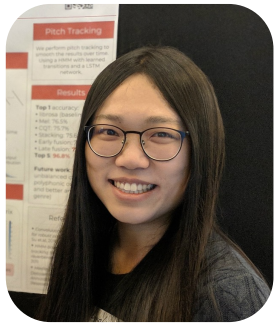
ICLR

Carnegie Mellon University | Penn University of Pennsylvania | Microsoft

# Transformers Learn Shortcuts to Automata

Bingbin Liu
CMU

Jordan T. Ash
MSR NYC

Surbhi Goel
UPenn

Akshay Krishnamurthy
MSR NYC

Cyril Zhang
MSR NYC

# Reasoning abilities of LLMs

Logical reasoning

Code completion

Math

B

Hi! There are two bunnies in front of a bunny, two bunnies behind a bunny and a bunny in the middle. How many bunnies are there?

There are three bunnies in total. Let's break it down:

1. Two bunnies in front of a bunny (this implies there's a bunny in the middle).
2. Two bunnies behind a bunny (these are the same two bunnies in front of the middle bunny).
3. A bunny in the middle (already mentioned in the first point).

So, there are two bunnies and the one in the middle, making a total of three bunnies.

GitHub Copilot

```go
package main

type Run struct {
    Time int // in milliseconds
    Results string
    Failed bool
}

// Get average runtime of successful runs in seconds
func averageRuntimeInSeconds(runs []Run) float64 {
    var totalTime int
    var failedRuns int
    for _, run := range runs {
        if run.Failed {
            failedRuns++
        } else {
            totalTime += run.Time
        }
    }

    averageRuntime := float64(totalTime) / float64(len(runs) - failedRuns) / 1000
    return averageRuntime
```

B

Hi, what is 999997911991919 + 87299919111191?

To add these two large decimal numbers, align them by their least significant digits and perform the addition:

markdown Copy code

```
  999997911991919
+  87299919111191
------------------
 1087297103103110
```

Wrong Answer

So, 999997911991919 + 87299919111191 = 1087297103103110.

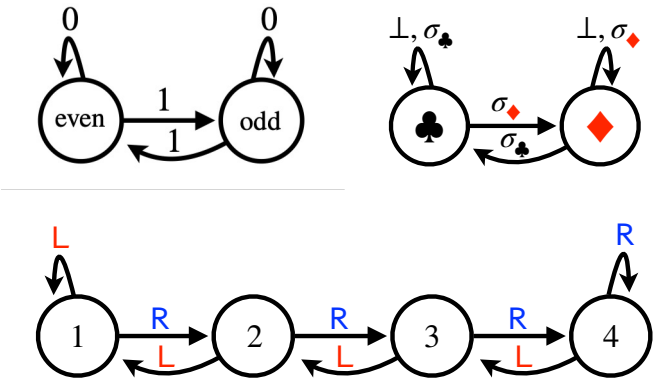
*Q: What internal mechanism does Transformer use to reason?*

# How Transformers reason

*Q: What internal mechanism do Transformers use to reason?*

**Theory**: how *can* Transformers reason?
… representation

**Empirics**: how *do* Transformers reason?
… optimization, generalization




**TL;DR**: Transformers reason with shortcuts.

# Reasoning through the lens of automata

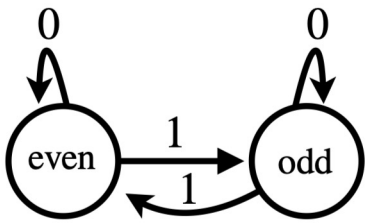
**Automaton $\mathcal{A}$**: a discrete-time dynamical system*

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

**states** **inputs** **transitions**

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

*Example*: which face of the coin is up?




**parity counter**

```
The coin shows heads.
Alice flips it.
Bob doesn't flip it.
Eve flips it.
Now the coin shows ___.
```

$Q$ = {head-up/even tail-up/odd }

$\Sigma$ = {flip/1 do nothing/0 }

$\delta$ = {face changed unchanged}

**This subsumes tasks in prior work such as Bhattamishra et al. 20 and Yao et al. 20.*

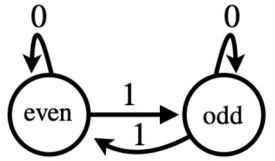
# Reasoning = simulating automata

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

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

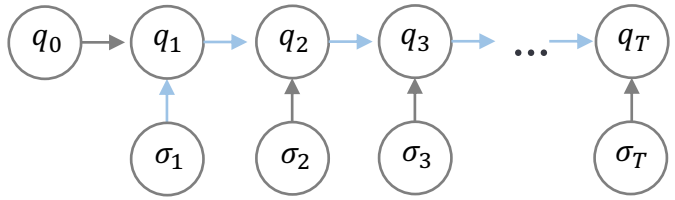
- Input = $\sigma_1, \sigma_2, \cdots, \sigma_T \in \Sigma$, output = $q_1, q_2, \cdots, q_T \in Q$.

*Example*: two solutions to simulate parity




**Iterative solution:** *"RNN solutions"*

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

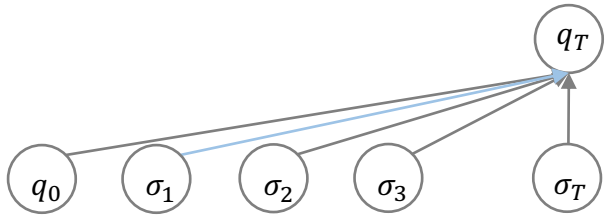



**Parallel solution:** *"Transformer solutions"*

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

Shortcut

$o(T)$ #steps



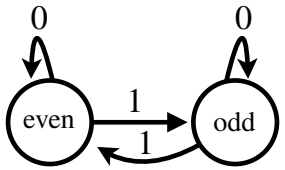
# Algebraic structure of $\mathcal{A}$

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

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

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

- Function composition ($\longleftrightarrow$ matrix multiplication): associative




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

parity counter

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

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

$\Updownarrow$

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

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

| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

cyclic group $C_2$

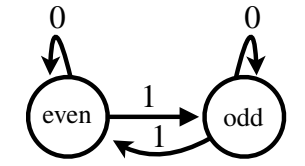
# Algebraic structure of $\mathcal{A}$

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

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

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

- Function composition ($\longleftrightarrow$ matrix multiplication): associative



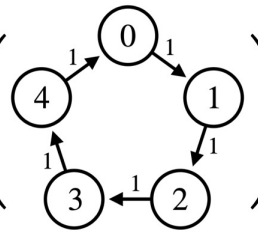

$Q = \{\text{even}, \text{odd}\}$
$\Sigma = \{0, 1\}$

parity counter

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

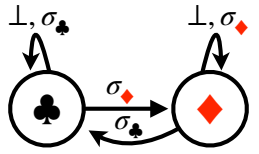
| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

cyclic group $C_2$

$\mathcal{T}\left(\right.$  $\left.\right) \cong C_5$



mod-5 counter




$Q = \{\clubsuit, \blacklozenge\}$
$\Sigma = \{\sigma_\clubsuit, \sigma_\blacklozenge, \bot\}$

1-bit memory unit

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

| | $\sigma_\blacklozenge$ | $\sigma_\clubsuit$ | $\bot$ |
|---|---|---|---|
| $\sigma_\blacklozenge$ | $\sigma_\blacklozenge$ | $\sigma_\blacklozenge$ | $\sigma_\blacklozenge$ |
| $\sigma_\clubsuit$ | $\sigma_\clubsuit$ | $\sigma_\clubsuit$ | $\sigma_\clubsuit$ |
| $\bot$ | $\sigma_\blacklozenge$ | $\sigma_\clubsuit$ | $\bot$ |

flip-flop monoid

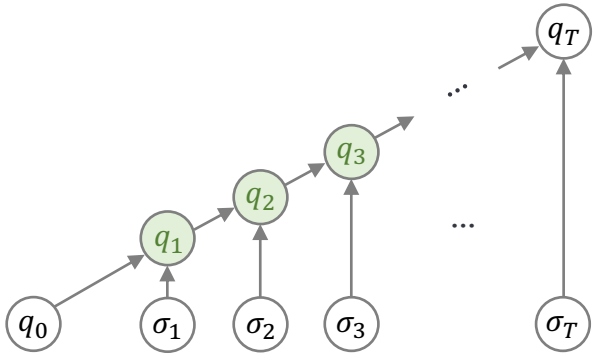
# Reasoning with shortcuts

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

# steps = $T$
no structure required

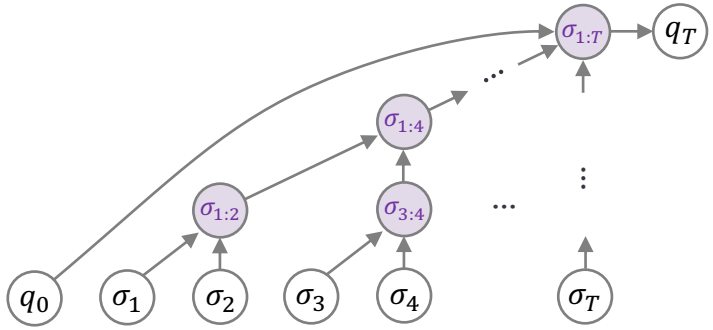
# steps = $O(\log T)$
associativity

# steps = $O_{|Q|}(1)$
Krohn-Rhodes theory



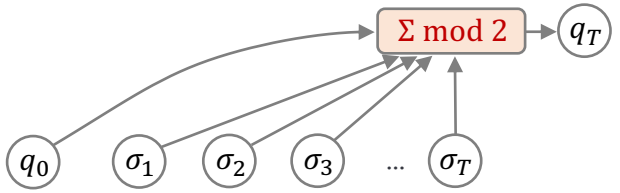

**iterative state emulation**

efficiently represented by RNNs




**multiscale function composition**

[ This only applies to *solvable* $\mathcal{A}$ ]




**global features via algebraic structure**

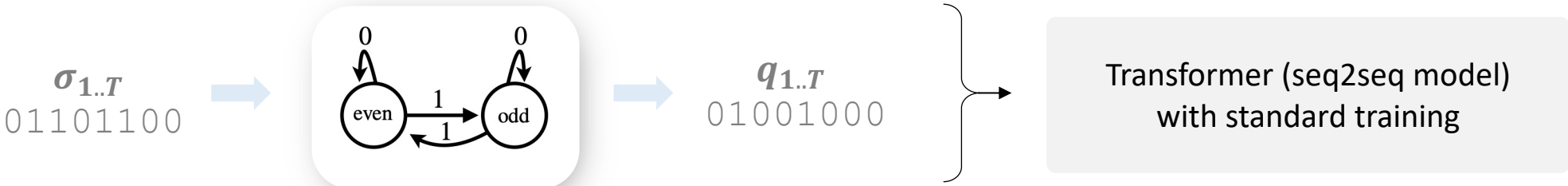
efficiently represented by shallow Transformers

- *# steps = number of sequential steps, i.e. the length of the longest path in the computation graph. This is the same as the number of Transformer layers.*
- *Special case: # steps = O(1).*

# Empirical results

*Transformers can learn shortcuts to automata. But do they in practice?*

Setup: a variety of synthetic tasks; models train to predict $q_t$ given $\sigma_{1..t}$.




***Q1***: Does SGD on Transformers find the shortcuts? Yes!

***Q2***: Does SGD work when there's limited supervision? Up to some point.

***Q3***: Do shortcuts generalize? Yes, but brittle OOD.

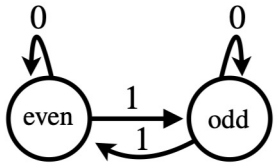
***Q4***: Can we guide Transformers to learn iterative solutions? Yes!

# Transformers Learn Shortcuts to Automata

arxiv: 2210.10749

**TL;DR**: Shallow Transformers can simulate $\mathcal{A} = (Q, \Sigma, \delta)$.

- **Theory:** for any length $T$, Transformers with $o(T)$ layers suffice.
  - All $\mathcal{A}$: $O(\log T)$ layers: divide-and-conquer.
    - This is also the lower bound for the general case.
  - All *solvable* $\mathcal{A}$: $O(|Q|^2 \log |Q|)$ layers: Krohn-Rhodes Theory.
    - Special case: $O(1)$-layer simulation.




- **Empirical study**: Transformers do find shortcuts in practice.
  - *Benefit*: parallel computation steps ≪ reasoning steps.
  - *Weakness*: the shortcuts are brittle OOD, hallucination.

Transformer depth

| semiautomaton | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 12 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|
| Dyck | 99.3 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| $\text{Grid}_9$ | 92.2 | 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 |
| $C_3$ | 54.6 | 94.6 | 96.7 | 99.4 | 100 | 100 | 99.8 | 100 | 100 | 100 |
| $C_2^3$ | 65.0 | 77.9 | 99.9 | 97.9 | 100 | 99.8 | 98.2 | 99.9 | 95.9 | 80.6 |
| $D_6$ | 25.4 | 27.2 | 47.4 | 75.2 | 100 | 100 | 100 | 100 | 100 | 100 |
| $D_8$ | 45.6 | 98.0 | 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 |
| $A_5$ | 12.5 | 23.1 | 32.5 | 46.7 | 71.2 | 98.8 | 100 | 100 | 100 | 100 |
| $S_5$ | 7.9 | 11.8 | 14.6 | 19.7 | 26.0 | 28.4 | 32.8 | 51.8 | 97.2 | 99.9 |