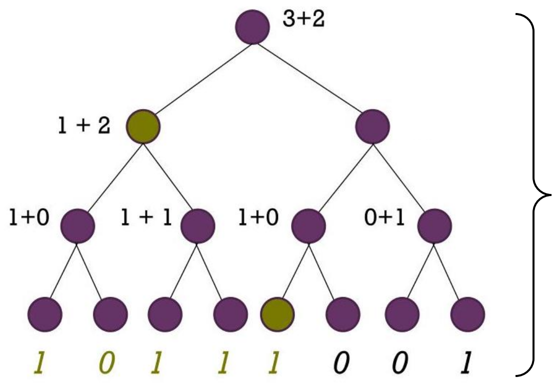
$Q = \{\text{even}, \text{odd}\}$
$\Sigma = \{0, 1\}$

parity counter

$$\delta(\cdot, 0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$\delta(\cdot, 1) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

associativity



$O(\log T)$

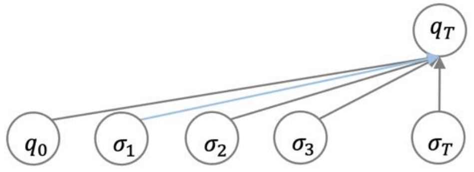
Can we use $o(\log T)$ layers?

$q_t = (\delta(\cdot, \sigma_t) \circ \cdots \circ \delta(\cdot, \sigma_1))\ (q_0)$

We already have positive results.

- Parity: only need to count #1s.

$q_t = (\sum_{\tau \le t} \sigma_\tau) \bmod 2$



$f \circ g = g \circ f$

Counting works for commutative function composition: $O(1)$ layers.

$f \circ g \neq g \circ f$

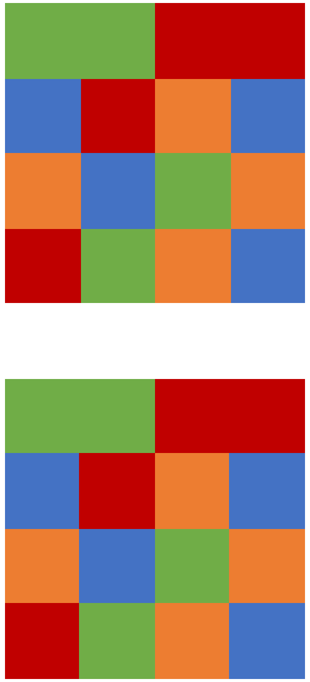
How about *non-commutative* compositions?

Decomposition

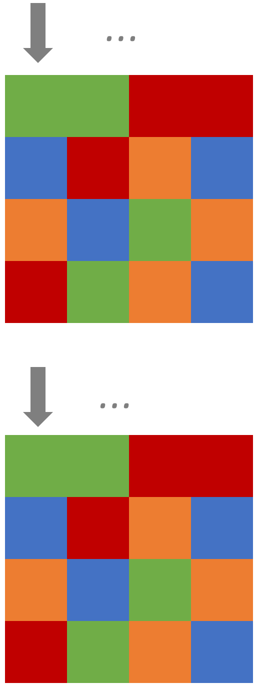
$\tilde{O}(|Q|^2)$ steps: decomposition

Aside: "Shortcut" in recognizing visual patterns [Huang and Pashler 07]

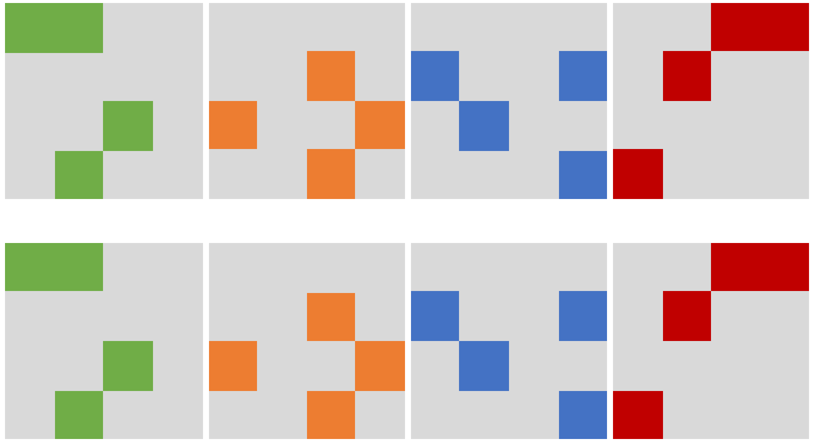
Matching?



1. tracing the cells one by one 🐢

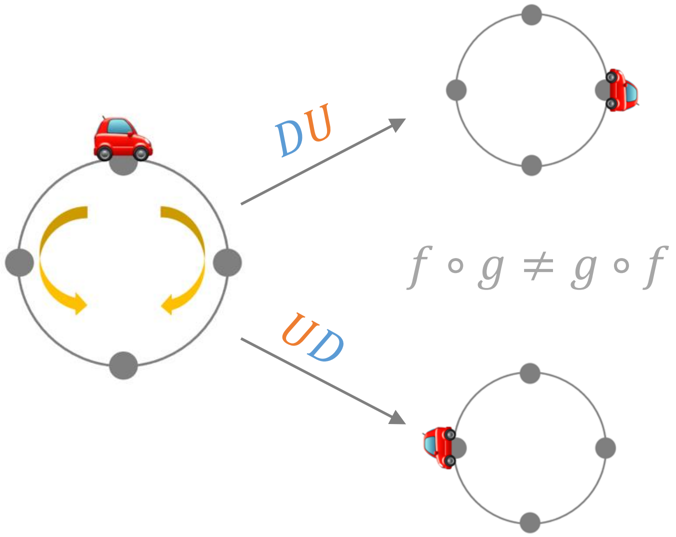


2. *shortcut* via *decomposition* (color)
(fewer sequential steps 🦅)



Decomposition: car on a circle

$Q = \{$🚗, 🚘$\} \times \{0,1,2,3\}, \Sigma = \{D$(drive)$, U$(U–turn)$)$



$q_0 = ($🚗$, 0), \ \sigma_{1:T} = DDDUDDUUD \rightarrow q_T$?

- Direction = parity (sum) of U. (parity: {1, -1} ↔ {0, 1})
- Position = signed sum mod 4 : sign = parity of U.

$O(1)$ layer each

$D \ D \ D \ U \ D \ D \ U \ U \ D$

Parity: 1 1 1 -1 -1 -1 1 -1 -1 → 🚘

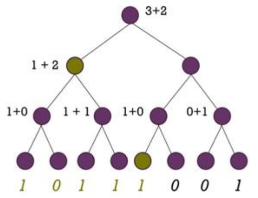
Signed sum: 1 1 1 0 -1 -1 0 0 -1 → 0

Decomposition: general

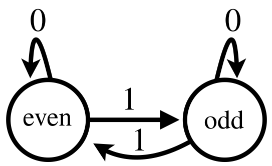
What are we decomposing?

Transformation group: $\mathcal{T}(\mathcal{A}) \coloneqq \{\delta(\cdot, \sigma) : \sigma \in \Sigma\}$ under composition.

Recall: group axioms



- Associativity: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
- Inverse: $a \cdot b = b \cdot a = e$ (identity)

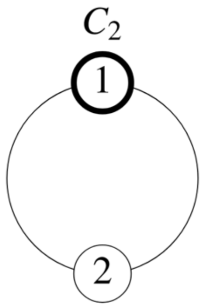


$Q = \{\text{even}, \text{odd}\}$

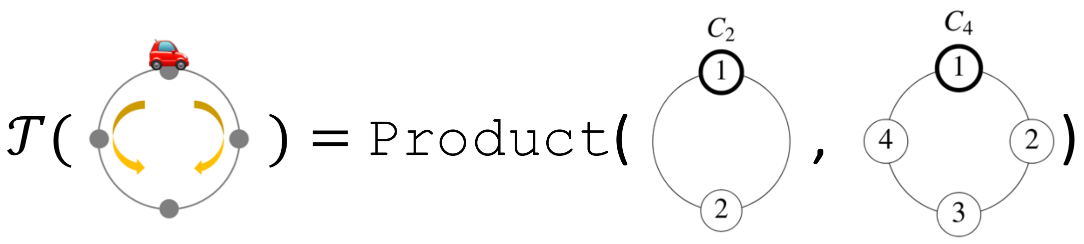
$\Sigma = \{0, 1\}$

parity counter (mod 2)

$\mathcal{T}(\mathcal{A})$



cyclic group C_2



Decomposition: general

Group: associative + invertible

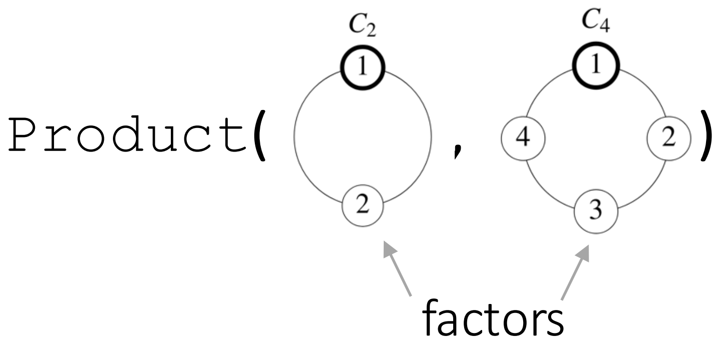
Transformation group: $\mathcal{T}(\mathcal{A}) := \{\delta(\cdot, \sigma) : \sigma \in \Sigma\}$ under composition.

"Prime factorization" for groups:

$G = H_n \rhd H_{n-1} \cdots \rhd H_1$ *(Jordan & Hölder)* [$N = p_n \cdot p_{n-1} \cdots \cdot p_1$ (Euclid)]

- H_{i-1} is a "factor" (normal subgroup) of H_i.
  $\rightarrow n = O(\log |G|)$
- H_{i+1}/H_i are "prime numbers" (simple groups).

If commutative (abelian): 1 layer



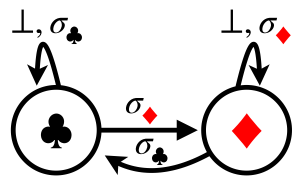
"Solvable G" *… with* $O(\log|G|)$ layers ***…What is*** $|G|$***?***

Decomposition: general

Group: associative + invertible

Transformation group: $\mathcal{T}(\mathcal{A}) \coloneqq \{\delta(\cdot, \sigma) : \sigma \in \Sigma\}$ under composition.

Invertible: $a \cdot b = b \cdot a = e$ (identity)



$Q = \{♣, ♦\}$
$\Sigma = \{\sigma_♣, \sigma_♦, \bot\}$

1-bit memory unit

(aka. flipflop)

$$\delta(\cdot, \sigma_♣) = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$

singular → no inverse

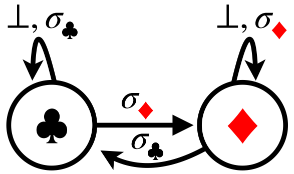
Decomposition: general

Semigroup: associative (+ identity)

Transformation semigroup: $\mathcal{T}(\mathcal{A}) \coloneqq \{\delta(\cdot,\sigma) : \sigma \in \Sigma\}$ under composition.

Invertible: $a \cdot b = b \cdot a = e$ (identity)

Semigroup: associativity only



$Q = \{♣, ♦\}$

$\Sigma = \{\sigma_♣, \sigma_♦, \bot\}$

1-bit memory unit

(aka. flipflop)

$$\delta(\cdot,\ \sigma_♣) = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$

singular → no inverse

$|\mathcal{T}(\mathcal{A})| \leq O(|Q|^{|Q|})$

i.e. with $O(|Q| \log |Q|)$ layers

$\tilde{O}(|Q|)$

Jordan & Hölder ($\mathcal{T}(\mathcal{A})$: group)

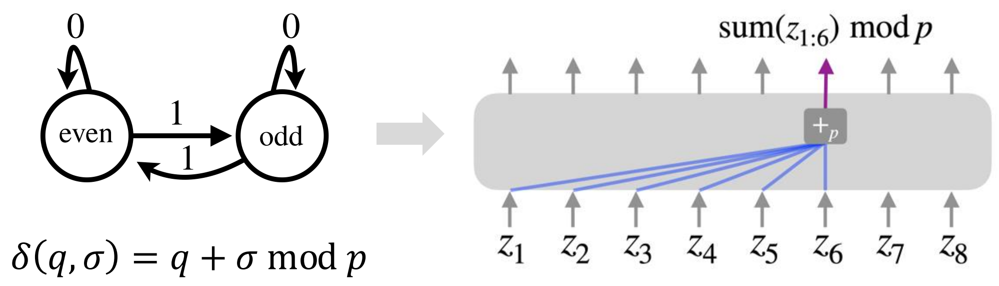
↓

Krohn-Rhodes ($\mathcal{T}(\mathcal{A})$: semigroup)

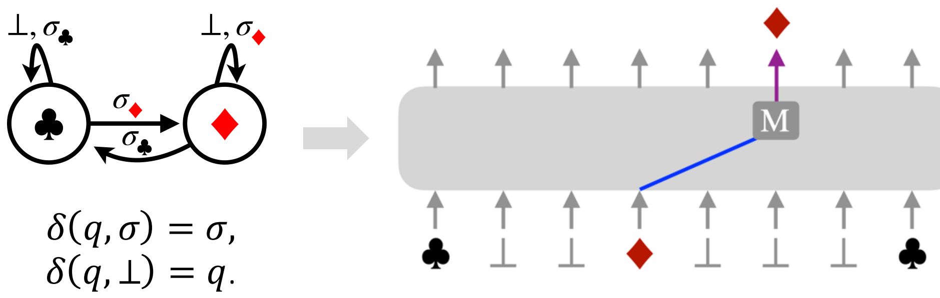
$\tilde{O}(|Q|^2)$ steps: decomposition

constrain the type of "factors"   #factors $\leq \mathrm{poly}(|Q|)$

Krohn-Rhodes: solvable $\mathcal{A}$ decomposes into 2 types of factors.



mod counter   uniform attention



memory unit   sparse attention

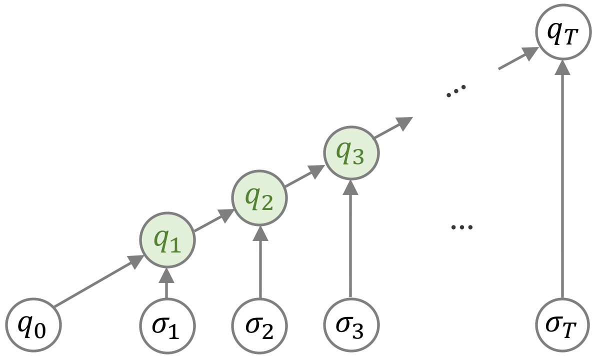
Each representable by 1 Transformer layer

+ "gluing" with O(1) layers (using MLP).

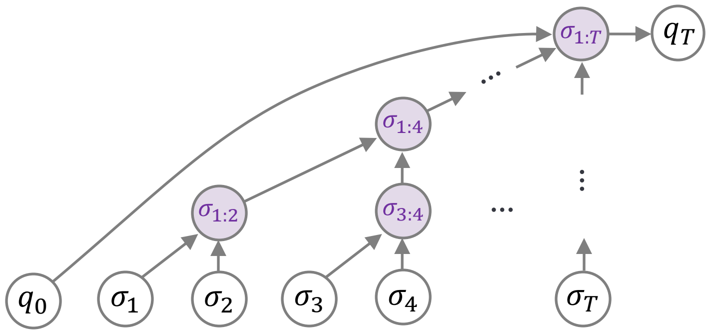
Solutions of Reasoning

$q_t = \left(\delta(\cdot, \sigma_t) \circ \cdots \circ \delta(\cdot, \sigma_1)\right)(q_0)$

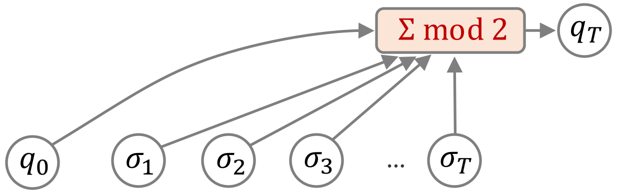
# steps = T definition of δ	# steps = $O(\log T)$ associativity	# steps = $\tilde{O}(\lvert Q \rvert^2)$ algebraic structure



iterative state emulation



multi-scale function composition



Krohn-Rhodes decomposition

represented by RNNs

represented by Transformers

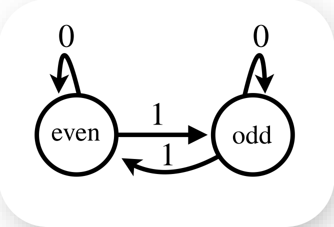
for *all* $\mathcal{A}$

for *solvable* $\mathcal{A}$

Simulating $\mathcal{A}$ in practice

19 automata

$\sigma_{1..T}$
01101100



$q_{1..T}$
01001000

Transformer
with standard training

Can shortcuts be found?

Can shortcuts be learned?

Yes, across 19 automata & 16 depths.

- Shortcuts are found.
- Deeper factorization → more layers.
- Open challenges:
  - Stabilize training?
  - Interpret the solutions?

→ example: Gridworld

lighter > darker

Transformer depth L (T=100)

Gridworld

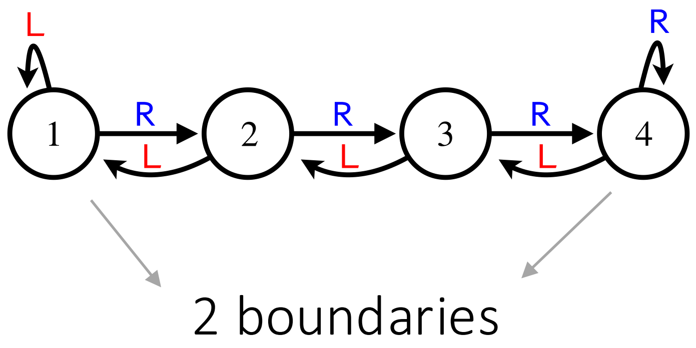
Cyclic groups (mod-n)

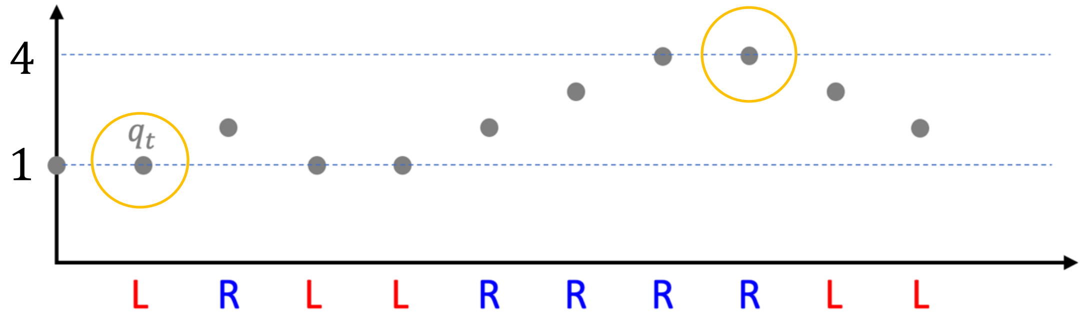
Non-solvable groups

automaton	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Dyck	99.3	100	[illegible]	100	100	100	100	100	100	100	100	100	100	100	100	100
$Grid_4$	99.9	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
$Grid_9$	92.2	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
C_2	77.6	99.8	99.9	100	100	99.5	100	99.7	100	100	100	100	100	100	100	100
C_3	54.6	94.6	96.7	99.4	100	100	99.8	100	99.9	100	100	100	100	100	99.8	100
C_4	95.1	92.3	84.2	99.9	99.7	99.9	100	100	100	100	100	100	100	100	100	100
C_5	89.0	99.1	99.9	100	100	100	100	100	100	100	100	100	100	100	100	100
C_6	59.8	98.7	75.5	99.9	99.8	99.9	99.9	100	100	100	99.8	99.9	100	99.8	99.9	99.9
C_7	90.9	95.0	99.9	99.9	100	99.9	100	100	100	100	100	99.8	100	100	100	100
C_8	79.6	96.2	99.8	99.8	99.9	100	99.9	99.9	100	99.4	99.9	99.9	99.9	100	99.9	99.9
C_2^2	90.5	98.8	99.9	100	100	99.9	100	100	99.9	99.9	100	100	100	100	100	100
C_2^3	65.0	77.9	99.9	97.9	100	99.8	98.2	99.9	100	100	91.9	95.9	91.7	90.6	87.5	80.6
D_6	25.4	27.2	47.4	75.2	100	100	100	100	100	100	100	100	100	100	100	100
D_8	45.6	98.0	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Q_8	31.6	49.2	59.6	60.4	73.5	99.3	100	100	100	100	100	100	100	100	100	100
A_4	25.0	35.4	49.1	59.3	62.6	82.3	90.9	98.0	98.0	99.1	99.8	100	99.7	100	100	100
A_5	12.5	23.1	32.5	46.7	71.2	98.8	100	100	100	100	100	100	100	100	100	100
S_4	11.3	17.6	22.0	27.1	37.7	44.8	50.8	72.5	91.3	97.1	97.9	98.7	99.9	100	99.8	99.9
S_5	7.9	11.8	14.6	19.7	26.0	28.4	32.8	51.8	86.3	94.8	90.2	97.2	99.3	99.1	99.9	99.9

Interpreting gridworld

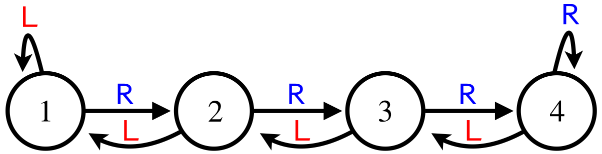
1d gridworld: $Q = \{1, 2, 3, 4\}$, $\Sigma = \{$L, R$\}$.





- State matters: LR $\neq$ RL at state 1, but LR $=$ RL at state 3.

~~*"You can only figure out where you are if you know q_{t-1}."*~~

$O(1)$ layer for 

Puzzle: design a parallel algorithm to compute $\sigma_{1:T} \mapsto q_{1:T}$.

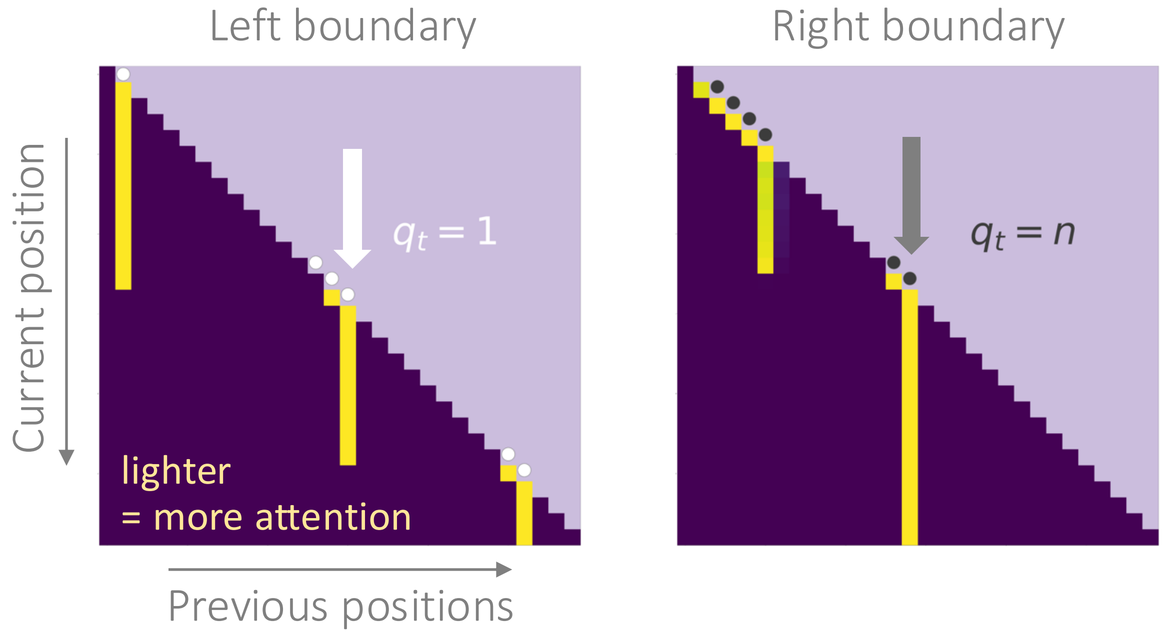
- Hint: *boundary detection*: no boundary = prefix sum.

Transformers find boundaries:  **$O(1)$-layer** (Krohn-Rhodes: $\tilde{O}(|Q|^2)$)

attention heatmaps
(GPT solved this before us)

→ **algorithm extracted**
"mechanistic interpretability"

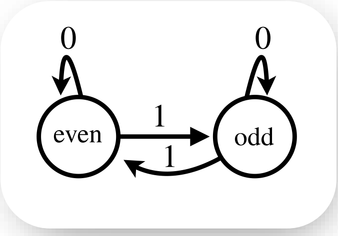
**Caution*: challenges of interpreting attention maps [WLLR NeurIPS23]



Simulating $\mathcal{A}$ in practice

19 automata

$\sigma_{1..T}$
01101100



$q_{1..T}$
01001000

Transformer
with standard training

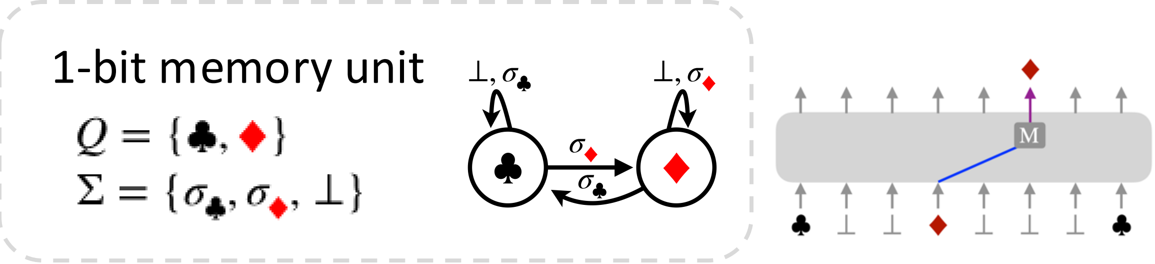
Can shortcuts be found?

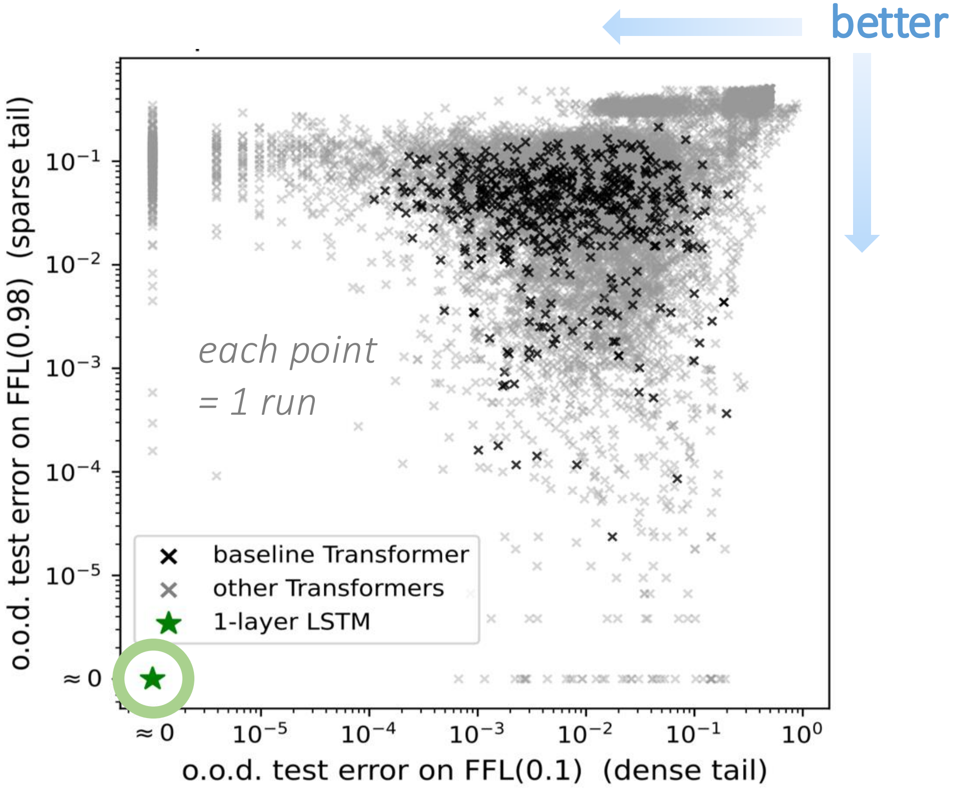
Yes; e.g. gridworld.

Robust Out-Of-Distribution?

Problems with shortcuts?

Flip-flop: A simple task where Transformers struggle out-of-distribution (OOD).





LAGKZ23b [NeurIPS23, spotlight]

Attention glitches: imperfect retrieval.

- Inherent limitations of attention.
- Potential contributor to hallucination.

Mitigation: No perfect mitigations.

- unless introducing *long-tailed data*
  also in prior work, e.g. priming [Jelassi et al. 23].

Flip-Flop Language Modeling (FFLM)

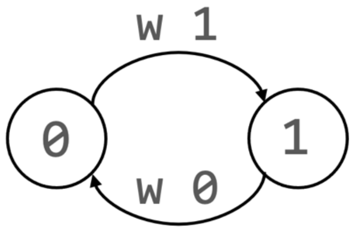
Flip-Flop Language (FFL): sequences of instruction-value pairs.

- 3 instructions: `w` (write) , `i` (ignore) , `r` (read).
- 2 values: {0, 1}; the value for `r` must be the same as the last `w`.

`w 1 i 0 i 1 r 1 w 0 i 1 r 1`

✅ ❌

Task: predict values following `r` (i.e. locate the most recent `w`)



"flip-flop" the state

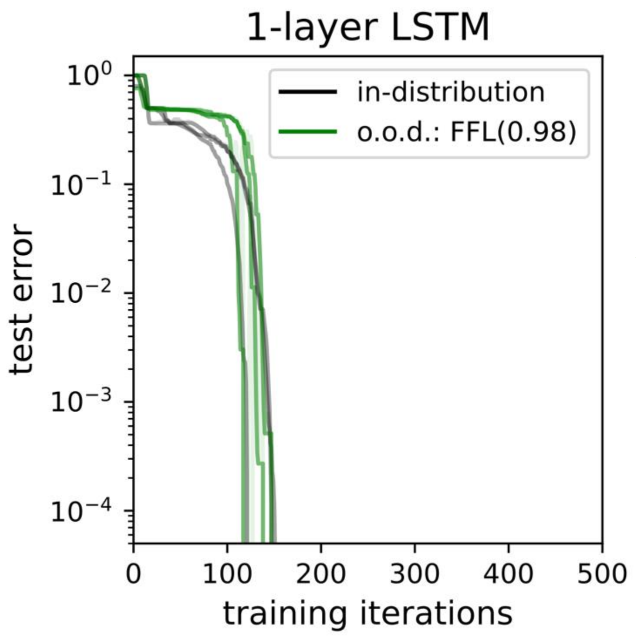
Distributions: FFL(p_i): $p_w = p_r = \frac{1-p_i}{2}$.

Why FFLM?

- An atomic unit underlying reasoning tasks [LAGKZ23a].
- Simple yet interesting.

FFLM Results

Train: FFL(0.8) $T = 512$
Test: FFL(0.9) ... *sparser*



Attention glitches

Attention Glitches

FFL(p_i): various T, $p_w = p_r = 0.2$.

Def: *imperfect hard retrieval*.

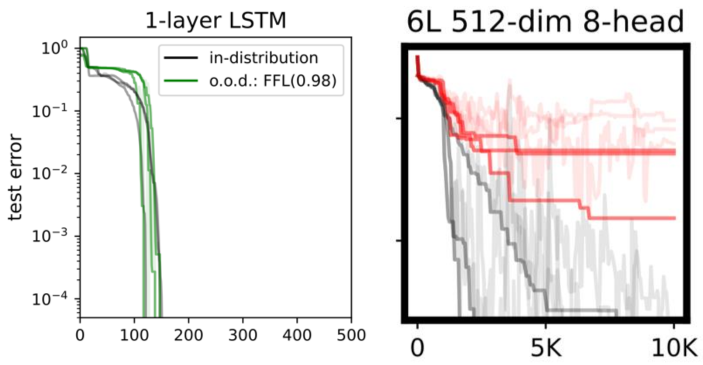
(R1) Transformers exhibit a long tail of errors.

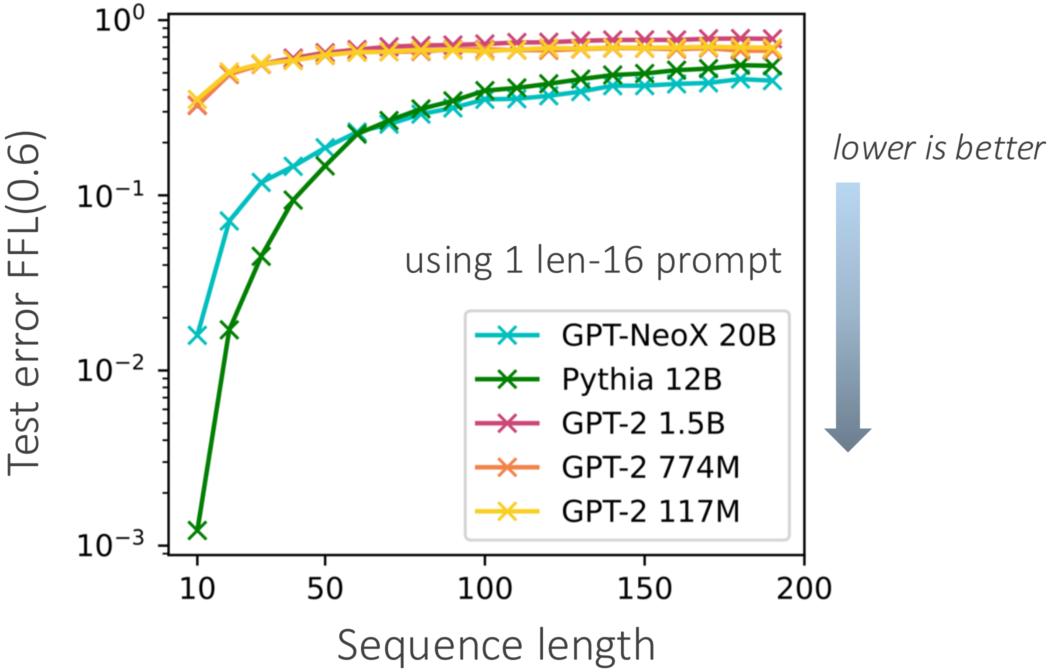
(R2) 1-layer LSTMs extrapolate *perfectly*.

(R3) 10B-scale *natural-language* models are not robust either.

`w 0 i 0 i 1 w 1 i 1 i 0 r ?`

```
Alice turns the light off.
Then, Bob eats an apple.
Then, Bob eats a banana.
Then, Alice turns the light on.
Then, Bob eats a banana.
Then, Bob eats an apple.
Now, the light is
```





What causes glitching attentions?

FFL(p_i): $p_w = p_r = \frac{1-p_i}{2}$.

Not because of limitation on representation power (solvable with *2-layer 1-head*).

Diluted soft attention: caused by more items in the softmax.

$$a_{\max} = \frac{\exp(z_{\max})}{\underbrace{\exp(z_1)+\cdots+\exp(z_t)}_{\text{e.g. ignores, earlier writes}}+\exp(z_{max})}$$

- Pointed out in prior work [Hahn 20, Chiang & Cholak 22].
- Possible mitigation: scaling the logits (e.g. by $\log T$), hard attention.

What causes glitching attentions?

FFL(p_i): $p_w = p_r = \frac{1-p_i}{2}$.

Not because of limitation on representation power (solvable with *2-layer 1-head*).

Diluted soft attention: more items in the softmax: scaling, hard attention.

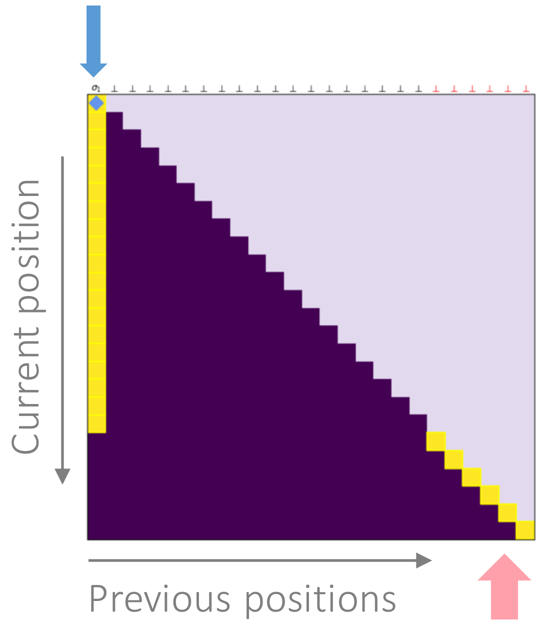
→ *failure on denser sequences (more* **w***)*

Wrong argmax: hard attention won't work.

Setting: simple flip-flop: 1-layer 1-head.

Unlikely to precisely meet a necessary condition for linear positional encoding.

→ *failure on sparser sequences (fewer* **w***)*

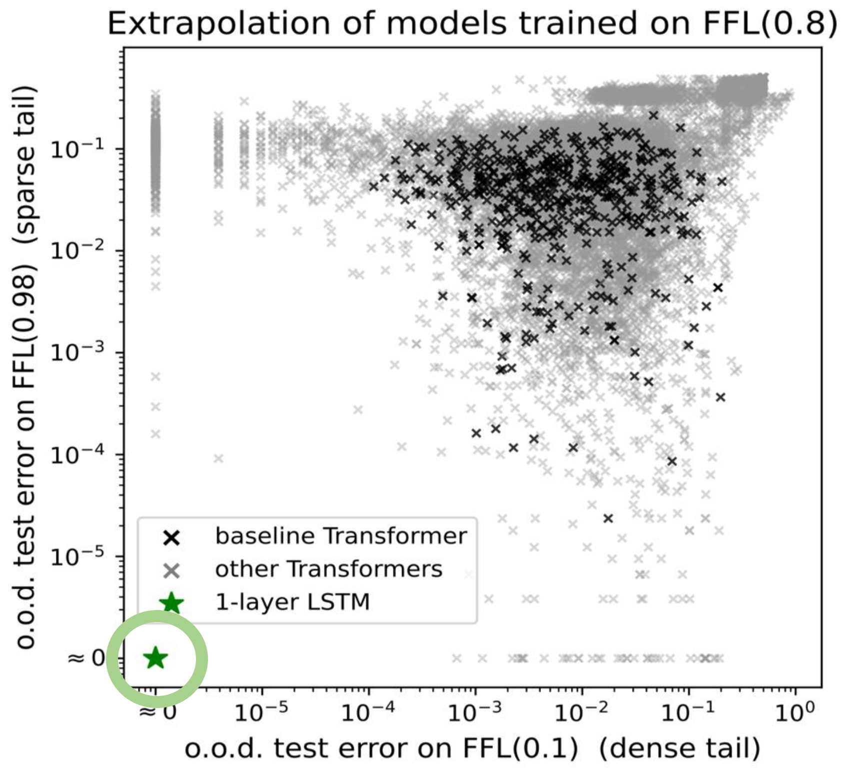


Mitigations to attention glitches

- Incorporating OOD data. *direct*
  Ideal solution – No OOD issue if everything is in distribution! :)
- Resource scaling: larger, train for longer.
  Fresh samples → better coverage

- Standard regularization *indirect*
  e.g. weight decay, dropout, position encoding.
- Attention-sharpening losses (entropy, $-\ell_2, -\ell_\infty$)

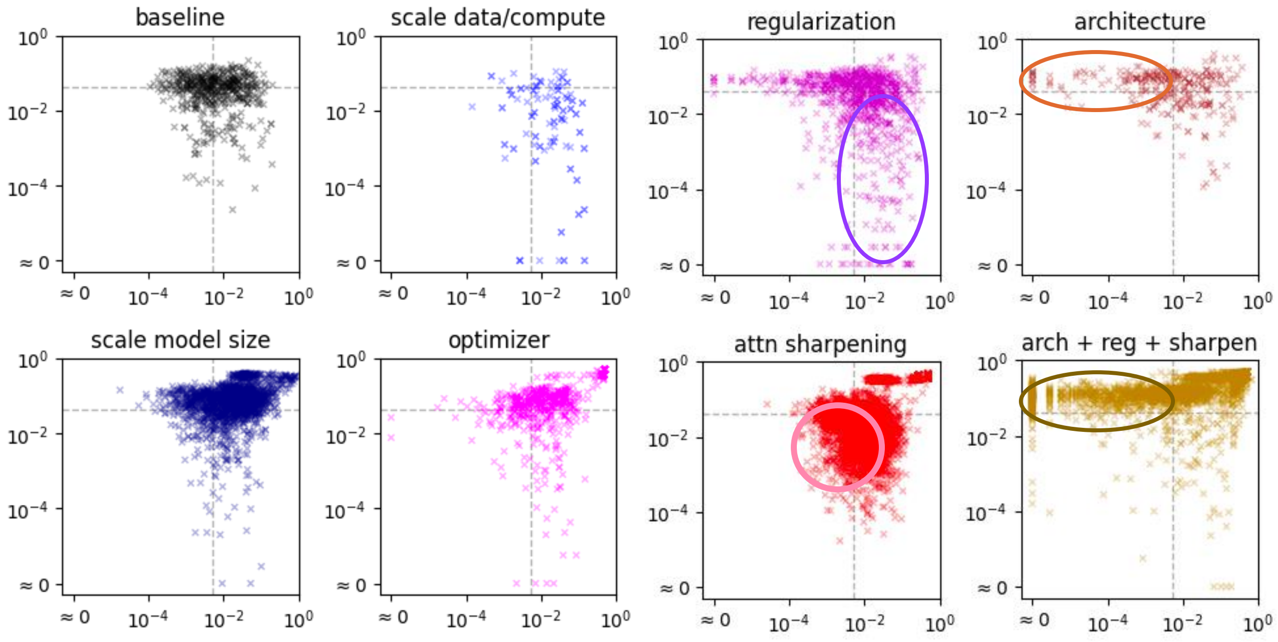
No perfect mitigations, except for OOD data.

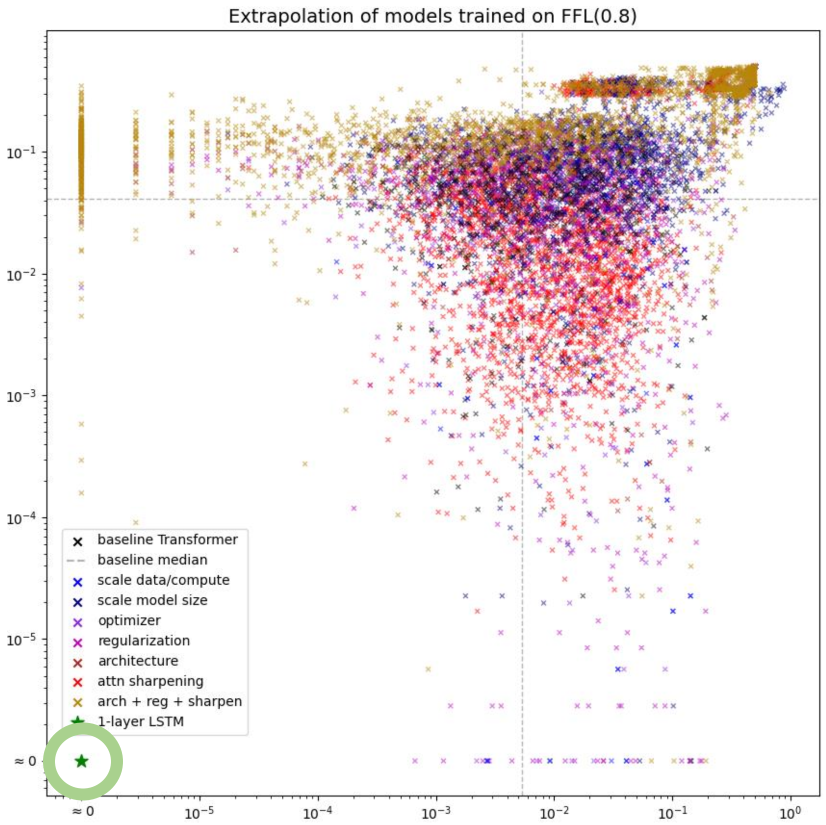


Attention glitches: no perfect mitigations

Dense-sparse trade-off: seldom improve both.

denser = x-axis, sparser = y-axis.

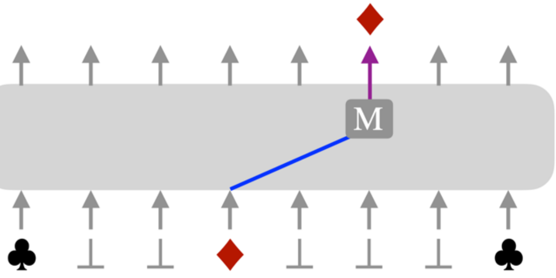




LSTM

OOD failures – the 2 atomic units

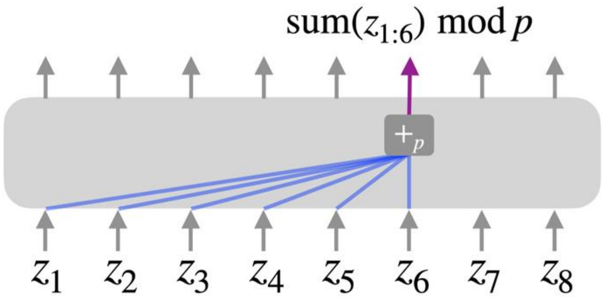
- Flip-flop → sparse attention



Attention: Flip-Flop Language Modeling

... the simplest setup where (closed-domain) hallucination occurs.

- Parity → uniform attention



MLP: $q_t = \left(\sum_{i\in[t]} \sigma_i\right) \bmod 2$

↓

memorized → fail at unseen $\left(\sum_{i\in[t]} \sigma_i\right)$.

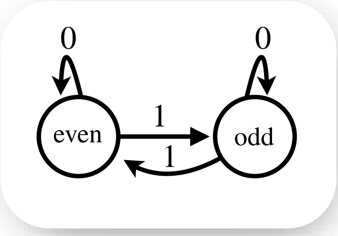
(failure of ReLU [Xu 20])

Solution: periodic activation, e.g. $\sin(x)$.

Empirical results

19 automata

$\sigma_{1..T}$
01101100



$q_{1..T}$
01001000

Transformer
with standard training

Can shortcuts be found?

Yes; e.g. gridworld.

Robust Out-Of-Distribution?

Failure of: *1. Attention (flipflop)*
2. MLP (parity)

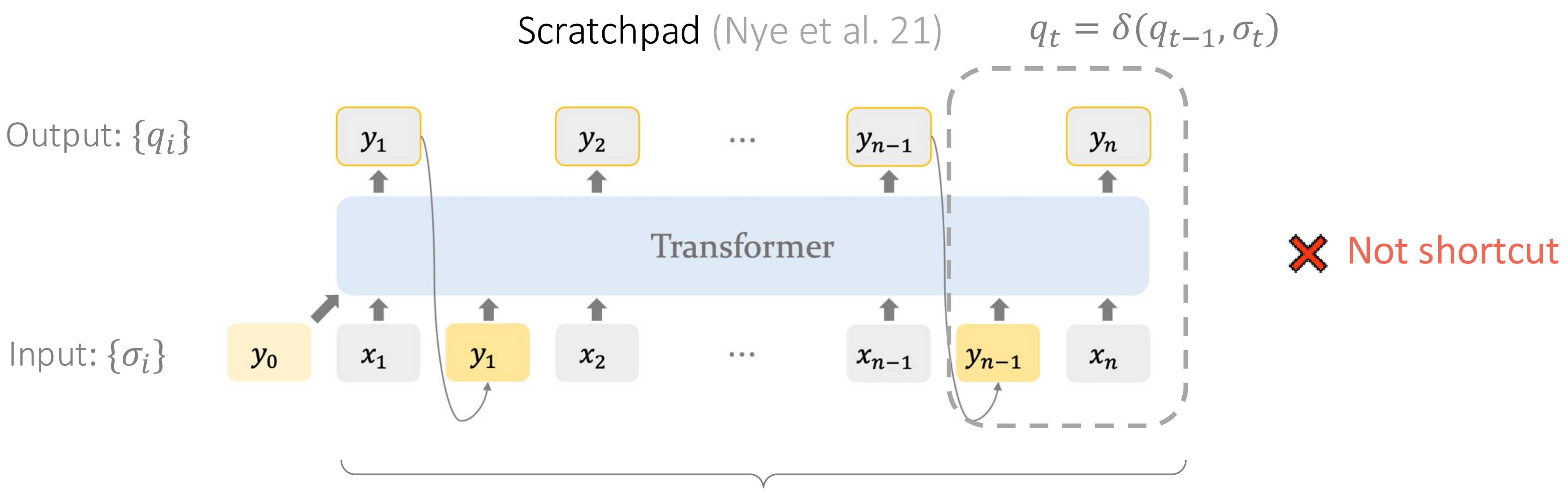
Any fixes?

Computational shortcuts exist, but practical statistical shortcuts are brittle.

😃 🙁

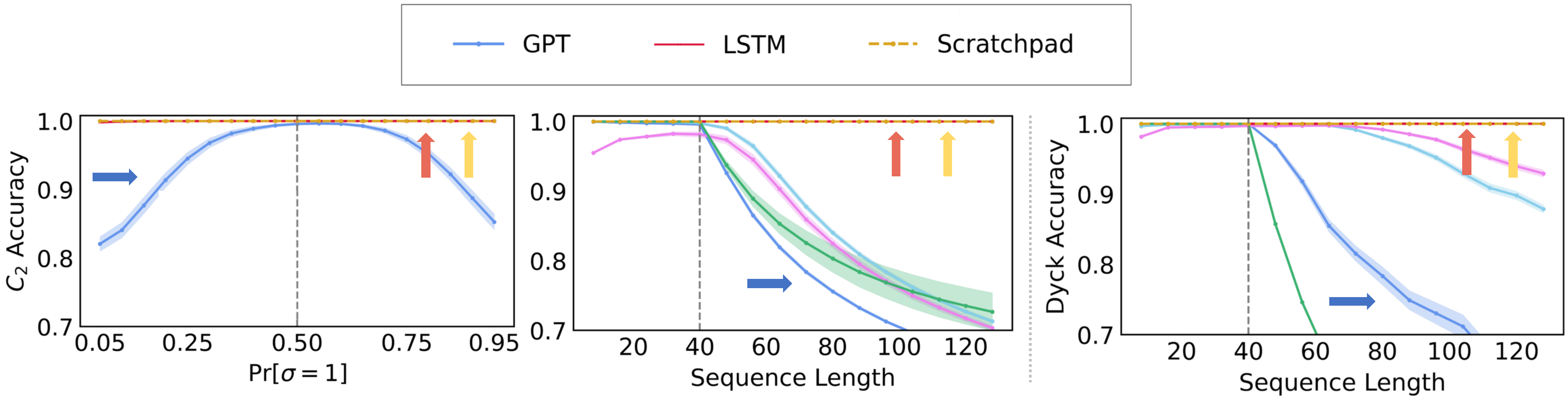
Autoregressive mode of Transformers

Fix to OOD: iterative/autoregressive solutions: use q_{t-1} as inputs.



No longer parallel across positions

Autoregressive mode of Transformers

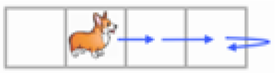


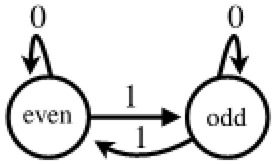
Transformers generalize, when made autoregressive with scratchpad [Nye et al. 22].

→ *Can we learn shortcuts that generalize?* *... attention glitches (flipflop)*

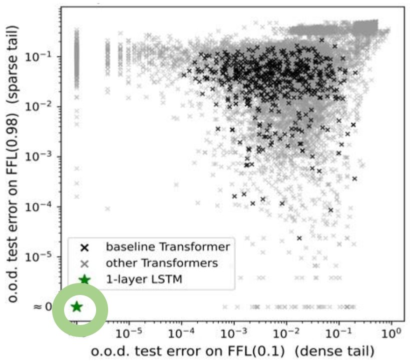
Transformers Learn Shortcuts to Automata

Parallel solutions to sequential reasoning problems.

- **Theory:** Transformers learns $o(T)$ layers ***shortcuts***.
  - All $\mathcal{A}$: $O(\log T)$ layers: divide-and-conquer.
    - This is also the lower bound for the general case.
  - All *solvable* $\mathcal{A}$: $O_{|Q|}(1)$ layers: Krohn-Rhodes Theory.
    - Special case: $O(1)$-layer simulation. 



- **Empirical study**: shortcuts can be found in practice.
  - *Benefit*: sequential computation steps ≪ reasoning steps.
  - *Weakness*: the shortcuts are **brittle OOD, hallucination**.
    - No perfect parallel solutions yet.



Discussions

What can we learn from small-scale experiments?

- FFLM extensions: more values, selection criteria (multi-step reasoning).
- What insights transfer across scale? e.g. sharpen attention for code/math?

Perfect accuracy? More comprehensive metrics; understand the errors.

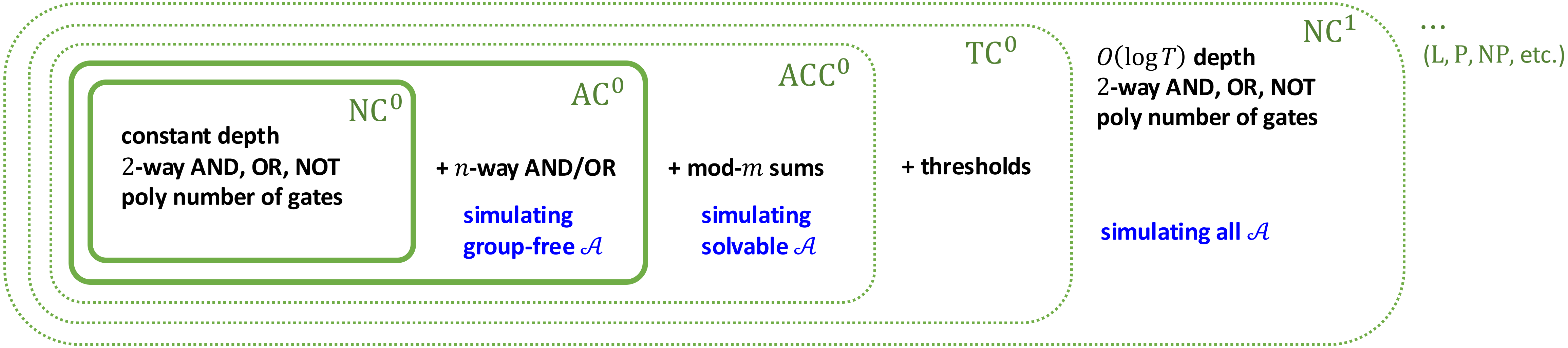
Architectural changes? e.g. recurrence, Mamba (S6), built-in operators.

Theory? Representational, optimization (stability), generalization.

Appendix

Quantifying efficient parallel circuits

- **Goal:** formalize *“Krohn-Rhodes implies efficient simulation”*
- Low-depth parallel algorithms are best captured by **circuit complexity**



Embarrassingly open: are any of the ⬚ proper? $\mathrm{ACC}^0 \overset{?}{=} \mathrm{NP}$?

Factorization: from integers to groups

$$8 = 2 \times 2 \times 2$$

- Why groups get complicated: **combinatorial explosion**

C_8: mod-8 addition
$E_8 \cong C_2 \times C_2 \times C_2$: 3-bit vectors under XOR
$C_4 \times C_2$: non-interacting mod-4 & parity
$D_8 \cong C_4 \rtimes C_2$: rotations/reflections of a square
Q_8: multiplication of unit quaternions

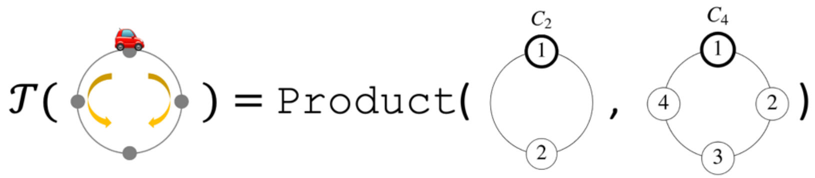
(the last two:) **non-abelian:** $gh \neq hg$

- **Finite group theory:** classical toolbox for understanding symmetries

$$C_8, E_8, C_4 \times C_2, D_8, Q_8 \leq (C_2 \wr C_2) \wr C_2$$

Jordan-Hölder factors (simple groups)
Krasner-Kaloujnine embedding (wreath product)

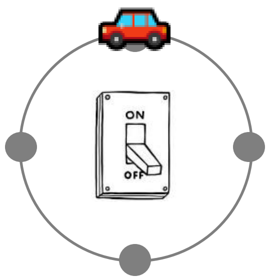
Decomposition: the glue



Direct product $\times$, e.g. $C_4 \times C_2$

Two independent groups

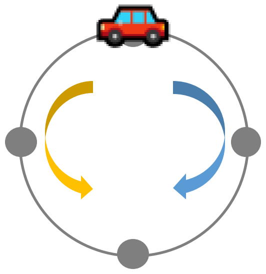
- $(g_1, h_1) \cdot (g_2, h_2) = (g_1 g_1, h_1 h_2)$
- e.g. car + a light switch

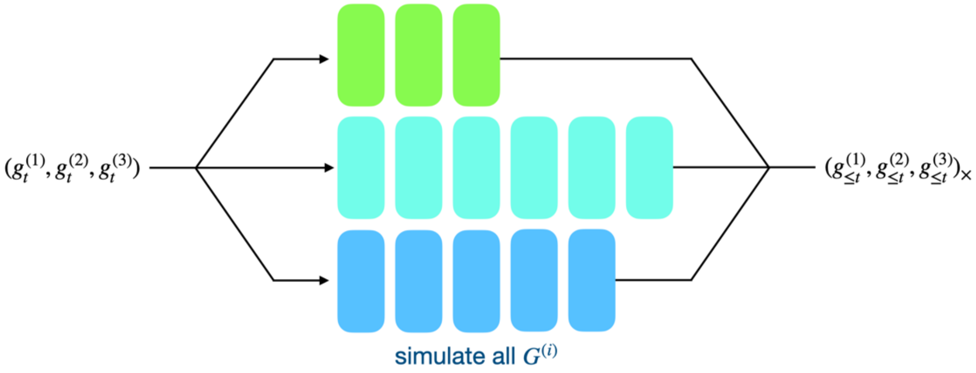


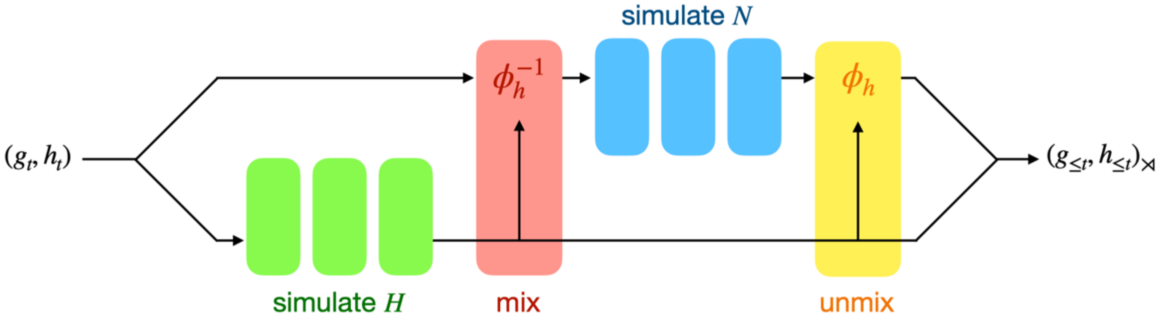
Semidirect product $\rtimes$, e.g. $D_8 \cong C_4 \rtimes C_2$

Two *interacting* groups

- $(g_1, h_1) \cdot (g_2, h_2) = (g_1 h_2 g_2 h_2^{-1}, h_1 h_2)$
- e.g. car + direction toggle



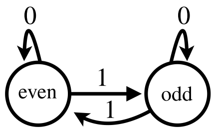




Transformation semigroups

$$q_t = \left(\delta(\cdot, \sigma_t) \circ \cdots \circ \delta(\cdot, \sigma_1)\right)(q_0)$$

$\mathcal{T}(\mathcal{A}) \coloneqq \{\delta(\cdot, \sigma) : \sigma \in \Sigma\}$ under composition (associativity).



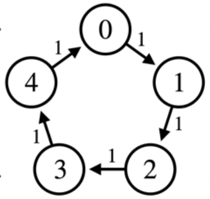
$Q = \{\text{even}, \text{odd}\}$
$\Sigma = \{0, 1\}$

parity counter

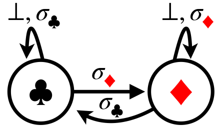
$\mathcal{T}(\mathcal{A})$ →

	0	1
0	0	1
1	1	0

cyclic group C_2

$\mathcal{T}$ () $\cong C_5$

cyclic group C_5



$Q = \{♣, ♦\}$
$\Sigma = \{\sigma_♣, \sigma_♦, \bot\}$

1-bit memory unit

$\mathcal{T}(\mathcal{A})$ →

	$\sigma_♦$	$\sigma_♣$	$\bot$
$\sigma_♦$	$\sigma_♦$	$\sigma_♦$	$\sigma_♦$
$\sigma_♣$	$\sigma_♣$	$\sigma_♣$	$\sigma_♣$
$\bot$	$\sigma_♦$	$\sigma_♣$	$\bot$

non-invertible (rows $\sigma_♦$, $\sigma_♣$)

flip-flop monoid

Group G: a set G with operation $G \times G \to G$.

- Associativity: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
- Identity: $a \cdot e = e \cdot a = a$
- Inverse: $\forall a \in G, \exists b \in G$ s.t. $a \cdot b = b \cdot a = e$

Semigroup G: a generalization of group.

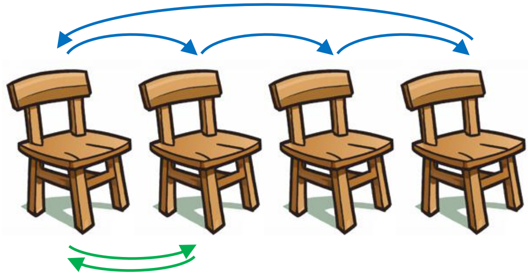
- Associativity.
- (+ Identity: a *monoid*.)

What about *semi*groups?

$\mathcal{T}(\mathcal{A}) \coloneqq \{\delta(\cdot,\sigma) : \sigma \in \Sigma\}$ under composition

More complicated: rank collapses.

n-player musical chairs



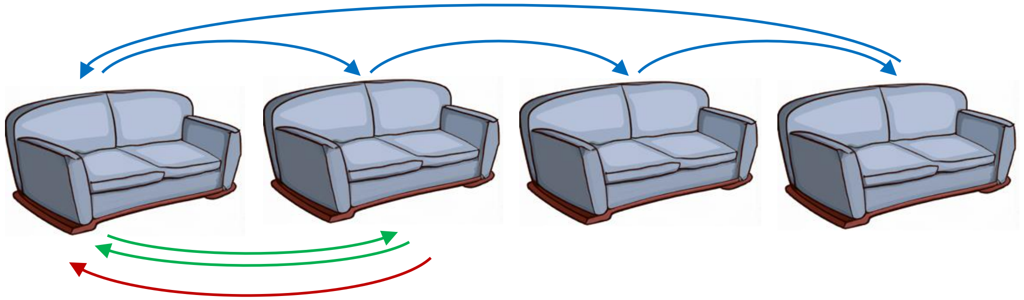
$Q = \{\text{positions of } n \text{ players}\}$

$\Sigma = \{ \text{ cycle}, \quad \text{swap } \}$

$\begin{bmatrix} & & & 1 \\ 1 & & & \\ & 1 & & \\ & & 1 & \end{bmatrix} \quad \begin{bmatrix} & 1 & & \\ 1 & & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$

$\mathcal{T}(\mathcal{A}) = S_n$: all $n!$ permutations on $[n]$

n-player musical sofas



$Q = \{\text{positions of } n \text{ players}\}$

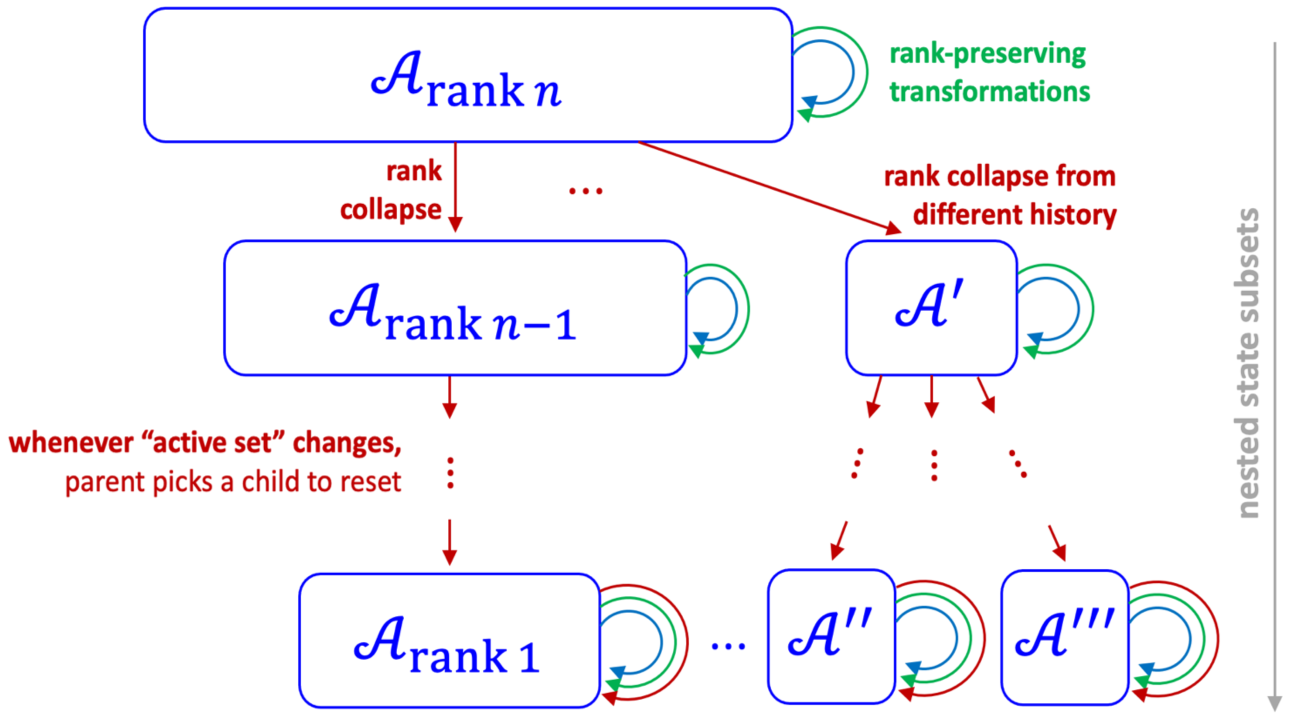
$\Sigma = \{ \text{ cycle}, \quad \text{swap}, \quad \text{merge } \}$

$\begin{bmatrix} & & & 1 \\ 1 & & & \\ & 1 & & \\ & & 1 & \end{bmatrix} \quad \begin{bmatrix} & 1 & & \\ 1 & & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & & \\ & & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$

$\mathcal{T}(\mathcal{A}) = T_n$: all n^n functions $[n] \to [n]$

Krohn-Rhodes Intuitions

Tracking rank collapses (*holonomy decomposition)*



Number of layers: (recall: $|G| \leq n^n$)

- Solvable groups: $O(\log|G|)$
  - mod counter
- Permutation-reset semiautomaton: $O(\log|G|) + 2 \leq O(|Q| \log|Q|)$.
  - mod counter + memory unit
- Semiautomaton: $\leq |Q|$ levels.

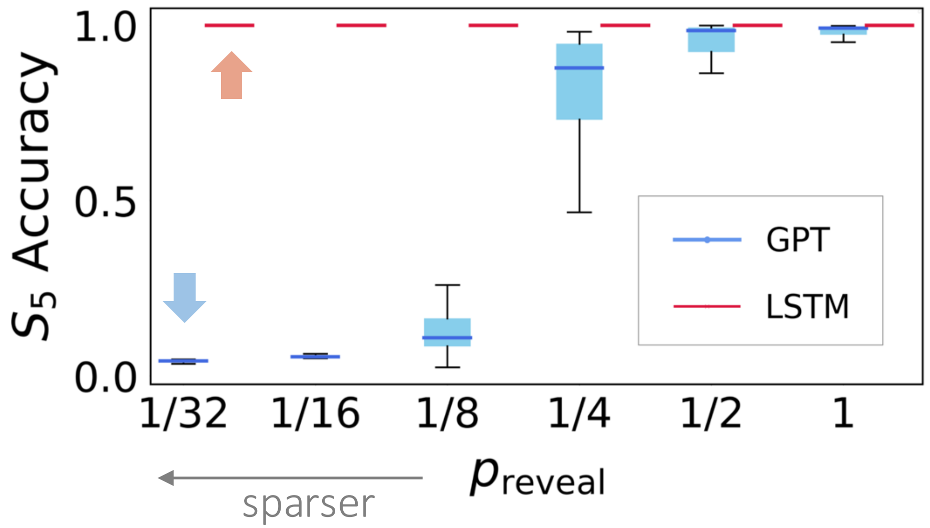
Training with limited supervision

Less ideal setups?

Indirect supervision
train & test on a function of q_t.

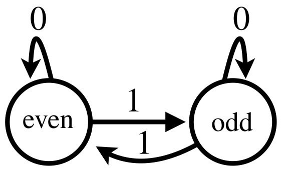
$\text{Dyck}_{4,8}$	Grid_9	S_5	C_4	D_8
stack top	$\mathbb{1}_{\text{boundary}}$	$\pi_{1:t}(1)$	$\mathbb{1}_{0 \bmod 4}$	location
100.0	99.8	99.8	99.7	99.8

Incomplete supervision
q_t is revealed w.p. $p_{\text{reveal}} \in [0,1]$.



LSTM is always 100% → Open: *How to improve Transformer training?*

OOD Generalization - Parity



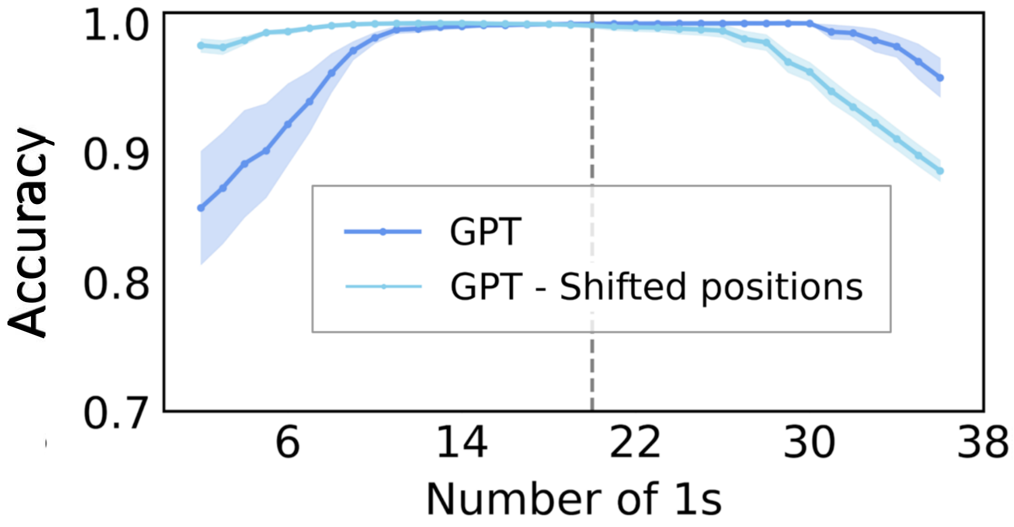
$(T = 40)$

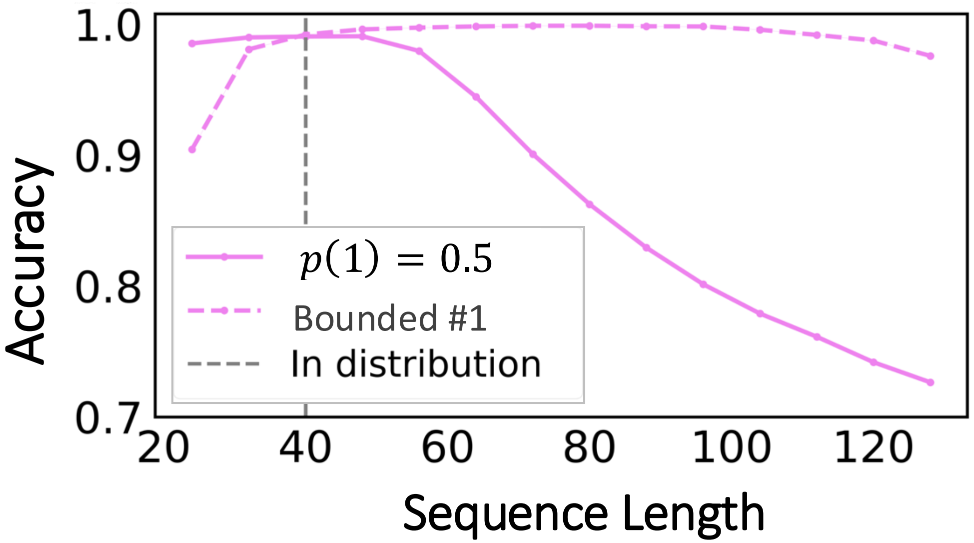
- train: $p(1) = 0.5$
- test: other $p(1)$.

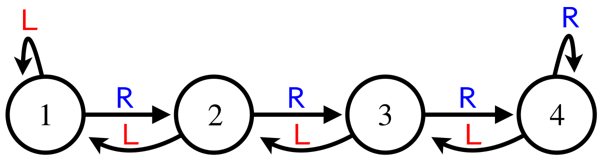
$$q_t = \left(\sum_{i\in[t]} \sigma_i\right) \bmod 2$$

concentrates $\approx p(1) \cdot t$

memorized by MLP $\rightarrow$ fail at unseen $\left(\sum_{i\in[t]} \sigma_i\right)$.

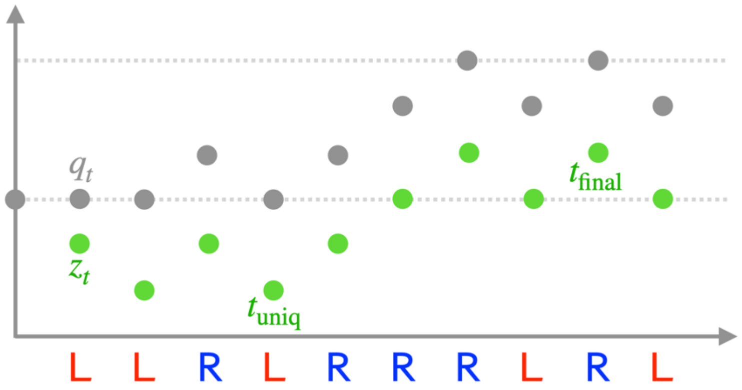




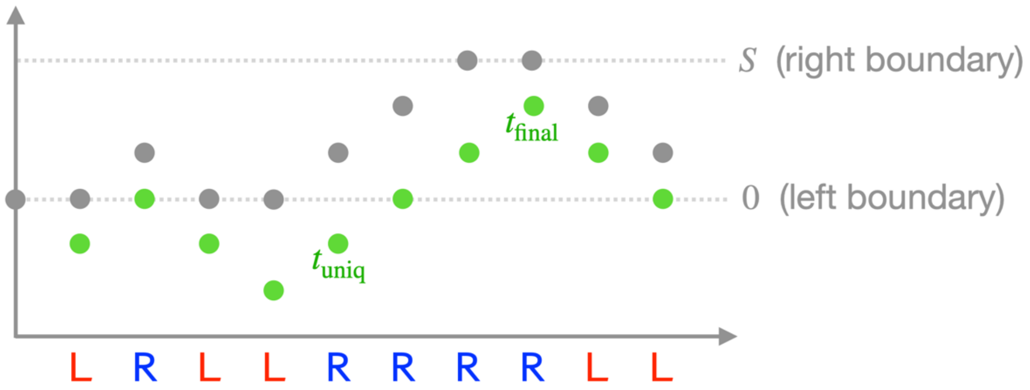
$O(1)$ layer for 

- **Parallel boundary detector**:
  - Compute prefix sums $z_t \coloneqq \sum \sigma_{1:t}$ (ignoring boundaries);
  - At each t, find most recent $t_{\text{uniq}} < t$ such that $z_{t_{\text{uniq}}:t}$ has n(#states) unique values;
  - Then $t_{\text{final}} \coloneqq \max\left(\underset{t_{\text{uniq}} \le \tau \le t}{\operatorname{argmax}} z_\tau,\ \underset{t_{\text{uniq}} \le \tau \le t}{\operatorname{argmin}} z_\tau\right)$ is last boundary collision.

Case 1: traversal



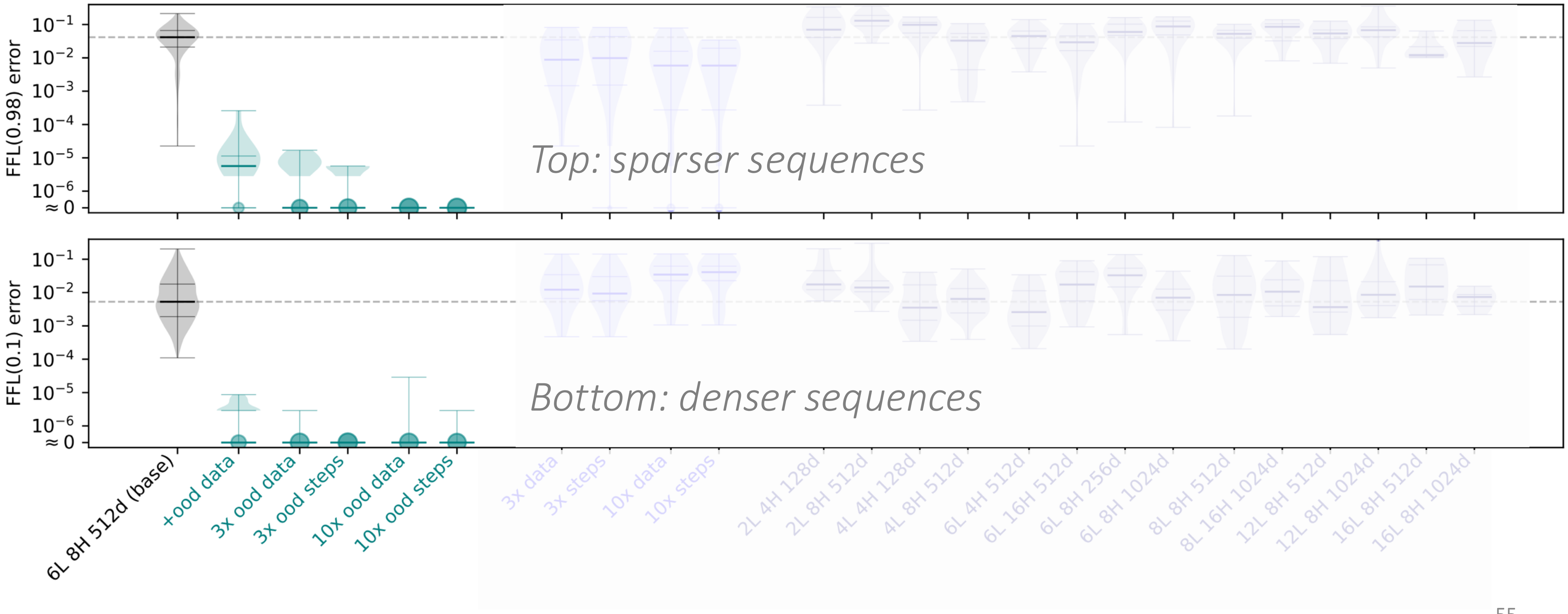
Case 2: hit one wall



Direct mitigations

(R4) Incorporating OOD data (*"priming"*) works the best, by far.

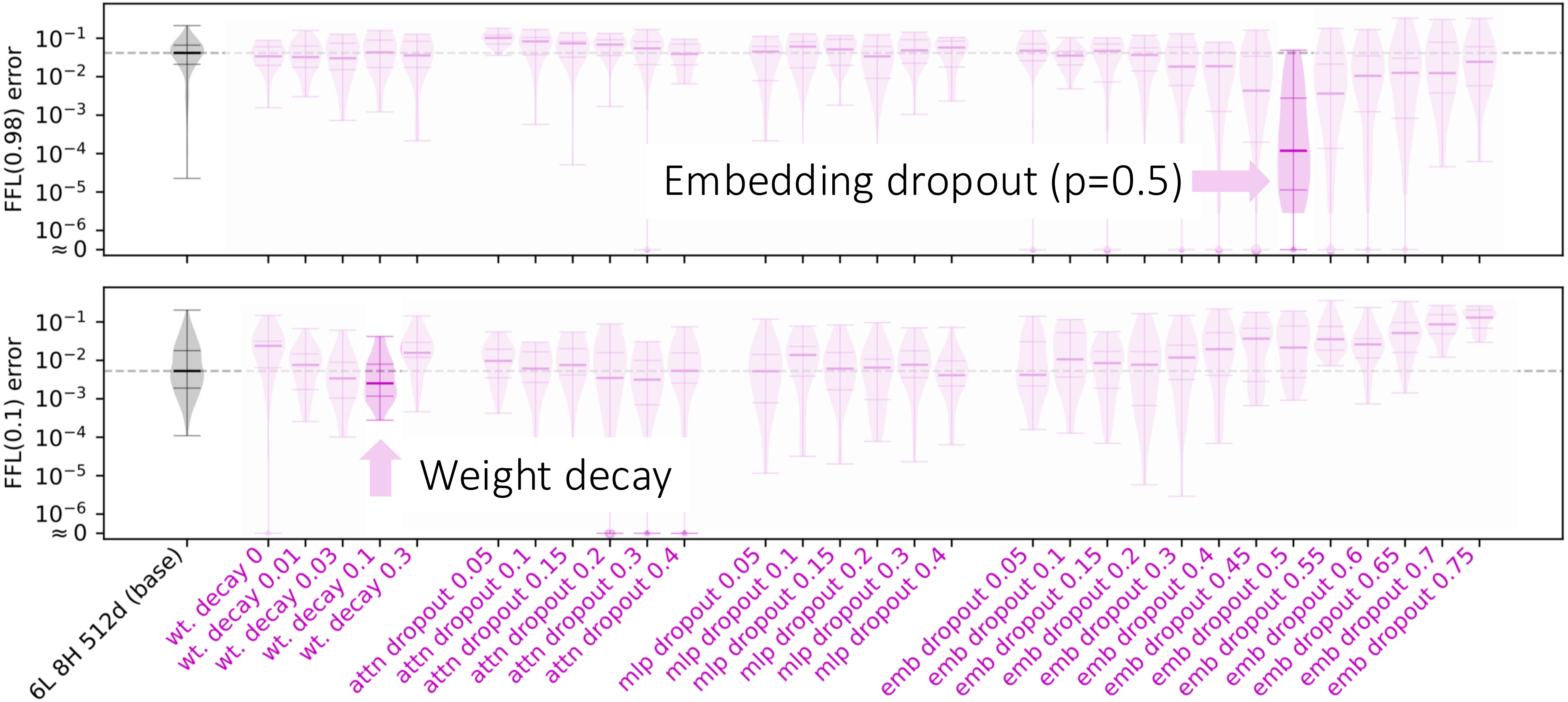
[Jelassi 23]



Indirect mitigations

- Weight decay
- Dropout (attention, MLP, embedding)

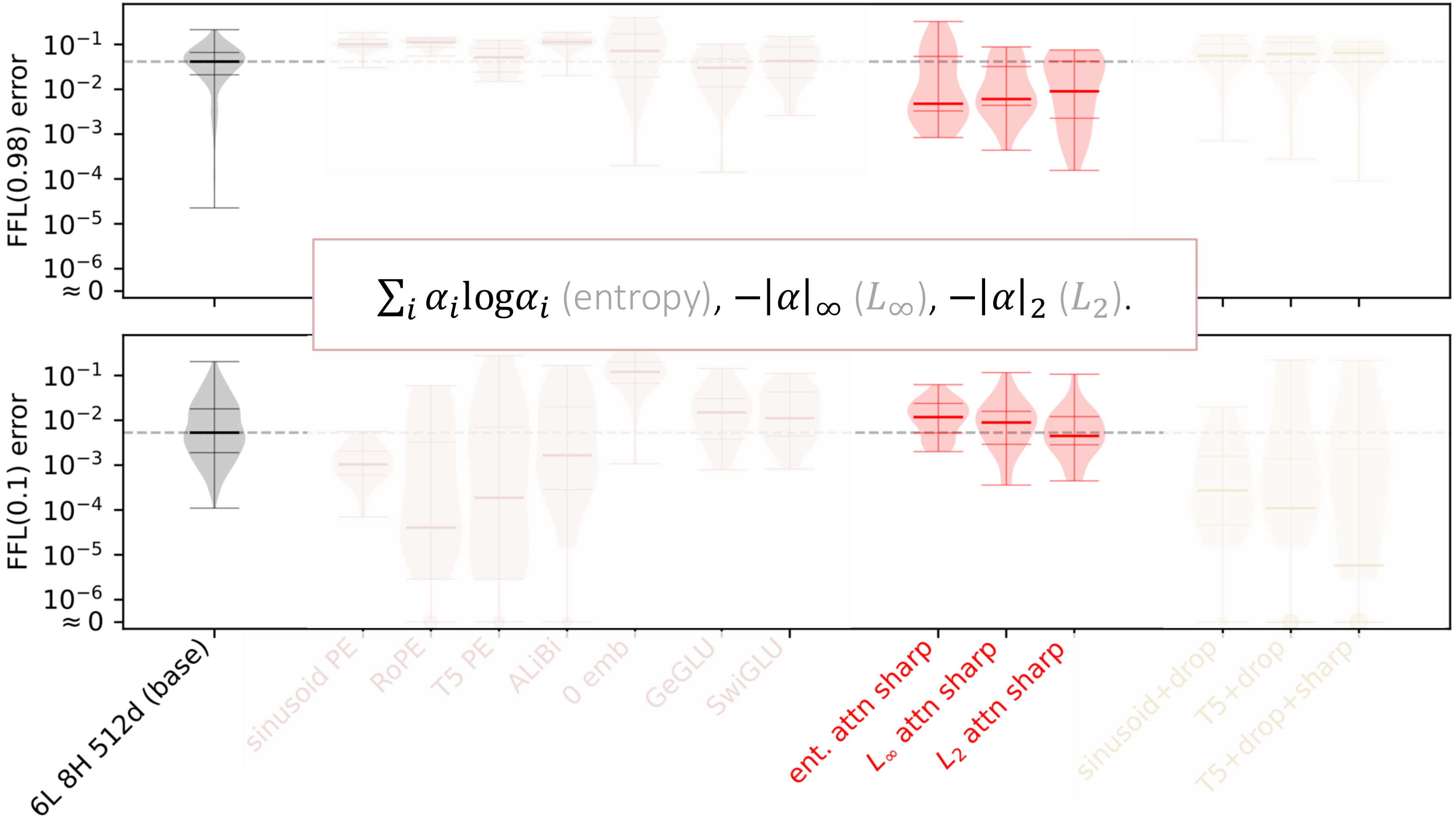
(R6) Standard regularizations have various influences.



Indirect mitigations

Attention-sharpening regularization

(R6) Standard regularizations have various influences.

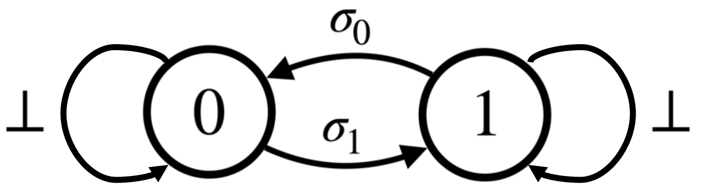


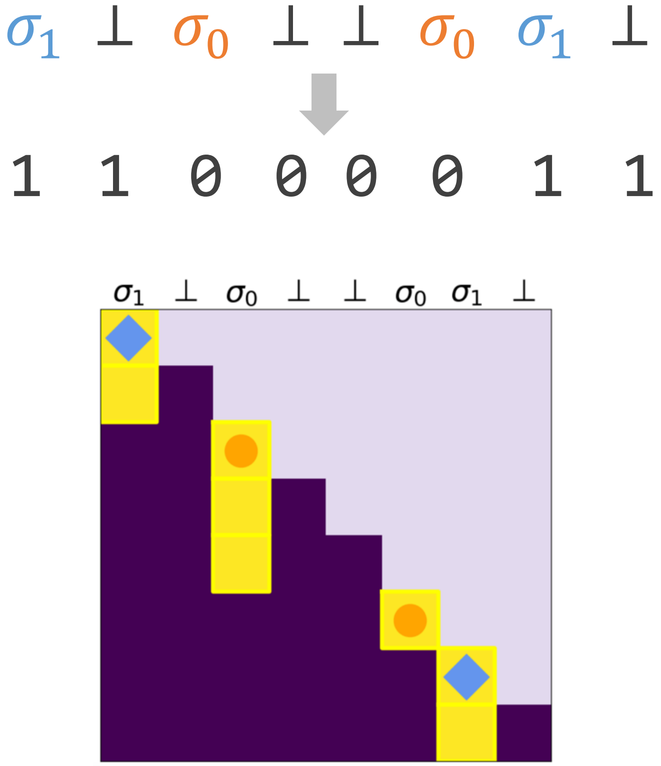
$\sum_i \alpha_i \log \alpha_i$ (entropy), $-|\alpha|_\infty$ (L_∞), $-|\alpha|_2$ (L_2).

Preliminary interpretability results

Simpler setup: flip-flop monoid

- `w0` $= \sigma_0$, `w1` $= \sigma_1$, `i` $= \bot$; read at each step.

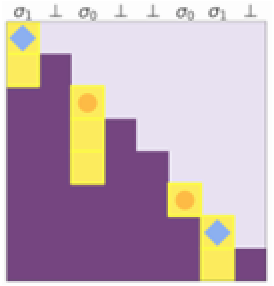




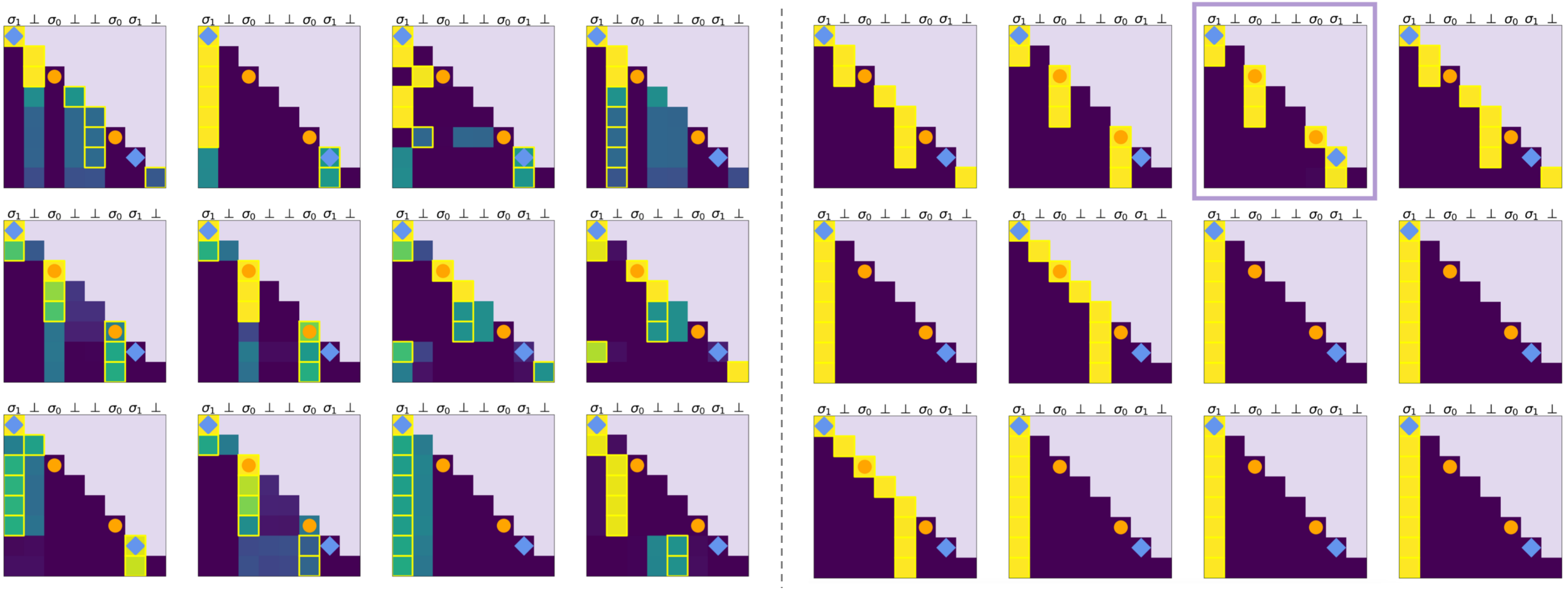
Solvable by 1-layer 1-head Transformers.

- 1-sparse attention: on the closest **`0`**,**`1`**.

What solutions are found?

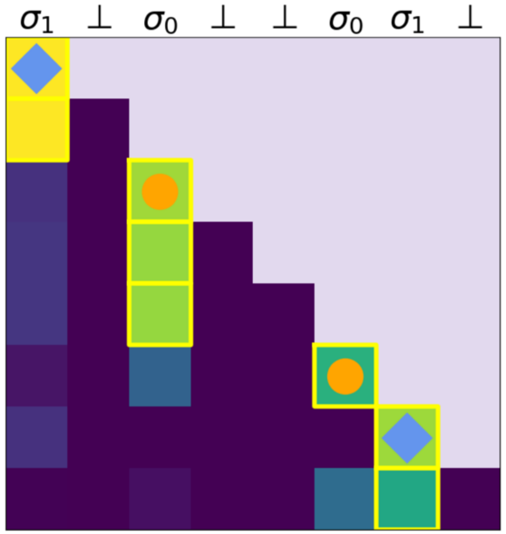


6-layer 8-head ... normal (left) vs attention-sharpened (right)

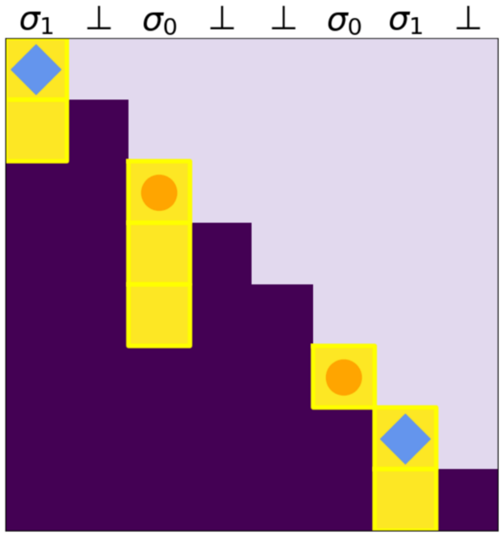


What solutions are found?

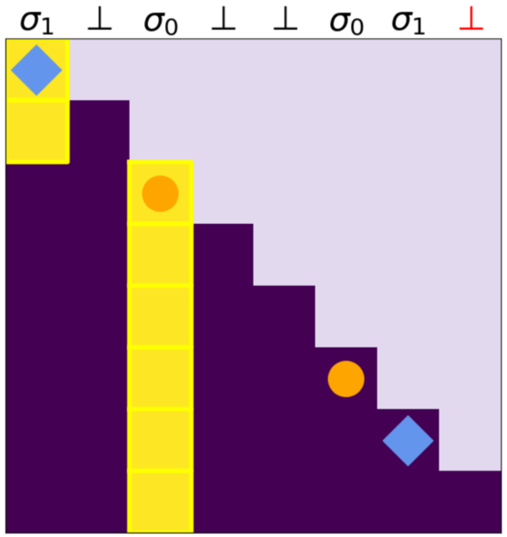
1-layer 1-head: normal vs attention-sharpened.



(a) normal



(b) sharpened 1



(c) sharpened 2

other dense/sparse patterns exist

wrong prediction!