The Topological Trouble With Transformers

State tracking, feedback paths, and serial computation

Paper by Michael C. Mozer, Shoaib Ahmed Siddiqui, and Rosanne Liu
Google DeepMind

Independent research group discussion

arXiv:2604.17121v5 (September 1, 2026)
Reading: October 19, 2026

A computed state must become available to the next computation.

The question is how a computed state is reused

State tracking

Update a representation as new events arrive: $s_t = f(s_{t-1}, x_t)$.

- Long context provides access to historical information.
- Sequential updates also require access to prior *computed* states.
- The paper links this requirement to the model's computation graph.

Contribution: a structural argument, a recurrence taxonomy, and a research agenda.

Position and synthesis paper; no new controlled benchmark study. Paper, Secs. 1–6, PDF pp. 1–10.

A short dialogue can require persistent constraints

Guess	Reported response	Consistent integer range
60	lower	$[1, 59]$
41	lower	$[1, 40]$
70	higher	$\varnothing$

The task can be solved without choosing a hidden number

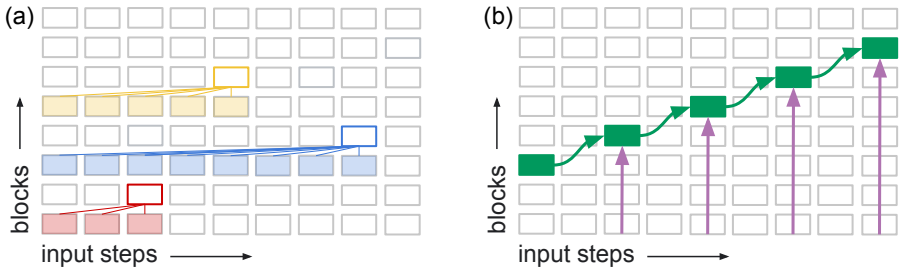
Maintaining a nonempty interval is enough to remain consistent.

- A second trace explicitly selects 42, then misses the guess 42.
- A state written in context must still be used correctly later.

Paper, Sec. 2, PDF p. 2. Gemini 3 Fast / Thinking examples; illustrative traces, not failure-rate estimates.

Direct state-update chains climb the layer stack

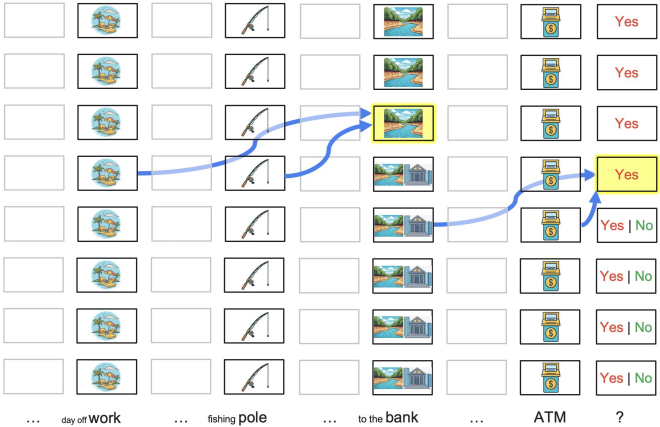
$$h_t^{(\ell)} = F_\ell\big(h_1^{(\ell-1)}, \ldots, h_t^{(\ell-1)}\big)$$



- Horizontal: input positions. Vertical: complete Transformer blocks.
- A result formed at layer ℓ reaches later positions through deeper blocks.

Paper, Fig. 1, PDF p. 3. Equation is explanatory notation; shortcuts can change the computation path.

A correct deep state can arrive too late



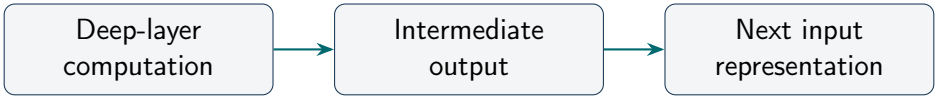
Fishing context resolves *bank* as a river bank.

Later shallow computation can still use the bank–ATM association.

State availability has a timing requirement.

Paper, Sec. 2 and Fig. 3, PDF pp. 4–5; adapted from Lepori et al. (2025). Layer six is schematic.

Intermediate outputs create a feedback path



- CoT exposes intermediate results as tokens.
- Latent thought can feed back continuous representations.
- Both may add serial passes and memory positions.

The authors' efficiency hypothesis

Routine state updates could be handled by implicit activation dynamics.

Teaching schematic based on Paper, Sec. 2, p. 4 and Sec. 4, pp. 8–9. No matched-cost study in this paper.

Layer, input, and execution steps are distinct

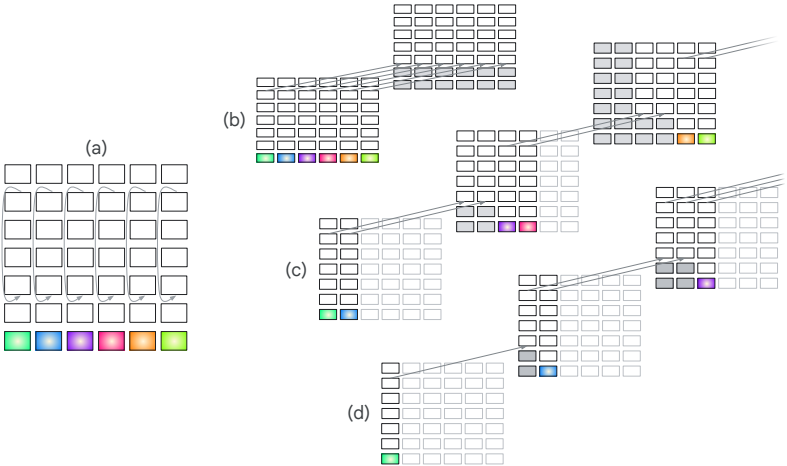
Layer ℓ	Depth within the network; a shared block can be repeated.
Input t	Position in the input sequence; several positions may run together.
Execution r	A feedback update that may precede the next input or block.

Teacher forcing supplies tokens, not computed hidden states

If block b consumes the memory produced by block $b-1$, that dependency remains during training.

Paper, Sec. 3, PDF p. 6. Ordinary token-by-token decoding alone is a different notion of sequentiality.

A feedback arrow needs an execution schedule



(b) Depth loops

(c) Block updates

(d) Continuing historical updates

Gray fill: frozen computation that can be cached.

Paper, Fig. 5, PDF p. 6. Colored inputs are distinct tokens; pale empty boxes are inactive positions.

Fixed depth loops increase a finite compute budget

$$L_{\text{effective}} = KL$$

K repetitions of a group containing L blocks

- Shared weights reduce parameter duplication.
- More repetitions create a longer computation path.
- A fixed K still limits direct dependent updates.

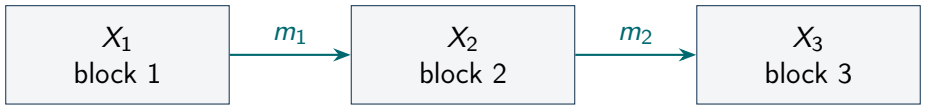
The budget distinction matters

If K grows with input length, depth scaling can solve additional state-tracking tasks.

Paper, Sec. 3, PDF pp. 6–7. Dynamic-depth qualification: Merrill and Sabharwal (2025), Theorem 1.

Block recurrence carries a computed memory forward

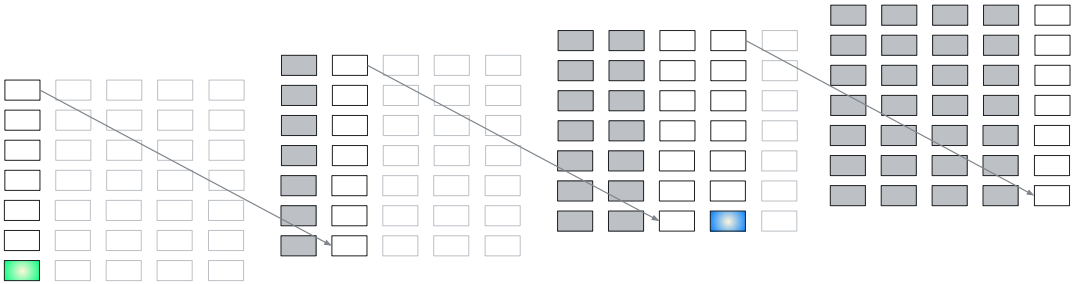
$$(Y_b, m_b) = U(X_b, m_{b-1})$$



- Parallel work inside each block; state dependency between blocks.
- For block size B, approximately $\lceil n/B \rceil$ memory updates.
- Larger blocks reduce update frequency but require more local integration.

Teaching notation and schematic. Paper, Fig. 5c, p. 6; Sec. 5.3, pp. 9–10. Update count is not a speedup claim.

Latent thought can spend multiple updates per input



$$\rho = \frac{\text{new input tokens}}{\text{recurrence updates}} < 1$$

- Intermediate latent outputs feed back before the next external input.
- Persistent tracking depends on which state survives that transition.

Paper, Fig. 6 and Sec. 3, PDF p. 7. The paper notes that not every latent-thought model tracks state.

Continuing feedback may create attractor dynamics

The schedule in Figure 5d

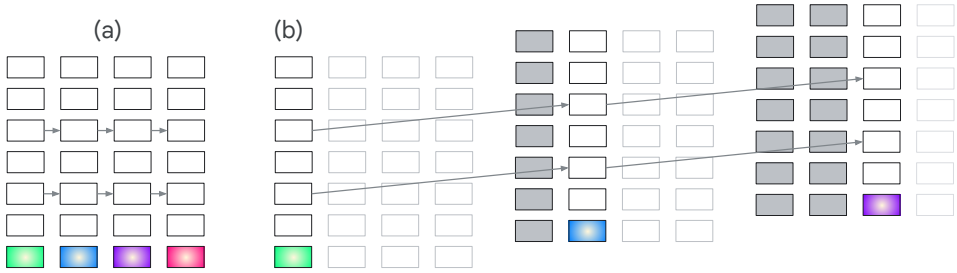
Earlier deep activations feed shallower computation while past positions may continue to update.

- One new input is introduced per execution step in the illustration.
- Historical activations need not all freeze after their first pass.
- Convergence depends on the update dynamics and previous states.

The architecture permits iterative refinement; the paper does not prove convergence.

Paper, Fig. 5d, PDF p. 6; discussion in Sec. 3, p. 7.

Within-layer recurrence changes the state path



$$S_t = A_t S_{t-1} + B_t$$

- State moves directly across input steps at the same layer.
- The transition matrix determines what this recurrent state can do.

Paper, Fig. 7, p. 7; Secs. 4–5.1, pp. 8–9. Equation illustrates an affine state update.

The taxonomy separates path and update frequency

Axis	$\rho > 1$	$\rho = 1$	$\rho < 1$
Depth	Looped / Universal Transformer, RINS	Recirculation	Empty in Table 1
Step	Block-Recurrent Transformers	Linear Attention, DeltaNet, Mamba	DeltaProduct, FSRM
Depth + Step	Recurrent Memory Transformer	Feedback Transformer	COCONUT, CYB

- $\rho > 1$: several input tokens per update.
- $\rho < 1$: several updates before advancing the input.

Selected entries from Paper, Table 1, PDF p. 8. These are the table's labels, not an expressivity ranking.

Transformers often find a cheaper computation

Retrieve an old fact

A latch task can become a lookup into the original input history.

Compose updates in parallel

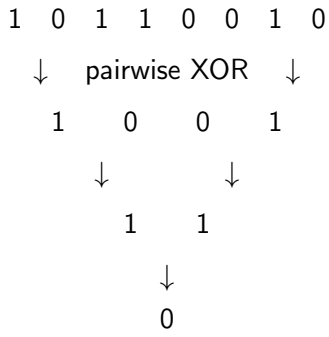
When the operation permits it, merge partial results in a tree.

Factor the state

Independent entity variables can be updated separately and asynchronously.

Paper, Sec. 4, PDF p. 9. These mechanisms explain why a monolithic state-chain picture is incomplete.

Parity illustrates a parallel composition shortcut



Eight inputs, three combination levels: $\log_2 8 = 3$.

Teaching example of the construction in Paper, Sec. 4, PDF p. 9. This is a construction, not a universal lower bound.

Depth bounds depend on the computational model

Dynamic depth

Formal constructions with $\Theta(\log n)$ depth recognize regular languages and solve graph connectivity.

- Fixed-depth separations use standard complexity assumptions.
- Allowing depth to grow differs from holding the loop budget fixed.

Entity tracking, with a local dependency assumption:

$$L_{\min}(G) = \left\lceil \log_2\big(\operatorname{depth}(G) + 1\big) \right\rceil$$

Merrill and Sabharwal (2025), Secs. 2, 4–5; Fagnou et al. (2024), Thm. 1, PDF p. 4. The main paper's $\log n + 1$ paraphrase is inaccurate.

Negative eigenvalues enable qualitatively new updates

$$S_t = \underbrace{(I - \beta_t k_t k_t^\top)}_{A_t} S_{t-1} + \beta_t k_t v_t^\top$$

$$S_t \in \mathbb{R}^{d_k \times d_v},\ k_t \in \mathbb{R}^{d_k},\ v_t \in \mathbb{R}^{d_v}$$

$$\|k_t\|_2 = 1 \quad \Longrightarrow \quad A_t k_t = (1 - \beta_t) k_t$$

- $\beta_t \in [0, 1]$: preserve, attenuate, or erase along the key.
- $1 < \beta_t \leq 2$: the key direction can change sign.
- Expressivity depends on transition structure as well as recurrence.

Paper, Sec. 5.1, PDF p. 9. Formula: Grazzi et al. (2025), Table 1, PDF p. 3. Eigenvalue calculation is explanatory.

State dependence can coexist with parallel training

$$S_t = A_t S_{t-1} + B_t$$

$$(A_2, B_2) \star (A_1, B_1) = (A_2 A_1,\ A_2 B_1 + B_2)$$

- If the coefficients can be computed first, composition is associative.
- A prefix scan uses a tree of combinations across the sequence.
- General nonlinear updates need not have a compact, cheap composition.

Three separate questions

Is there a state? What can it express? How is its update executed?

Teaching derivation clarifying Paper, Sec. 3, p. 7 and Sec. 5.1, p. 9. Scan stages and matrix-operation cost are distinct.

Three directions change the state computation

1. **Enhance transition structure.**
   Broaden SSM updates and combine recurrence with attention.
2. **Improve the representation and objective.**
   Encourage useful state estimates and compositional variables.
3. **Choose a coarser update unit.**
   Pass state at chunk or sentence boundaries.

These are research directions supported by cited work, not a newly evaluated system.

Paper, Secs. 5.1–5.3, PDF pp. 9–10.

Training must make the feedback path affordable

Exploit representational alignment

Residual streams may make cross-layer and cross-step communication easier to introduce.

Introduce recurrence after parallel pretraining

The authors propose staged training, truncated gradients, and recurrent backpropagation for attractor dynamics.

Practical tradeoff: persistent forward state, gradient horizon, and hardware efficiency must be designed together.

Paper, Secs. 5.4–5.5, PDF p. 10. Representational alignment and the staged recipe are proposals here.

Evaluate usable state under a concrete compute budget

1. **Trace the dependency.**
   Identify where a computed state reaches the next update.
2. **Specify the update class and budget.**
   Fixed loops, growing depth, affine scans, and arbitrary recurrence differ.
3. **Measure the tradeoff.**
   Compare long-sequence consistency with latency, memory, and training cost.

Synthesis of Paper, Secs. 2–6. The evaluation recommendation is this discussion's interpretation.

References for the central claims

- Mozer, Siddiqui, Liu. *The Topological Trouble With Transformers*. arXiv:2604.17121v5, 2026. Main paper; all original figures.
- Merrill, Sabharwal. *A Little Depth Goes a Long Way*. arXiv:2503.03961v3, 2025. Dynamic-depth constructions.
- Fagnou et al. *Chain and Causal Attention for Efficient Entity Tracking*. EMNLP 2024. Theorem 1 and its dependency-graph assumption.
- Merrill, Petty, Sabharwal. *The Illusion of State in State-Space Models*. arXiv:2404.08819v3. Scope of SSM expressivity results.
- Grazzi et al. *Unlocking State-Tracking in Linear RNNs Through Negative Eigenvalues*. arXiv:2411.12537v5, 2025.

Backup: the feedback operator remains a design choice

Connection	What it changes
Ordinary KV caching	Reuses past computation at its original layer.
Deep-to-shallow attention	Lets shallow queries read deeper keys and values.
Linear / MLP / adapter	Transforms and injects a feedback representation.
Mutable recurrent state	Updates a persistent representation across events.

An arrow does not specify memory capacity, update dynamics, or a training algorithm.

Paper, Sec. 3, PDF p. 8. Explanatory comparison of the implementation choices listed by the authors.

Backup: read Table 1 with its definition of the ratio

- **Multiple updates per input means $\rho < 1$.**
  The prose on p. 8 uses $\rho > 1$ at one point; the table caption gives the opposite and consistent definition.
- **Figure 5c has inconsistent axis labels.**
  The p. 7 discussion groups it with depth recurrence; Table 1 places it under Depth + Step.
- **A logical recurrence need not ban all parallel algorithms.**
  The strict footnote on p. 7 needs qualification for the parallelizable updates discussed in Sec. 5.1.

Paper, Sec. 3, PDF pp. 7–8 and Sec. 5.1, p. 9. Classifications in this deck follow Table 1.

Backup: the entity-tracking bound has a local assumption

$$L_{\min}(G) = \lceil \log_2(\mathrm{depth}(G) + 1) \rceil$$

- G is the task's computational dependency graph.
- $\mathrm{depth}(G)$ is the length of its longest path.
- Each attention layer is assumed to have receptive field one in that graph.

The assumption controls the claim

This is not an unconditional lower bound on every algorithm that a Transformer could learn for the task.

Fagnou et al. (2024), Theorem 1, PDF p. 4 (printed p. 13177); assumption discussed in Appendix B.