

WHEN DOES RECURRENCE BECOME AN ALGORITHM? CONVERGENCE SELECTION IN WEIGHT-TIED LOOPED TRANSFORMERS

Tong Zhang¹ Junhao Hu² Yun Peng¹ Tao Xie^{2,3,*}

¹Fudan University ²Peking University

³Beijing Tongming Lake Information Technology Application Innovation Center

*Corresponding author: taoxie@pku.edu.cn

ABSTRACT

Weight sharing lets a transformer reuse computation, but it does not explain how training turns repetition into an iterative procedure. We ask whether the loop allowance during training shapes computational progress. We study simple sequence tasks with closed-form target definitions and exact answers at every position. Decoding each intermediate model state often reveals an accurate prefix that advances through the input. We call its leading edge a computation frontier. A budget specifies nominal demand in positions to resolve per loop. **For measurable frontiers in the permutation-product sweep, speed is approximately proportional to this demand.** State interventions in selected models provide additional evidence for dependencies organized around this frontier. An exact fixed point is a state that further updates leave unchanged. We establish why local measurements there cannot uniquely identify the preceding computation. Changing the training tasks or the loop allowance separates learning an operation from reusing it. **Extra loops can advance an unfinished answer, yet can also change one that is already correct.** Measurements at training lengths have task-dependent associations with longer-sequence accuracy. These findings connect training budgets to observable iterative progress and motivate measuring that progress before outputs settle.

1 INTRODUCTION

Looped transformers repeatedly apply a shared neural block to an evolving hidden state (Dehghani et al., 2019; Geiping et al., 2025; Zhu et al., 2025; Saunshi et al., 2025). They can therefore spend more computation on an input without adding parameters. This flexibility is attractive when longer inputs require more work. Its value depends on what the repeated updates accomplish. Extra loops can advance an unfinished computation or revisit an answer that is already available. The training objective must select how the model uses its recurrence.

Architectural recurrence alone does not reveal the learned procedure. A successful final answer also leaves its intermediate computation unobserved. Measurements made after the hidden state stops changing have a related ambiguity. An exact fixed point is a state that the shared update leaves unchanged. The specified local measurements then repeat across later loops, regardless of how the model reached that state. We therefore study the earlier trajectory. Our central question is *how the training loop budget shapes computational progress and how that progress relates to generalization*.

We choose tasks with simple descriptions and closed-form target definitions. Each position has an exact answer that can be checked independently of the model. Prefix parity reports whether each binary prefix contains an odd number of ones. Increment-with-carry adds one to a decimal number presented from its least significant digit. Its outputs are the incremented digits. Group prefix products compose a sequence of operations and report their cumulative composition at each position. All tasks use per-position sequence prediction. This common format exposes where an answer becomes correct and when it changes during recurrence.

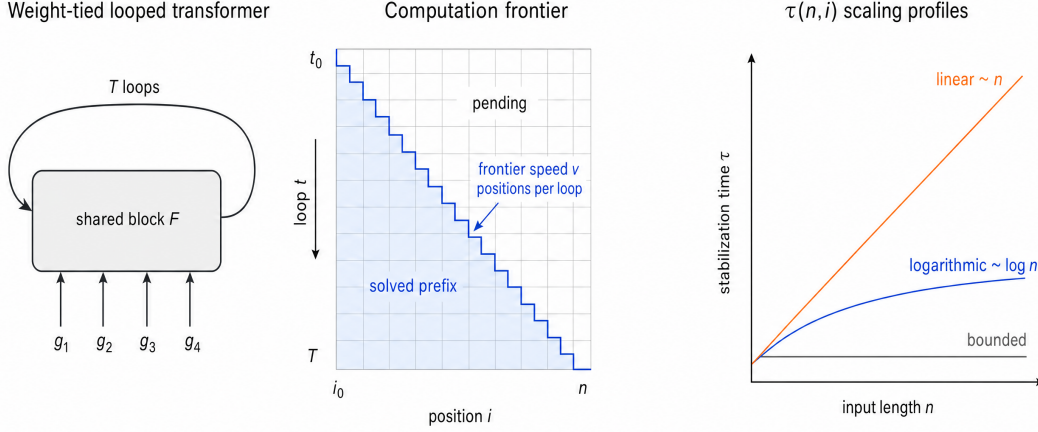


Figure 1: A shared update and its observable progress. The center shows an expanding correct prefix. The right panel illustrates possible stabilization-time profiles as input length grows. All panels are conceptual and contain no measured results. Growth profiles do not uniquely identify algorithms.

The group tasks provide controlled contrasts in the operation being composed. In $\mathbb{Z}_{60}$, composition is addition modulo 60. In S_4 and S_5 , inputs permute four and five objects, respectively. The product is their cumulative reordering. The group A_5 restricts the latter task to even permutations. These are reorderings expressible using an even number of pair swaps. A_5 matches the number of possible states in $\mathbb{Z}_{60}$ while changing the algebraic structure. Addition is commutative, whereas these permutation products generally depend on operation order. Prior analyses give task-specific shortcuts and conditional bounds on circuit depth (Liu et al., 2023; Barrington, 1989; Merrill & Sabharwal, 2025). Appendix G states the computation models and assumptions. Simple target definitions thus provide experimental control without making every task computationally easy.

Several procedures can produce the same target sequence. A model can accumulate operations sequentially. It can also combine partial products in parallel because group composition is associative. Some tasks admit further shortcuts. We do not prescribe one of these procedures through the labels. Instead, we vary the available loops and observe the resulting computation. Controls also vary when inputs become visible, the capacity of the shared block, and the extent of weight sharing. These comparisons distinguish task structure from the constraints imposed by the architecture and training schedule.

Figure 1 summarizes the measurement idea. Our measurements decode an answer from every intermediate state using the same trained readout. Output-stabilization time describes when a prediction settles within the observation window. It records the first observed loop after which the prediction remains equal to its final recorded label. A computation frontier tracks the end of a contiguous prefix that meets a specified accuracy criterion. Its speed describes how quickly this prefix advances through the input. These quantities measure observable progress without assuming that readable answers reveal every internal computation. Replacing selected intermediate states provides a complementary test of which positions and loops influence subsequent predictions.

In the studied S_4 sweep, frontier speed scales approximately proportionally with the nominal training demand. This demand is the inverse of the specified loops per input position. The relation is measured among models with observable frontiers, and its form depends on the task. Curriculum and transfer experiments further distinguish learning an operation from applying it repeatedly under a different loop allowance. **The association between progress at training lengths and longer-sequence accuracy depends on the task.** We make the following contributions.

1. **A framework for measuring iterative progress.** We define output stabilization and frontier motion. We establish what the specified local measurements cannot reveal at an exact fixed point. Input-release controls and state interventions help interpret the resulting trajectories (§3, §4).

2. **An empirical connection between training budget and computational pace.** We characterize the relation between demand and speed in the studied permutation task. Comparisons across tasks, model capacities, and weight-sharing patterns establish its observed scope (§5).
3. **A comparison of learning, adaptation, and generalization.** Curricula and budget transfer separate learning the operation from reusing it. Measurements at training lengths reveal task-dependent associations with accuracy on longer inputs (§6, §7).

2 SETUP

Tasks with explicit targets. We use sequence tasks whose answer is known at every position. For an input of length n , position $i \in \{1, \dots, n\}$ has target y_i . This lets us score intermediate predictions without fitting a separate probe. The tasks have compact closed-form specifications. They remain informative because the same targets can be computed by different procedures.

Our main task is a group prefix product. A group G is a set of invertible operations with an associative composition $\circ$ and an identity operation 1_G . Given operations $g_1, \dots, g_n$, the target at each position is their cumulative composition,

$$p_0 = 1_G, \quad y_i = p_i = g_1 \circ \dots \circ g_i = p_{i-1} \circ g_i. \quad (1)$$

Associativity allows a model to combine subproducts in parallel. The recursive form also admits a serial accumulator. The target therefore specifies what to compute without prescribing how many loops the model should use.

A prefix as an observable subproblem. Consider a group input whose prefix ends at position i . Its target p_i summarizes every operation seen up to that position. The next target is $p_i \circ g_{i+1}$. A serial procedure could carry this product forward and apply the next operation. An associative procedure could instead combine products of separate input segments. Both procedures produce the same supervised answers. The model is free to learn another representation of these dependencies.

This makes intermediate decoding useful without treating it as a complete trace of the internal algorithm. If a prefix is correct at a recorded loop, its answers are already available through the trained readout. Later positions can remain incorrect at that same loop. A subsequent update may extend the correct prefix or change an earlier answer. The schematic in Figure 1 illustrates the extending case. Recording the trajectory reveals which behavior occurs in a model. State interventions then ask whether the readable progress corresponds to consequential intermediate states.

The task family varies the algebra while retaining this interface. In $\mathbb{Z}_{60}$, composition is addition modulo 60. In S_4 and S_5 , inputs are permutations of four and five objects. Their targets are cumulative reorderings. In A_5 , inputs are the even permutations of five objects. These groups distinguish properties that are otherwise easy to conflate. Commutativity means that changing the order of operations leaves the product unchanged. $\mathbb{Z}_{60}$ is commutative, whereas the permutation groups are not. Solvability is a different algebraic property, defined in Appendix G. S_4 is solvable, whereas S_5 and A_5 are not. Finally, A_5 and $\mathbb{Z}_{60}$ have equally many elements. Their comparison controls the number of possible input and output symbols while changing the algebra.

We also use prefix parity and decimal increment to calibrate the measurements. Prefix parity outputs whether the number of ones seen so far is odd. For input bits b_j , its target is $y_i = (\sum_{j=1}^i b_j) \bmod 2$. Decimal increment adds one to a digit string presented from the least significant digit to the most significant. The output digit changes only when every preceding digit is nine. This gives a direct target formula and a familiar carry-propagation procedure. Both tasks allow shortcuts, so neither forces the model to follow the familiar sequential procedure. Exact generators and algebraic definitions are given in Appendix G.

Shared recurrent computation. We prepend a beginning-of-sequence token, abbreviated BOS, at position $i = 0$. The model therefore receives $N = n + 1$ tokens. Each token is mapped to a learned vector, giving the input embedding $e \in \mathbb{R}^{N \times d}$, where d is the hidden width. The hidden state $h^t \in \mathbb{R}^{N \times d}$ records the representation after t completed loop updates. In the free-input setting, the initial state h^0 is a learned scalar multiple of e . A loop applies a transformer module F_θ with parameters θ . The module contains attention and feedforward layers. The same parameters are

reused at every loop. A trained linear readout W maps each normalized state to class scores,

$$h^{t+1} = F_\theta(h^t, e), \quad \hat{y}_i^t = \arg \max_c [W \text{LN}(h_i^t)]_c. \quad (2)$$

Here LN denotes layer normalization and c indexes the possible output classes. The readout is shared across recorded loops and is trained on the task itself. Intermediate predictions therefore use the model’s own output map. Some configurations also reintroduce e through a learned adapter at each loop, a procedure called input injection (Bansal et al., 2022). Attention is causal, so position i can use preceding positions but cannot access later ones. Architecture sizes and positional encoding controls appear in Appendix G.

Budgets and input availability. The loop budget is the number of recurrent updates allowed for a sequence. We distinguish the training allowance T_{tr} from the observation horizon T used to record an evaluation trajectory. For training length n_{tr} , the ratio $n_{\text{tr}}/T_{\text{tr}}$ measures the realized demand in positions per allowed loop. The archived proportional-budget fit uses nominal demand, the reciprocal of the specified loops per position. Integer rounding can make these ratios differ. Higher demand leaves less time for each position. We compare budgets that are fixed, grow logarithmically with length, or grow in proportion to length. This tests how the training allowance shapes the learned procedure (Li et al., 2020; Graves, 2016; Banino et al., 2021). The experiments use a length curriculum, which gradually introduces longer sequences (Merrill & Sabharwal, 2025).

In the main setting, all input tokens are available from the start. We call this free input availability. A streaming control reveals tokens in sequence as loops proceed. Streaming enforces sequential access to information, while free inputs permit parallel computation. The control therefore supplies a reference for delayed information arrival. It does not prescribe a unique internal algorithm. Appendix G specifies the schedules and release rule.

3 WHAT FIXED-POINT MEASUREMENTS CAN ESTABLISH

The same final answer can be reached by different computations. To understand a repeated update, we must therefore distinguish its progress from its eventual stationary behavior. With free input availability, write $\Phi(h) = F_\theta(h, e)$ for the fixed update map. A fixed point is a state h^* satisfying $\Phi(h^*) = h^*$.

Proposition 1 (Tail blindness, informal). Suppose the trajectory reaches an exact fixed point and the update map is differentiable there. Removing subsequent updates leaves the state unchanged. Local derivatives of the update also coincide at subsequent loops. A deterministic measurement based only on that state and its local derivative has the same reading at each such loop. These readings do not uniquely identify the preceding computation.

The local derivative, or Jacobian, describes how a small perturbation to the current state affects the next update. Once the state is fixed, this derivative is evaluated at the same point each time. This explains the proposition without assuming any particular training method. Appendix E gives the formal statement, proof, and bounds for approximate fixed points.

At an exact fixed point, the specified local measurements do not uniquely identify the preceding computation. They can describe the final state and its local response to a perturbation. Our question instead concerns the progress that led to that state. We therefore read the model throughout the trajectory. The proposition is conditional on reaching a fixed state. An unchanged predicted label alone does not establish this condition.

4 THE OUTPUT-STABILIZATION INSTRUMENT

When each answer settles. We decode every supervised position after every completed update. For a chosen observation horizon T , the stabilization time at position i is the earliest recorded update after which its predicted label remains equal to the final recorded label,

$$\tau(n, i) = \min\{k \in \{1, \dots, T\} \mid \hat{y}_i^t = \hat{y}_i^T \text{ for every } t \in \{k, \dots, T\}\}. \quad (3)$$

We summarize each input by its latest supervised stabilization time, $\max_{1 \leq i \leq n} \tau(n, i)$. The reported sequence-level statistic is the median over inputs whose entire final output is correct, denoted

$\tau_{\max}(n)$. This condition separates the timing of a correct answer from the ability to solve the task. Accuracy is retained separately for all evaluated inputs. The unsupervised BOS position is excluded.

The resulting profile shows where predicted answers settle as computation proceeds. A profile that moves steadily along the sequence is consistent with sequential progress. A profile that changes more slowly with sequence length can indicate a different organization of the computation. These are finite-range observations. They do not by themselves classify asymptotic computational complexity. The streaming control helps interpret the profiles because later answers cannot use information before it arrives. Appendix F reports calibration and eligibility rules.

How far the solved prefix advances. Some models make reliable progress without solving the whole input. For these cases, we measure a computation frontier. At loop t , let $c_t(n)$ be the length of the initial contiguous prefix whose positions each meet a fixed batch-accuracy threshold. The frontier thus records how far reliable answers extend along the sequence. A frontier rate v summarizes the number of positions gained per loop. Prefix completion is the fraction $c_T(n)/n$ at the chosen horizon. In the prediction comparison, completion uses the native allowance assigned by each model’s trained loop schedule. In the archived budget analysis, frontier speed is an endpoint rate,

$$v = \frac{c_b(n) - c_a(n)}{b - a}, \quad (4)$$

where a precedes the first positive frontier increment and b marks the last positive increment. The rate is defined only when there are enough increments to measure progress. We inspect the full trajectory alongside this summary because the endpoint rate can span stalls or regressions.

Keeping measurement conventions distinct. Stabilization concerns decoded labels, which can stop changing while hidden states continue to evolve. Both stabilization and frontier rates also depend on the observation horizon. A later label change can alter the measured stabilization time, and additional loops can alter the endpoint used for the rate. Appendix F records the archived timing convention separately from Eq. 3. Appendix G distinguishes the archived endpoint rate from the first-passage rate used in the corrected prediction analysis. The latter stops at the first attainment of the largest observed prefix. Each empirical summary retains its own estimator and window. This keeps an observed difference between protocols separate from a difference between learned computations.

4.1 READING THE EXPERIMENTAL COMPARISONS

Competence and progress. We distinguish correctness of a whole answer from correctness of its parts. Exact-match accuracy is the fraction of inputs for which every supervised output is correct. Per-token accuracy is the fraction of supervised positions predicted correctly across the evaluated inputs. These scores answer different questions. A model can get most positions right while making an error on nearly every sequence. Exact match captures completion of the task. Per-token accuracy retains information about partial success. Neither score alone describes when the correct predictions appeared.

The trajectory measurements add temporal and spatial information. Stabilization summarizes final-solved inputs, while accuracy includes both solved and unsolved inputs. Frontier estimation uses accuracy across a batch at each position. It can therefore describe a reliable prefix even when few entire sequences are correct. Prefix completion is a fraction of positions passing this batch criterion. It is not the fraction of fully solved inputs. A correct region beyond an unreliable position also does not extend the contiguous frontier. This convention keeps progress along the sequence distinct from isolated correct predictions.

Training pressure and available computation. Changing the allowance during training asks a different question from adding loops during evaluation. The budget sweep compares models trained with different allowances. Evaluation loop sweeps instead keep a trained checkpoint fixed and vary how long it executes. Budget transfer occupies an intermediate role. It starts from an existing checkpoint and continues optimization with a changed allowance. These comparisons distinguish the dynamics learned during training from the usefulness of extra computation after training. They

also distinguish adaptation of an existing model from learning under a schedule from initialization. Recipe and curriculum differences within the archived sweep are recorded in Appendix G.1.

Capacity and sharing. The capacity comparisons change the amount of processing available inside each recurrent update. Increasing block depth adds successive transformations before the next loop begins. Increasing width expands the representation carried through those transformations. A loop count therefore has to be interpreted together with the shared module that it repeats. The tying comparison asks a separate question about reuse. It holds total executed depth fixed while changing how many distinct parameter groups occupy that depth. Keeping width fixed then changes the parameter count. Matching parameters instead requires changing width. We report these designs separately because they hold different resources fixed. The capacity cohorts also retain their recorded optimization settings.

Learning an operation and composing it. The curriculum comparison changes the allocation of training across chain lengths. Pair composition presents the local operation directly. Longer sequences require repeated composition of the same operation. Allocating more training to the initial pair task tests whether a stronger starting point helps learn longer chains. It changes exposure to the local operation as well as the training schedule. We therefore compare local learning, competence on trained lengths, and competence on longer lengths as separate outcomes. The exact stage allocations are given in Appendix G.

Information availability. Streaming changes which inputs can contribute to an update. The target sequence remains the same, but unrevealed tokens have no available embedding and cannot be attended to. A delayed answer may then reflect delayed information rather than slow processing of an already visible input. Comparing this control with free inputs helps interpret output timing. Intermediate supervision is a further choice. It trains revealed positions during the trajectory, whereas final supervision trains predictions at the selected endpoint. The streaming comparisons keep these supervision choices explicit.

Generalization as a separate outcome. An in-distribution measurement uses a length within the task’s training range. The corresponding generalization outcome uses longer inputs. Each model receives the allowance prescribed by its own schedule at those lengths. The prediction comparison asks whether measurements available at training lengths order models in a way that agrees with their later accuracy. It requires both measurements for the same model. Models missing either value are omitted from that association calculation. This comparison concerns differences between models, whereas the frontier describes progress within a model. Keeping these units distinct prevents a fast local computation from being equated with reliable extrapolation.

5 THE LINEAR COMPUTATION FRONTIER

We first ask whether repeated updates reveal a coherent pattern of progress. The decoded prefix provides a direct answer for many of the successful group-task models. Early positions become correct before later positions. The boundary between these regions advances across successive loops. **Many analyzed models resolve a prefix at an approximately constant speed across the measured input lengths.** The trajectories and their recorded rates appear in Appendix Figure 3 and Table 1. This frontier describes observable progress without assigning a unique internal implementation.

The frontier also shows how additional loops affect incomplete computation. At a fixed input length, budget-starved models often fail first at later positions. Allowing more updates can extend the correct prefix into those positions. Continued iteration can eventually change answers that were already correct. The loop sweeps in Appendix Table 10 show both behaviors. A useful account of test-time computation must therefore track where progress occurs and when it ceases to help.

5.1 BUDGET DEPENDENCE

Training demand measures how many positions each allowed loop must resolve on average. We vary the proportional loop allowance and ask whether this pressure changes the learned frontier speed. For this comparison, the demand is the nominal reciprocal of the budget multiplier. The

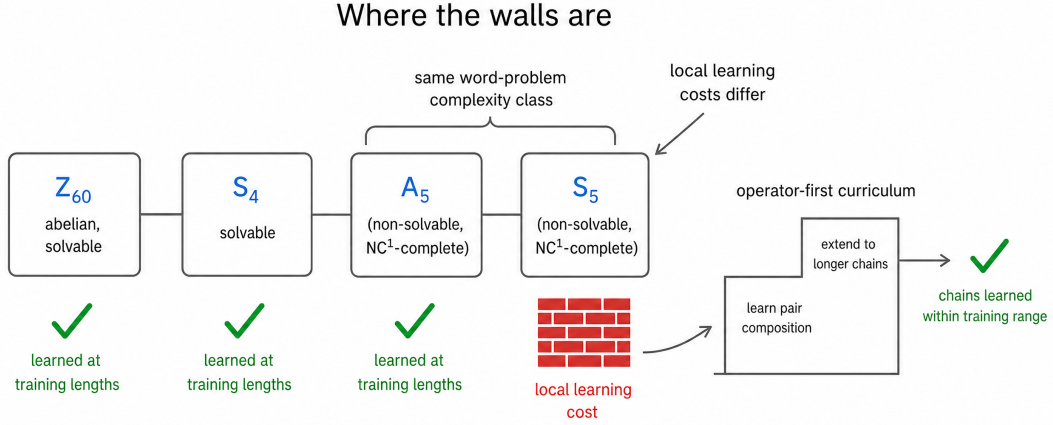


Figure 2: Learning a local operation and applying it along a chain. The group tasks differ in algebraic structure and local learning cost. The bracket refers to their fixed-group word problems. The curriculum emphasizes pair composition before extending to longer chains. Check marks indicate learning within the training range.

measured rate summarizes the recorded advance of the solved prefix. Appendix Table 11 gives the fitted population and the exact aggregation protocol.

In the measurable S_4 population, frontier speed grows approximately in proportion to training demand. A tighter allowance is associated with faster progress per loop. A looser allowance admits slower frontiers. This relationship concerns models with a measurable frontier under the archived observation window. Training recipes also differ across parts of the sweep. The result supports a relation between training pressure and observable dynamics within this population.

The schedule matters as well as its average demand. A logarithmic allowance becomes progressively tighter as the training length grows. Its learned speed can fall short of the demand at the longest stage. A proportional allowance instead maintains a consistent pressure across stages. Cross-task controls show that the response also depends on the operation being composed. The corresponding patterns on A_5 , parity, and Z_{60} do not establish the same precise scaling relation. These comparisons make budget selection an empirical property of a task and training recipe.

5.2 CAPACITY AND CAUSAL CHECKS

A fast frontier requires useful composition within each shared update. We therefore vary block depth and width under a demanding loop allowance. Deeper blocks advance farther than the shallowest block in the tested S_4 comparison. The wider cohort does not have a higher median speed. Other training settings change with that cohort, so the comparison does not isolate width. Appendix Table 12 records the settings and eligibility criteria. **Block depth can expand the observed progress per loop, but capacity alone does not order the outcomes across tasks.** The non-monotone A_5 depth results show that optimization remains part of this relationship.

To test whether the decoded frontier reflects consequential hidden states, we replace an intermediate state with a state from another input. We then continue the same recurrence and measure damage to the final answer. This intervention tests dependencies that a readout alone cannot establish. The causal attention mask already prevents effects on earlier positions. The informative observation concerns the organization of downstream damage.

In the selected S_4 models, downstream damage follows a moving boundary consistent with the decoded frontier. The effect of replacement depends on where and when the state is changed. Appendix Figure 5 shows these intervention maps. Recovery is a separate property. Some tested models largely restore their final answers after particular replacements, while others retain the damage. Appendix Table 6 reports this final-readout diagnostic. The interventions support frontier-organized dependencies in the selected models. They do not turn every observed frontier into a validated circuit.

5.3 LEARNING THE OPERATOR AND REUSING IT

A prefix product combines a local composition rule with its repeated application along the input. We separate these demands with an operator-first curriculum (Figure 2). The model first spends longer learning pair composition. Training then extends to longer chains. This choice is motivated by the different optimization costs of learning the tested operators in isolation (Appendix Table 5).

Operator-first training learns S_5 chains within the training range without establishing length generalization. Its observed training success is higher than that of the compared free-training cohort. The curricula also differ in their step budgets, so this comparison does not isolate curriculum order. On longer inputs, the learned frontier can advance into the sequence while exact-match accuracy remains low. Appendix Table 8 and Table 10 separate these outcomes.

This separation matters for interpreting algorithmic behavior. Learning the composition rule is a local achievement. Applying it across trained chains is a further achievement. Maintaining a complete solution on longer chains places an additional demand on the learned dynamics. A moving frontier can reveal useful partial computation even when the final sequence is incorrect. Frontier speed therefore complements competence measurements and cannot replace them.

5.4 STREAMING INPUT CONTROL

The free-input setting exposes the whole sequence before the first update. A model can therefore use parallel information or a shortcut. Streaming instead reveals successive tokens over successive loops. Later outputs must wait for the information they require. This provides a reference with an explicit constraint on when an answer can become available.

The streaming control links output timing to information arrival, while its training outcomes remain task dependent. Parity learns the training-length task under this release schedule. The tested difficult group tasks do not reach comparable competence, including with supervision on intermediate prefixes. Appendix Table 8 reports the populations and outcomes. Streaming changes input availability and attention masking together. It calibrates a release-limited timing profile without showing that free-input models use the same computation.

5.5 WEIGHT-TYING CONTROLS

We next ask whether the findings depend on reusing weights. The control holds effective depth fixed and varies the number of distinct weight groups. Full tying reuses a common group throughout. Partial tying cycles through several groups. The fully untied model uses distinct weights at each depth. We compare a fixed-width ladder with a parameter-matched ladder, which reduces width as more groups are introduced.

The effect of weight reuse depends on the parameter constraint, and partial tying can outperform full tying. Untying is mildly beneficial in parts of the fixed-width comparison. With parameters matched, the fully untied models extrapolate less successfully than some tied configurations. Appendix Table 9 contains the complete ladder. The results support weight reuse as a useful bias in these settings while leaving the best degree of sharing task dependent. Learned-position controls also retain observable frontiers and extra-loop rescue within their supported position range. These observations locate the behavior more precisely than an architectural claim about recurrence alone.

6 ANNEALING THE BUDGET CHANGES FRONTIER DYNAMICS

The budget comparison above trains separate models. We next test whether an existing model changes its dynamics after the allowance changes. Budget transfer starts from a trained checkpoint and continues training under a different loop schedule. This intervention keeps a learned starting point while changing the pressure on each subsequent update.

A changed budget can alter frontier speed without producing a complete solution under the new allowance. Transfer to a tighter schedule accelerates the measured frontier in the tested group models. The accelerated prefix still falls short of the full sequence at the target budget. Transfer in the reverse direction produces successful training-length solutions in the evaluated S_4 population.

Saved trajectories in that direction also show slower measured frontiers. Appendix Table 2 gives both outcomes and the available trajectory coverage.

The distinction is between adapting an existing partial computation and solving the target task. A faster rate can be a meaningful response to training even when it is insufficient for completion. Conversely, relaxing the allowance can support a complete answer without preserving the original speed. These observations motivate measuring the trajectory again after transfer instead of assuming that a checkpoint retains a fixed computational pace.

6.1 A CANDIDATE STOPPING RULE

A roughly constant frontier speed suggests a practical allocation rule. Estimate speed at training lengths and allow enough loops to cover the requested sequence. We add slack to accommodate variation around that estimate. This gives the candidate budget $T^* = \lceil n/\hat{v} \rceil + s$, where $\hat{v}$ is the estimated speed and s is the slack.

Frontier speed can guide a test-time loop allowance within a model’s observed operating range.

The archived comparison evaluates this rule against stored schedule references and the best accuracy observed in a finite loop sweep (Appendix Figure 4). The rule can improve on the reference for selected models. Its usefulness depends on whether the frontier continues to advance. Stalling can make the estimated budget too short. Changes after completion can make a larger allowance harmful. The rule allocates a sequence-level budget from prior measurements. It does not detect convergence separately for each input.

7 PREDICTING GENERALIZATION FROM IN-DISTRIBUTION MEASUREMENTS

We finally ask whether measurements at training lengths contain information about performance on longer inputs. The comparison uses frontier speed and prefix completion as progress measurements. It also includes adjacent-state cosine similarity, alignment between sampled local Jacobian actions, and agreement between outputs reached from different initial states. These baselines describe state similarity or initialization dependence. They do not themselves establish a hidden-state fixed point.

All predictors use lengths within each task’s training range. Generalization is the average per-token accuracy at the evaluated longer lengths. The corrected comparison uses a common observation window for frontier speed and each model’s native allowance for completion. Its speed estimator therefore differs from the archived budget-fit estimator. Appendix Table 3 specifies these protocols and restricts each association to available paired measurements.

The association between trajectory measurements and length generalization depends on the task.

Frontier speed and completion associate positively with longer-input accuracy in the evaluated group-task pairs. Parity shows a different pattern. Similarity measurements also carry information, including an association whose direction reverses across the group-task settings. The comparison therefore does not establish uniform superiority of progress measurements. Choosing a score’s direction after seeing the longer-input outcomes would overstate its predictive value.

These are associations within the observed populations. A deployed predictor would need its score direction and selection rule fixed before evaluating new models. The external prefix-sums check adds a complementary accuracy test. Successful models extrapolate over a limited length range, but that experiment has no saved frontier-speed measurement. Appendix B reports it as evidence about competence rather than a further test of the budget relation.

8 CONCLUSION

Training budgets are linked to observable iterative progress in the studied looped transformers.

The clearest relation is the approximately proportional response of measurable S_4 frontier speed to training demand. State interventions support frontier-organized dependencies in selected models. Curriculum and transfer studies distinguish this progress from learning a local operator and completing a task. Together, these findings motivate reading the evolving computation before allocating further loops. They also keep length generalization as a separate outcome to measure. Appendix H discusses the scope of this interpretation.

AI USE STATEMENT

Generative AI assistants were used to refine the research framing and hypotheses; provide feedback on experimental methodology; implement and revise training, evaluation, and analysis code; help formulate and check mathematical statements and proof text; interpret experimental results; and draft and edit the manuscript. They also assisted with literature search, experimental parameters, conceptual illustrations, figure preparation, and experiment orchestration. Synthetic task instances and their labels are produced by explicit arithmetic and group-composition programs. Reported empirical values are computed from executed experiments and stored artifacts. Verification included CPU regression tests, native forward and sham-intervention controls, checkpoint and source hashes, independent recomputation of the reported statistics, and separate code-to-evidence and manuscript-to-evidence reviews. The supplementary artifacts distinguish the scope of each verification from the experimental claims. The authors are responsible for the final text, claims, code, and results, including all AI-assisted contributions.

ETHICS STATEMENT

The experiments concern synthetic prefix-computation tasks and an existing algorithmic prefix-sum benchmark. They do not involve human participants or personal data. The released summaries identify their conditioning populations and retain the relevant model denominators. The findings concern the tested model scales, tasks and finite lengths; they do not establish performance or safety guarantees for deployed language models. Third-party datasets and software retain their original distribution terms.

REPRODUCIBILITY STATEMENT

Section 2 specifies the models, tasks and training budgets; Section 4 distinguishes the recorded timing and frontier estimators. Appendix G gives training settings and cohort membership, Appendix F specifies calibration and measurement conventions, and Appendix E states the assumptions and proofs. The supplementary archive separates the original recorded evaluations from subsequent corrected-protocol checks. It contains compact numerical evidence, source hashes, executable measurement code and verification scripts. Full checkpoints, individual prediction tensors and the external benchmark dataset are not bundled. Historical and corrected observations are labeled by their estimator, window and conditioning population rather than silently pooled.

REFERENCES

- Emmanuel Ameisen, Jack Lindsey, Adam Pearce, et al. Circuit tracing: Revealing computational graphs in language models. transformer-circuits.pub, 2025.
- Cem Anil, Ashwini Pople, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J. Zico Kolter, and Roger Grosse. Path independent equilibrium models can better exploit test-time computation. *NeurIPS*, 2022a.
- Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. *NeurIPS*, 2022b.
- Artur Back de Luca and Kimon Fountoulakis. Simulation of graph algorithms with looped transformers. *ICML*, 2024.
- Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoung Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *NeurIPS*, 2025. arXiv:2507.10524.
- Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *NeurIPS*, 2019.
- Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. In *ICML Workshop on Automated Machine Learning*, 2021. arXiv:2107.05407.

- Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Logical extrapolation without overthinking. *NeurIPS*, 2022.
- David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1 . *Journal of Computer and System Sciences*, 38(1):150–164, 1989.
- David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1 . *Journal of the ACM*, 35(4):941–952, 1988.
- Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.
- Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the ability and limitations of transformers to recognize formal languages. *EMNLP*, 2020.
- Hugh Blayney, Álvaro Arroyo, Johan Obando-Ceron, Pablo Samuel Castro, Aaron Courville, Michael M. Bronstein, and Xiaowen Dong. A mechanistic analysis of looped reasoning language models. *arXiv preprint arXiv:2604.11791*, 2026.
- Jannik Brinkmann, Abhay Sheshadri, Victor Levoso, Paul Swoboda, and Christian Bartelt. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. *Findings of ACL*, 2024.
- Vivien Cabannes, Charles Arnal, Wassim Bouaziz, Alice Yang, Francois Charton, and Julia Kempe. Iteration head: A mechanistic study of chain-of-thought. *NeurIPS*, 2024.
- David Chiang and Peter Cholak. Overcoming a theoretical limitation of self-attention. *ACL*, 2022.
- Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. *ICML*, 2023.
- Mark M. Churchland, John P. Cunningham, Matthew T. Kaufman, Justin D. Foster, Paul Nuyujukian, Stephen I. Ryu, and Krishna V. Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012. doi: 10.1038/nature11129.
- Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *NeurIPS*, 2023.
- Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber. The neural data router: Adaptive control flow in transformers improves systematic generalization. *ICLR*, 2022.
- Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D. Manning. MoEUT: Mixture-of-experts universal transformers. *NeurIPS*, 2024.
- Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *ICLR*, 2019.
- Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. Neural networks and the Chomsky hierarchy. *ICLR*, 2023.
- Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, et al. Faith and fate: Limits of transformers on compositionality. *NeurIPS*, 2023.
- Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *ICLR*, 2020.
- Nelson Elhage, Neel Nanda, Catherine Olsson, et al. A mathematical framework for transformer circuits. *transformer-circuits.pub*, 2021.

- Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. *arXiv preprint arXiv:2409.15647*, 2024.
- Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. *NeurIPS*, 2023.
- Khashayar Gatmiry, Nikunj Saunshi, Sashank J. Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? *ICML*, 2024.
- Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. *NeurIPS*, 2021.
- Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kaillkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.
- Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. *ICML*, 2023.
- Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *ICLR*, 2024.
- Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.
- Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *TACL*, 2020.
- Michael Hahn and Mark Rofin. Why are sensitive functions hard for transformers? *ACL*, 2024. *arXiv:2402.09963*.
- Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *NeurIPS*, 2023.
- Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.
- Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *NeurIPS*, 2023.
- DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. Block-recurrent transformers. *NeurIPS*, 2022.
- Borja Ibarz, Vitaly Kurin, George Papamakarios, Kyriacos Nikiforou, Mehdi Bennani, Róbert Csordás, Andrew Dudzik, Matko Bošnjak, Alex Vitvitskyi, Yulia Rubanova, et al. A generalist neural algorithmic learner. *Learning on Graphs Conference (LoG)*, 2022.
- Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.
- Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. *ICLR*, 2016.
- Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *NeurIPS*, 2023.
- Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. *arXiv preprint arXiv:2604.07822*, 2026.
- Kenneth Krohn and John Rhodes. Algebraic theory of machines. I. prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116: 450–464, 1965.

- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *ICLR*, 2020.
- Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *ICLR*, 2024.
- Mengtian Li, Ersin Yumer, and Deva Ramanan. Budgeted training: Rethinking deep neural network training under resource constraints. In *ICLR*, 2020.
- Nuo Li, Kayvon Daie, Karel Svoboda, and Shaul Druckmann. Robust neuronal dynamics in premotor cortex during motor planning. *Nature*, 532(7600):459–464, 2016. doi: 10.1038/nature17643.
- Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *ICLR*, 2024.
- David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: Compiled transformers as a laboratory for interpretability. *NeurIPS*, 2023.
- Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *ICLR*, 2023.
- Wenquan Lu, Yuechuan Yang, Kyle Lee, Yanshu Li, and Enqi Liu. Latent chain-of-thought? decoding the depth-recurrent transformer. *COLM Workshop on LLM Explainability for Reasoning and Planning*, 2025. arXiv:2507.02199.
- Valerio Mante, David Sussillo, Krishna V. Shenoy, and William T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, 2013. doi: 10.1038/nature12742.
- Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kaikhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. Transformers can do arithmetic with the right embeddings. *NeurIPS*, 2024.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. *NeurIPS*, 2022.
- William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *TACL*, 2023.
- William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. *ICLR*, 2024.
- William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. *NeurIPS*, 2025. arXiv:2503.03961.
- William Merrill, Ashish Sabharwal, and Noah A. Smith. Saturated transformers are constant-depth threshold circuits. *TACL*, 2022.
- William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. *ICML*, 2024. arXiv:2404.08819.
- Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *ICLR*, 2023.
- nostalgebraist. Interpreting GPT: The logit lens. LessWrong, 2020.
- Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.
- Catherine Olsson, Nelson Elhage, Neel Nanda, et al. In-context learning and induction heads. *transformer-circuits.pub*, 2022.

- Jacob Pfau, William Merrill, and Samuel R. Bowman. Let’s think dot by dot: Hidden computation in transformer language models. *COLM*, 2024.
- Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.
- Gleb Rodionov and Liudmila Prokhorenkova. Neural algorithmic reasoning without intermediate supervision. *NeurIPS*, 2023.
- Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Transformers, parallel computation, and logarithmic depth. *ICML*, 2024.
- Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. *ICLR*, 2025.
- Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *NeurIPS*, 2022.
- Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. *NeurIPS*, 2021.
- Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. *ICLR*, 2025.
- Dashiell Stander, Qinan Yu, Honglu Fan, and Stella Biderman. Grokking group multiplication with cosets. *ICML*, 2024.
- Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? a survey. *TACL*, 2024.
- Shawn Tan, Yikang Shen, Zhenfang Chen, Aaron Courville, and Chuang Gan. Sparse universal transformer. *EMNLP*, 2023.
- Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. The clrs algorithmic reasoning benchmark. *ICML*, 2022.
- Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. *NeurIPS*, 2020.
- Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization. *NeurIPS*, 2024.
- Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.
- Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. *ICLR*, 2023.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 2022.
- Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. *ICML*, 2021.
- Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. *arXiv preprint arXiv:2311.12424*, 2023.
- Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. Knowledge circuits in pretrained transformers. *NeurIPS*, 2024.

Qifan Yu, Zhenyu He, Sijie Li, Xun Zhou, Jun Zhang, Jingjing Xu, and Di He. Enhancing auto-regressive chain-of-thought through loop-aligned reasoning. *EACL*, 2026. arXiv:2502.08482.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. *NeurIPS*, 2023.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. *ICLR*, 2024.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

A EMPIRICAL EVIDENCE

A.1 FRONTIER SPEEDS AND BUDGET DEPENDENCE

Table 1: Frontier rates by training budget. Each model’s rate is the standard median over available $n \in \{16, 32, 64\}$; the table then takes the standard median across models [range]. Models are rate-measurable/attempted; A_5 additionally requires exact match $> .5$ at $n = 32, T = 32$. The last column is the median final solved-prefix fraction at $n = 64$ over the stated cohort, including models without a rate, at $T = 64$ (proportional) or $T = 22$ (log). A rate need not describe linear motion on $\mathbb{Z}_{60}$. Cohort membership is in Appendix G.1.

Group	Budget (demand n/T)	models	rate	frac.@64
S_4 S	$T = 2n$ (0.5)	3/4	0.51 [0.51, 0.52]	0.54
S_4 S	$T = n$ (1)	2/2	1.00 [1.00, 1.00]	0.71
S_4 S	$T = n/2$ (2)	4/4	2.00 [1.93, 2.13]	0.10
S_4 S	$T = n/4$ (4)	2/2	3.80 [3.60, 4.00]	0.00
S_4 S	log- T (mixed)	16/16	2.00 [1.86, 2.42]	0.30
S_4 M	log- T (mixed)	4/4	1.88 [1.86, 1.88]	0.36
S_4 L	log- T (mixed)	4/4	2.66 [2.09, 3.33]	0.07
A_5 S	$T = n$, horizon	5/16	1.00 [0.86, 1.03]	n/a
S_5 M	op.-first, $T = n$	2/2	1.00 [1.00, 1.00]	0.28
$\mathbb{Z}_{60}$ S	log- T (mixed)	8/8	2.08 [1.82, 5.17]	n/a

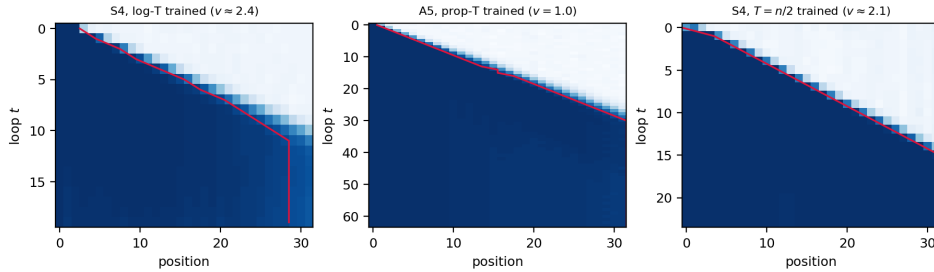


Figure 3: Per-(loop, position) accuracy heatmaps at $n=32$ (blue. Solved; red line. Contiguous solved prefix). These selected trajectories show an approximately linear decoded frontier at $n = 32$ under different training budgets. Tight log- T training yields $v \approx 2.4$ (left; a fast-end seed of the log- T cell, median 2.0), $T=n$ training yields exactly $v=1.0$ (middle), $T=n/2$ yields $v \approx 2.1$ (right).

The archived fit uses 23 measurable S_4 models from 27 attempted arms, at eight eligible nominal demands spanning $8\times$. Its endpoint-rate exponent is 0.98 ± 0.04 (95% normal-approximation interval; $R^2=0.99$; Fig. 4), with $v = 0.995 d^{0.983}$. Here $d = 1/m$ is nominal demand, rounded to three decimals in the fitting script; realized demand is $n/\lceil mn \rceil$. One rate per run aggregates $n \in \{16, 32\}$ in the archived $T_{\text{obs}} = 40, 64$ windows (one earlier $n = 32$ trajectory uses 80 loops). It supports

approximate proportionality in this sweep, rather than an exact equality or a theorem that optimization selects the slowest possible solution. The fit is conditional on a measurable frontier. None of the three $T=3n$ attempts contributes a rate, and three of four $T=2n$ attempts do. Family-specific curricula and stage allowances also vary. Under the log- T schedule, demand rises across curriculum stages and speed under-delivers at the long end. A consistent demand of 4 can instead produce speeds near 4 at the same model scale.

Frontier evaluation windows. At loop t , the frontier is the longest prefix whose positions each have batch accuracy at least .9. The archived endpoint rate is defined when at least two positive increments exist. Its endpoints are the loop before the first increment and the loop of the last increment. General frontier evaluations use $T_{\text{obs}} = \min\{64, \max[16, \lfloor 2.5T_{\text{base}}(n) \rfloor]\}$. For proportional schedules $T_{\text{base}} = n$ independently of multiplier m . This window is not generally 2.5 times the training allowance. The depth study applies the completion gate stated with Table 12.

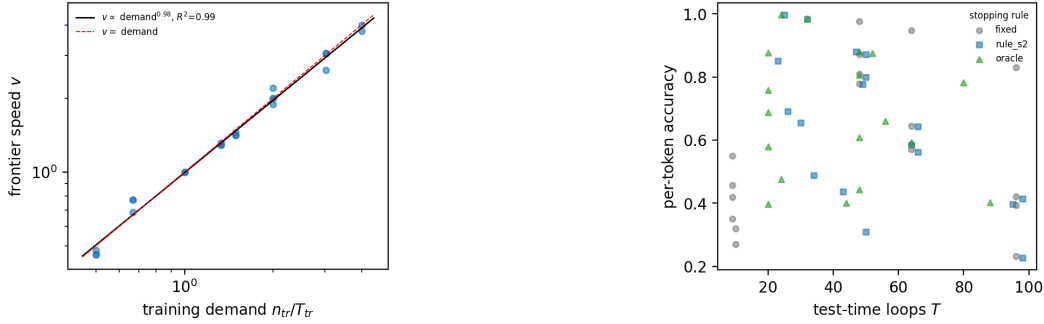


Figure 4: Archived budget fit (left) and candidate loop allocation (right). The fit uses 23 rate-measurable models from 27 attempted S_4 arms. Exponent 0.98 ± 0.04 , $R^2 = 0.99$, nominal demand $1/m$ (dashed. $v = d$). Right. $\lceil n/\hat{v} \rceil + 2$ (blue), finite-sweep oracle (green), and stored $T = n$ reference (grey). For $m = .5$, the reference is twice the native training budget; these are finite-sweep comparisons.

A.2 STATE REPLACEMENT

Recipients are drawn from 192 inputs and conditioned on a correct clean final output (at least 32 retained). Donors are selected by one random batch permutation at the same position and loop; self-pairs and unsolved donors are allowed. Damage is the fraction of clean-solved recipients made incorrect at each position. The $t > i$ subset is a coordinate proxy for a passed frontier and need not equal a measured per-model convergence boundary. Recovery is a separate property. A_5 models have mean self-damage on the $t > i$ grid subset $\leq .02$ in 4/5 tested seeds, while the representative tight-budget and operator-first models retain more corruption (Table 6).

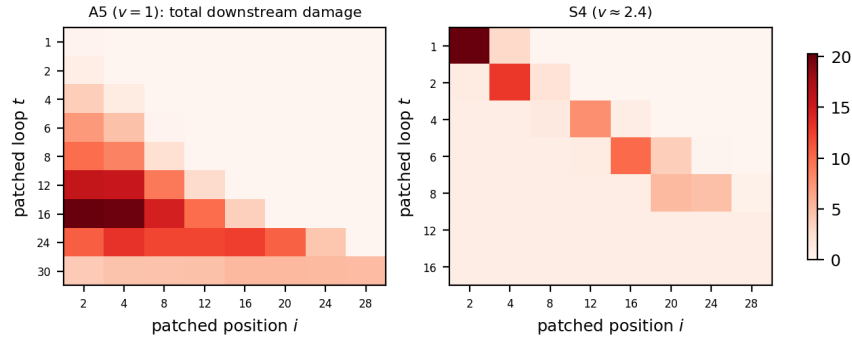


Figure 5: Causal intervention maps for the selected S_4 models. Total downstream damage follows the decoded frontier in these models. Zero upstream damage follows from the causal mask.

A.3 BUDGET TRANSFER

The forward transfers use the first four seeds for each group. At the target budget $n = 32, T = 8$, their per-token accuracies range from .606 to .614 for A_5 and from .72 to .73 for S_4 . The reverse S_4 transfer uses $n = 32, T = 32$. Its final task evaluation covers four models, while only two have saved frontier trajectories. The archived stopping comparison tests slack $s \in \{0, 2, 4\}$ and plots $s = 2$. Its proportional reference is $T = n$, including the model trained with $m = .5$. That reference is not the latter model’s native training schedule. The oracle is the best observed value in the recorded finite loop sweep.

Table 2: Budget transfer and training-length exact-match success. Success means exact match $> .5$ at $n = 32$ under the target schedule. $T = 8$ for log- T and $T = 32$ for proportional schedules. Transfer rows use seeds 0 to 3. Rates are standard medians over saved frontier measurements; only two reverse-transfer models have such measurements. Cohort provenance is given in Appendix G.1.

Arm	target budget	success	v before	v after
A_5 transfer	$T = n \rightarrow \log$	0/4	1.0	1.96
S_4 transfer	$T = n \rightarrow \log$	0/4	1.0	2.38
S_4 transfer	$\log \rightarrow T = n$	4/4	n/a	1.12 (2 models)

A.4 PREDICTION AT TRAINING LENGTHS

We race strictly in-distribution frontier and similarity measurements against mean per-token accuracy at $n \in \{64, 128\}$ under native schedules, across three 16-seed single-recipe populations. Group-task predictors use $n = 32$; parity uses $n = 16$, its actual training cutoff. The corrected comparison retains finite complete pairs. 16 for S_4 , 7 for A_5 , and 16 for parity. The remaining A_5 OOD outcomes are unavailable rather than zero. We report tie-aware Spearman and raw-score AUROC with a within-cell median outcome split; these are observed associations, not a fitted predictor evaluated on independent seeds.

For this corrected race, head measurements use 1,024 ID inputs per model, similarity probes use 128, and each OOD endpoint uses 1,024. The frontier rate ends at the first attainment of the observed maximum prefix in a common $4n$ window, requiring at least two positive increments; completion is read at the native schedule ($T = 8, 32, 16$ for the three cells). This differs from the archived endpoint-rate protocol in Eq. 4. Similarity measurements use $n = 16$. Adjacent-state cosine, shared-probe JVP alignment at sampled transition pairs, and output agreement between nonidentical initializations (AA). They do not establish a hidden fixed point.

Table 3: Corrected strictly-ID comparison. Spearman ρ (tie-corrected raw-score AUROC). Complete pairs number 16/7/16 for S_4/A_5 /parity, with 8/8, 3/4, and 8/8 positive/negative outcomes above/below-or-at the within-cell median. ID lengths are 32/32/16; OOD lengths are 64, 128. Parity completion is constant (undefined ρ , AUROC .50). AA denotes output agreement across initializations (Anil et al., 2022a).

Metric at ID length	S_4 log- T	A_5 $T=n$	parity $T=n$
frontier rate [head]	+0.81 (0.82)	+0.63 (1.00)	−0.08 (0.61)
completion at native budget	+0.80 (0.76)	+0.95 (1.00)	n/a (0.50)
cosine (adjacent)	−0.00 (0.45)	+0.50 (0.67)	−0.04 (0.36)
sampled-JVP alignment	−0.65 (0.09)	+0.64 (0.75)	−0.04 (0.44)
AA / path agreement	+0.51 (0.66)	+0.45 (0.67)	+0.13 (0.53)

Score orientation. Reversing a score replaces AUROC A by $1 - A$. The sampled-JVP values are .09375 and .90625 in the two orientations for S_4 . The corresponding A_5 values are .75 and .25. These are algebraic transformations rather than new experiments. Score orientation must be fixed without using held-out outcomes.

B OPEN-BENCHMARK VALIDATION

On the community easy-to-hard prefix-sums benchmark (Schwarzschild et al., 2021) (official 32-bit training split, official 512-bit test set), we compare accuracy across the same loop-schedule families (Table 4). Under the $T=n$ budget, successful seeds extrapolate to four times the training length at near-perfect exact-match accuracy; the seed-to-seed variance replicates; and the log- T arm fails at 512 bits. No saved ETH frontier-speed artifact is available, so this experiment tests accuracy extrapolation rather than the budget relation quantitatively. Exact-match accuracy falls sharply between 128 and 256 bits, short of the 512-bit extrapolation of convolutional Deep Thinking networks (Bansal et al., 2022). Convolutional locality is one possible explanation, but architecture, training, and evaluation settings differ across studies. Our experiment establishes the shorter observed range, not its architectural cause. Full 10,000-example re-evaluation gives exact match .9853/.9585 at $n = 128$ for proportional seeds 2/3, and zero at $n = 512$ for all eight models. Small nonzero outcomes include .0017 for proportional seed 3 at $n = 256$ and .006 for log seed 0 at $n = 64$.

Table 4: Easy-to-hard prefix sums (official 32-bit training split). Exact-match accuracy by length and seed. The $n = 32$ column uses the final logged training evaluation; longer lengths use archived 512-example test evaluations. Full-split re-evaluation is summarized above.

Arm	$n=32$	$n=64$	$n=128$	$n=256$	official 512
$T=n$, seed 2	1.00	1.00	1.00	0	0
$T=n$, seed 3	1.00	1.00	0.96	0	0
$T=n$, seed 0 (failed)	0.00	0	0	0	0
$T = n$, seed 1 (failed)	0.01	0	0	0	0
log- T , seed 0	1.00	0	0	0	0
log- T , seed 1	0.18	0	0	0	0
log- T , seed 2	0.05	0	0	0	0
log- T , seed 3	0.42	0	0	0	0
DT-Recall + prog. loss (Bansal et al., 2022)	reported > 97% at 512 bits (separate setting)				

C EXTENDED RELATED WORK

Looped models and their mechanisms. Universal Transformers (Dehghani et al., 2019) tie weights across depth. Other architectures share parameters (Lan et al., 2020; Hutchins et al., 2022; Csordás et al., 2022; Tan et al., 2023; Csordás et al., 2024; Bae et al., 2025). Open models scale this idea (Huginn-3.5B, Geiping et al., 2025; Ouro, Zhu et al., 2025). Latent chain-of-thought moves reasoning into activations (Hao et al., 2024), while small recursive networks solve puzzles by iteration (Wang et al., 2025; Jolicoeur-Martineau, 2025). Looped transformers can be programmed to simulate graph algorithms (Giannou et al., 2023; Back de Luca & Fountoulakis, 2024). Their reasoning performance can rival deep non-looped models (Saunshi et al., 2025). Constructions length-generalize on n -RASP-L tasks (Weiss et al., 2021; Zhou et al., 2024; Fan et al., 2024) and emulate iterative solvers (Yang et al., 2023; Gatmiry et al., 2024). Blayney et al. (2026) study state convergence and attention stabilization in Ouro and Huginn. Huginn approaches a fixed point while Ouro need not. Attention stabilizing with the state motivates our tail-saturation analysis. Lens studies find limited latent-CoT structure in Huginn (Lu et al., 2025). Our per-loop decode extends logit-lens readouts (nostalgebraist, 2020; Belrose et al., 2023) along the loop axis. Kohli et al. (2026) patch states at (position, iteration) granularity in toy recurrent-depth models, localizing retrieval and composition. We pair finite-range output stabilization with targeted interventions on controlled group tasks. Activation patching supports circuit discovery in fixed-depth models (Vig et al., 2020; Geiger et al., 2021; Meng et al., 2022; Wang et al., 2023; Hanna et al., 2023; Conmy et al., 2023; Yao et al., 2024); we apply it along loops. Feature-level tracing (Ameisen et al., 2025) and compiled ground-truth circuits (Lindner et al., 2023) are complementary. Deterministic attribution using only identical tail states and local Jacobians inherits our fixed-point limitation. Earlier-loop tracing can remain informative. Path independence, or convergence to input-determined fixed points, predicts upward generalization in deep equilibrium models (Bai et al., 2019; Anil et al., 2022a). Recall-style training promotes it (Bansal et al., 2022). We use asymptotic alignment as a baseline and measure how training pressure shapes convergence.

A measurement analogy from systems neuroscience. Decoding identifies when information becomes readable. Perturbations test evolving states’ causal roles and recovery (Churchland et al., 2012; Mante et al., 2013; Li et al., 2016). We borrow this experimental logic without asserting biological equivalence. τ measures decodable-output stabilization; state patching tests causal dependencies.

Complexity, expressivity, and imposed step-indexing. Under the cited bounded-precision and architecture assumptions, fixed-depth transformers sit in TC^0 (Merrill et al., 2022; Merrill & Sabharwal, 2023); Strobl et al. (2024) surveys these results. Solvable-group automata admit learned constant-depth shortcuts. Non-solvable groups (S_5 , A_5) admit none unless $TC^0 = NC^1$ (Liu et al., 2023; Barrington, 1989). The backdrop is Krohn-Rhodes theory and the fine structure of NC^1 (Krohn & Rhodes, 1965; Barrington & Thérien, 1988). Log-depth looped transformers use associative scans (Merrill & Sabharwal, 2025; Sanford et al., 2024); fixed-depth SSMs fail S_5 state tracking (Merrill et al., 2024). This ladder defines our hardness axis. Parity has constant-depth solutions (Chiang & Cholak, 2022), overcoming a soft-attention limitation (Hahn, 2020), yet length-generalizable learning remains hard (Bhattamishra et al., 2020; Delétang et al., 2023; Hahn & Rofin, 2024). Our claims therefore concern training selection rather than architectural capacity. Length generalization in standard transformers depends on positional encoding and input format (Anil et al., 2022b; Kazemnejad et al., 2023; Lee et al., 2024; McLeish et al., 2024). Scratchpads externalize intermediate state (Nye et al., 2021; Wei et al., 2022; Dziri et al., 2023). Chain of thought extends constant-depth models through serial token-space computation (Feng et al., 2023; Merrill & Sabharwal, 2024; Li et al., 2024; Goyal et al., 2024; Pfau et al., 2024). Loops provide an activation-space channel without additional parameters. Halting and allocation policies determine its compute budget (Elbayad et al., 2020; Schuster et al., 2022; Snell et al., 2025). Iteration-wise supervision aligns loops with CoT steps (Yu et al., 2026). CLRS-style hints align processor iterations with algorithm steps (Veličković et al., 2022; Ibarz et al., 2022), following the Neural GPU (Kaiser & Sutskever, 2016). Hint-free models can instead learn parallel strategies (Rodionov & Prokhorenkova, 2023). Our streaming arm imposes step structure architecturally; the free-input arms test whether it emerges without this constraint.

Group operations as a mechanistic testbed. Grokked networks implement Fourier or coset circuits for modular and permutation arithmetic (Power et al., 2022; Nanda et al., 2023; Zhong et al., 2023; Chughtai et al., 2023; Stander et al., 2024). Induction heads are sequence-processing primitives (Elhage et al., 2021; Olsson et al., 2022). Fixed-depth reasoners perform depth-bounded parallel search, caching intermediate states at token positions (Brinkmann et al., 2024; Wang et al., 2024; Cabannes et al., 2024). We study when loops compose operators along a sequence, rather than the circuit for $g \circ h$.

Table 5: Recorded optimization cost. First logged training step with per-token accuracy $\geq .98$ on pair composition ($n=2$). Costs differ across these operators; one scale up (M), the S_5 operator that takes 11k steps at S is learned in 6.5k steps.

	S_4 ($ G =24$)	$\mathbb{Z}_{60}$	A_5 ($ G =60$)	S_5 ($ G =120$)
scale S	$\sim 0.5k$	2.5k	$\sim 4k$	11k
scale M	0.5k	2.0k	2.0k	6.5k

Table 6: Repair diagnostic. Mean self-damage after resampling a state using grid cells with $t > i$, a coordinate proxy for a passed frontier. Low/high denote small/persistent final-readout damage. A_5 $T=n$ column. Survey over the five cracked seeds with retained checkpoints.

	A_5 $T=n$ (5 seeds)	S_4 log	S_4 $T=n/2$	S_5 op-first
self-damage on $t > i$ cells	.00 through .02 (4/5); .10 (1/5)	.95	.96	.99

D EXPLORATORY NEURON INTERVENTIONS

Shared-weight edits affect every recurrence and cannot isolate one step. We use activation interventions while distinguishing localization from editing (Hase et al., 2023). The archived loop-preference control permuted loop firing probabilities before computing Shannon entropy. Entropy

is permutation-invariant, so this control cannot support its claim of no loop specialization. A corrected per-example loop-shuffle null finds temporal firing alignment in almost all active units of four representatives. Alignment is not causal specialization.

The archived clamps remain descriptive perturbation results. Twelve selected FFN neurons per model were clamped to their window mean at every token position outside a nominal five-loop window around their peak, clipped at the rollout boundaries. On solved clean inputs, summed positional damage exceeded .1 for 10/12 clamps in the log-budget S_4 model, versus 1/12 for the $T = n$ S_4 model and 0/12 for the tested A_5 and operator-first S_5 models. The largest log-budget S_4 damage was 7.04 summed positions. These selected-model comparisons do not establish a general budget-redundancy law or identify loop-specialized neurons. Later held-out, duration-matched clamps likewise show model- and dose-dependent effects, without establishing such a law.

E TAIL BLINDNESS. FORMAL STATEMENT

Proposition 2 (Exact form). Let $G : \mathbb{R}^{N \times d} \rightarrow \mathbb{R}^{N \times d}$ be the loop map $G(h) = F_\theta(h, e)$ for a fixed input embedding e , assume G is differentiable at the tail state, and suppose the trajectory $h^{t+1} = G(h^t)$ reaches a fixed point. $h^t = h^*$ for all $t \geq \tau^*$, with $\tau^* \leq T - k$, using a fixed deterministic readout. Then. (i) for any $j \geq \tau^*$ and any $k \leq T - j$, skipping loops $j, \dots, j+k-1$ (i.e., outputting $G^{T-j-k}(h^j)$ in place of $G^{T-j}(h^j)$) leaves the final state, and hence the decoded output, unchanged; (ii) $J_t := \partial G / \partial h|_{h^t} = J_{\tau^*}$ for all $t \geq \tau^*$, so any similarity functional of local Jacobians is constant on the tail; (iii) for a fixed deterministic feature encoder E and an attribution rule that depends only on activations $z^t = E(h^t)$ and local linearizations at h^t (without explicit loop-index or remaining-horizon dependence), the resulting quantities coincide for $t \geq \tau^*$. The coincidence in (i) through (iii) does not uniquely identify the preceding computation; the Jacobian still constrains local first-order behavior.

Proof. (i) $G(h^*) = h^*$ implies $G^m(h^j) = h^*$ for every $m \geq 0$ when $h^j = h^*$; both branches output h^* . (ii) through (iii) are immediate since all quantities are functions of the (constant) tail state. Thus these equalities concern the stationary suffix, without identifying a unique earlier computation. $\square$

Proposition 3 (Approximate form). Suppose G is L -Lipschitz on a set containing the compared trajectories and $\|G(h^t) - h^t\| \leq \epsilon$ on the measured tail. If k loops are skipped from loop j , where $j \geq \tau^*$ and $0 \leq k \leq T - j$, then the difference between the original and skipped final states obeys

$$\|G^{T-j}(h^j) - G^{T-j-k}(h^j)\| \leq L^{T-j-k} \|G^k(h^j) - h^j\| \leq k\epsilon L^{T-j-k}.$$

If $L < 1$, the sharper bound is $\epsilon L^{T-j-k}(1 - L^k)/(1 - L)$. Under the same $L < 1$ condition, if G is differentiable and its Jacobian is M -Lipschitz on this set, then, for $s \geq t \geq \tau^*$, $\|J_s - J_t\| \leq M\epsilon(1 - L^{s-t})/(1 - L)$.

Proof. Apply Lipschitz continuity to the common $T - j - k$ remaining iterates, then telescope $G^k(h^j) - h^j$ over the k omitted updates. For $L < 1$, successive update norms are bounded by $\epsilon, L\epsilon, \dots, L^{k-1}\epsilon$, which yields the geometric sum. The same argument gives $\|h^s - h^t\| \leq \epsilon(1 - L^{s-t})/(1 - L)$; apply the M -Lipschitz property of the Jacobian. $\square$

These are state-space bounds. Unchanged decoded labels additionally require a sufficient readout margin, and continuous attribution statistics require their own continuity assumptions.

Small empirical update norms do not establish contraction. The propositions describe measurements under their explicit assumptions; we do not claim that all evaluated trajectories satisfy them. Models that deteriorate under additional loops fall outside an exact-fixed-point interpretation.

F CALIBRATION PROTOCOL AND PILOT DETAILS

Archived timing convention. Let $\hat{y}_i^t$ be the label decoded from state h^t . The stored diagnostic uses zero-based indices for the T post-update states $h^1, \dots, h^T$.

$$\tau_{\text{archived}}(n, i) = \min\{k \in \{0, \dots, T-1\} : \hat{y}_i^{t+1} = \hat{y}_i^T \forall t \in \{k, \dots, T-1\}\}, \quad (5)$$

$$\tau_{\text{archived,max}}(n) = \text{median}_{\text{solved inputs}} \max_{0 \leq i \leq n} \tau_{\text{archived}}(n, i). \quad (6)$$

Here $i = 0$ is the BOS position. The archived maximum includes its unsupervised readout; archived calibration statistics use this convention and the lower sample median over solved inputs. The supervised convention in Eq. 3 excludes BOS and uses the ordinary sample median. Fits over solved lengths are finite-range diagnostics, not asymptotic complexity classifications.

For the corrected check, $\tau_{\text{sup}}(n, i)$ is the first completed update $t \in \{1, \dots, T_{\text{native}}\}$ after which all recorded labels at supervised position $i \in \{1, \dots, n\}$ equal its final label. We take the ordinary median of $\max_{1 \leq i \leq n} \tau_{\text{sup}}(n, i)$ over final-solved inputs. At least 16/256 solved inputs are required per cell. Across the five native lengths 8, 16, 32, 64, 128, this gives 223 eligible model-length cells and median $\tau_{\text{sup,max}}/T_{\text{native}} = 0.28125$; the archived 144-cell, .71 summary is not reproduced. These values use a different mask, indexing convention, and horizon protocol from the archived calibration.

Table 7: Archived pilot, calibration, and endpoint-fit summaries, except the explicitly corrected target-only timing row (Appendices F and G). Archived ranges and slopes are not revalidated by the corrected timing protocol; their original intervals retain their scope.

Measurement	Population or condition	Reported value
Pilot exact match	64 models; ID / OOD range	1.00 / 0.00 through 1.00
Within-recipe OOD spread	Parity, NoPE, injection; $n = 32$	0.20 through 1.00
Jacobian alignment	ID all-pair probes; 64 pilots	0.85 through 0.99
Similarity association	Cosine/JVP vs. $n = 32$, by task	$\max \rho \approx 0.323$
Corrected timing / horizon	223 eligible model-length cells	median $\tau_{\text{sup,max}}/T_{\text{native}} = 0.28125$
Streaming parity slope	$d\tau_{\text{max}}/dn$, eight models	median 1.00; range 1.00 through 1.02
Log-budget $\mathbb{Z}_{60}$ fit	M, seed 1; $n = 8, 16, 32$	linear 0.24; $\log_2$ 3.0
Fixed-budget parity slope	Narrow solved range	0.39 through 0.41

The Jacobian range in Table 7 uses the archived ID all-pair probe mean (`id.jac_all`) over all 64 pilot models.

The within-recipe example in Table 7 uses parity, NoPE, input injection, proportional loops, and seeds 0 to 3 at $n = 32, T = 32$ (exact-match range .1973 through .9971). The similarity row uses cosine and sampled-JVP measures versus native-schedule $n = 32$ accuracy within each task. The $\mathbb{Z}_{60}$ calibration example is scale M, seed 1. $\tau_{\text{max}} = 1, 4, 7$ at $n = 8, 16, 32$, giving ordinary least-squares slopes .2411 against n and 3.0 against $\log_2 n$.

Archived τ calibration (legacy mask and indexing). These slopes retain their historical protocol; the corrected native-horizon summary in Table 7 does not revalidate them. Streaming parity (release-limited control). Median $d\tau_{\text{max}}/dn = 1.00$ across eight models (range 1.00 through 1.02). Fixed- T parity (budget-bounded). Local slopes 0.39 through 0.41, correctly capped by T . Proportional- T parity. Slopes 0.50 through 0.70, ordered with seed-level generalization. Learned-PE models, which fail beyond training length, yield fewer than three solved lengths and are reported as out of instrument range rather than assigned a class. Per-length τ requires at least 16/256 solved inputs; a slope is reported only with at least three eligible lengths; for tight-budget models that fail exact match at long n , frontier speed v (measured per length) substitutes, as in §7.

A valid omission starts from h^j and applies $T - j - k$ remaining updates. The corrected pilot re-evaluation covers all 64 checkpoints and passes 576 replay controls, with no replay or update-count errors. Phase-conditioned retention remains descriptive. For an autonomous map, the total executed updates determine the output, so phase strata alone do not localize causation.

The 64-model pilot’s archived single-skip diagnostic is excluded. Starting from h^{t+1} and applying $T - 1 - t$ updates returns h^T . Thus 1.0 retention in all 50 eligible models reflects identity replay. The timing code’s multi-loop routine correctly skips updates but does not validate those pilot values. Pilot cosine and Jacobian comparisons aggregate trajectory states rather than exclusively post-convergence tails. This limits their empirical use of Proposition 1.

Table 8: Control outcomes from the archived evaluations. Training success, length generalization, and intervention directionality are distinct measurements.

Control	Reported reading
State resampling	2 S_4 models, 56 grid locations each; upstream effect excluded by causal mask
A_5 length contrast	successful archived models have exact match 0 at twice training length
S_5 operator-first	10/10 trained-length successes; exact match $\leq .001$ at twice training length
S_5 free-training replications	scale L (~ 13.26 M) and learned PE. Exact match .00, 2 seeds each
Streaming parity	training-length exact match $\geq .999$, 8/8; at $4\times$ length. 0 through .998
Streaming, final-only	A_5 . 2/2 failed; S_5 . 8/8 failed; per-token $\leq .09$
Streaming plus prefix supervision	A_5 . 2/2 failed; S_5 . 4/4 failed; per-token $\leq .09$

G TRAINING AND EXPERIMENT DETAILS

G.1 COHORT PROVENANCE

The archived S_4 fit uses 23 measurable models among 27 attempted multiplier configurations. Its model-level rate aggregates lengths $n = 16, 32$. The A_5 frontier cohort uses a5v2 seeds 0 through 15. Five meet training-length exact match $> .5$ and seven have a recorded rate. The pooled A_5 scratch cohort combines a5v2 seeds 0 through 15, 18, and 19 with a5chain seeds 0 through 7. It has 8/26 training successes. Seeds 100 and 101 form a separate replication with 1/2 successes. The S_5 operator-first population combines the earlier seeds 0 and 1 with the later seeds 0 through 7. The success count appears in Table 8. Only the earlier pair has the rates in Table 1. Other speed rows use consecutive seeds beginning at 0. Their cohort sizes appear in Table 1. Copied artifacts are not counted again.

G.2 EXACT TASK DEFINITIONS

The archived free-training S_5 medium-scale cohort succeeds at the training length in 3/4 models. Its training allocation differs from operator-first training. The streaming schedule reveals token i from loop $i - 1$ onward and uses $T = n + 2$.

For decimal digits $d_1, \dots, d_n$ listed least significant first, increment has target

$$y_i = \left(d_i + \prod_{j < i} \mathbf{1}\{d_j = 9\} \right) \bmod 10.$$

The empty product equals one. The generator prevents overflow by leaving a non-nine digit beyond the controlled leading chain. A group is solvable when repeatedly taking its commutator subgroup eventually reaches the identity subgroup. A commutator $aba^{-1}b^{-1}$ measures failure to commute. S_4 is noncommutative but solvable, whereas S_5 and A_5 are non-solvable. Solvable groups admit constant-depth solutions in the circuit model of Liu et al. (2023). The non-solvable group word problems considered here are NC¹-complete (Barrington, 1989). These statements concern circuit complexity rather than the ease of learning a target from examples.

Architecture. Pre-norm transformer blocks, 4d FFN, GELU; weight-tied across loops; input injection via a learned linear adapter on $[h; e]$; readout head on $\text{LN}(h^T)$. Scales. s (2 layers, $d=128$, 4

heads, $\sim 0.44\text{M}$ params), M (2L, $d=256$, 8H, $\sim 1.72\text{M}$), L (4L, $d=512$, 8H, $\sim 13.16\text{M}$ on S_4); counts vary with vocabulary. The S_5 large model has $\sim 13.26\text{M}$ parameters. Untied control U12 (12 distinct layers, $T=2$, depth-matched to S at $T=12$); block-depth ladder D1/D4/D8 ($d=128$, 4 heads, $1/4/8$ layers, tied). NoPE throughout except the learned-PE arms.

Training. AdamW, lr 3×10^{-4} (2×10^{-4} for L), weight decay 0.01, grad clip 1.0, batch 256 (128 for L). Deeper tied blocks at loose budgets are lr-sensitive. The D8 and L demand-1 cells of Table 12 use lr 10^{-4} and 7×10^{-5} respectively (at the default lr they reach high mean accuracy but with ragged frontiers or late divergence). Length curriculum with stages $n_{\max} \in \{4, 8, 16, 32\}$ (6k through 12k steps each), promotion at per-token accuracy ≥ 0.98 after at least 500 steps; a stage also advances when its step budget expires. Lengths are sampled uniformly within a stage. Loop schedules. Log ($T = \lceil \log_2 n \rceil + 3$, jitter ± 1), prop ($T = \lceil n \cdot m \rceil$ for multiplier $m \in \{0.25, 0.33, 0.5, 0.67, 0.75, 1, 1.5, 2, 3\}$), fixed ($T=12$), streaming ($T=n+2$). Horizon curriculum (used for chained A_5/S_5). The per-position loss is truncated to the first h positions, h ramping $4 \rightarrow n_{\max}$ within each stage. Operator-first S_5 curriculum. Stages 2:20k, 4:10k, 8:10k, 16:10k, 32:12k at scale M. Annealing arms warm-start from a converged checkpoint of the source schedule and re-train under the target schedule with stages 16:6k, 32:8k.

Population sizes. 64 pilot models; 56-configuration Stage-2 grid ($S_5/S_4/\mathbb{Z}_{60} \times \text{scales} \times 4 \text{ seeds} + \text{streaming arms}$); expansion to 16 seeds in the three race cells; controls (A_5 , matched-order; untied; learned-PE; T -multiplier; streaming $\pm$ deep supervision; operator-first; annealing); 8 easy-to-hard models; 44-configuration depth-width grid with lr controls (Table 12) and S_5 wall replications at scale L and with learned PE. Experiments use RTX 3090 hardware. Model-level configurations provide the inventory; counts can overlap when checkpoints are reused.

Table 9: Tying-degree control at fixed effective depth 12 (G weight groups cycled; mean per-token accuracy at $n=64$, 3 seeds per cell). With parameters matched, $G=2$ outperforms full tying; the fully untied S_4 model is worst in this ladder. The A_5 , $G \geq 3$ cells retain short prefixes near two positions. A two-position prefix with chance accuracy elsewhere predicts $2/64 + (1 - 2/64)/60 \approx .0474$, about $2.84 \times$ the chance floor.

		$G=1$ (tied)	$G=2$	$G=3$	$G=4$	$G=6$	$G=12$ (untied)
S_4	const. width	.21	.27	.30	.30	.32	.26
	param-matched	.21	.28	.27	.26	.25	.15
A_5	const. width	.10	.23	.17	.25	.23	.15
	param-matched	.10	.15	.05	.05	.05	.05

Tying-degree control. Table 9 reports the per-token accuracies behind the tying ladder of §5.

Table 10: Rescue surface and overthinking. Archived mean per-token accuracy vs. test-time loops T (saved three-decimal evaluation aggregates over the first three sorted seeds, or fewer if unavailable, without success filtering). The ETH row includes two failed models and one successful run. Tight-budget ($\log T$) groups are rescued by T beyond their training budget and then degrade; operator-first S_5 reaches near-perfect per-token accuracy at $n=32$; $\mathbb{Z}_{60}$ and annealed models collapse sharply past their operating range.

Group	n	test loops T						
		4	8	12	16	24	32	48
S_4 log s	32	0.382	0.688	0.947	0.969	0.907	0.846	0.780
S_4 log s	64	0.213	0.365	0.517	0.632	0.655	0.599	0.552
S_4 log L	32	0.384	0.788	0.957	0.859	0.741	0.686	0.644
S_5 op-first M	32	0.134	0.259	0.384	0.510	0.756	0.994	0.992
S_5 op-first M	64	0.071	0.134	0.197	0.258	0.382	0.507	0.640
transfer A_5 p $\rightarrow$ l	32	0.327	0.612	0.861	0.758	0.326	0.255	0.243
$\mathbb{Z}_{60}$ log s	32	0.348	0.748	0.858	0.741	0.284	0.124	0.082
ETH prop (mixed success)	32	0.579	0.633	0.715	0.730	0.756	0.760	0.769

Rescue surface. Table 10 reports the test-time-loop sweep behind the rescue and overthinking claims of §5.

Budget-law fit. The regression in Figure 4 uses 23 measurable scale- s S_4 models from 27 attempted multiplier arms (families $s_4v_2/s_4v_4/s_4v_7$). Nominal demand $1/m$ is the regression predictor; curricula and stage allowances are not identical across families. Log-schedule, learned-PE, annealed, and larger-scale models are excluded. Table 11 retains the archived endpoint fit $v \propto d^{0.98 \pm 0.04}$ ($R^2=0.99$). A separate common- $4n$ re-evaluation gives first-passage exponent .907 ($R^2 = .983$), or 1.182 ($R^2 = .932$) with the last-increment endpoint. These sensitivity fits change the observation protocol and do not replace the archived estimator or its table.

Table 11: Points in the budget-law fit. Standard median rate over $n \in \{16, 32\}$ within a model, then across models at each demand. The same 23 model-level rates determine the regression.

nominal demand $1/m$	0.50	0.67	1.00	1.33	1.49	2.00	3.03	4.00
seeds	3	3	2	3	3	4	3	2
median v	0.46	0.77	1.00	1.29	1.41	2.00	3.05	3.89

G.3 DEPTH AND WIDTH RECONSTRUCTION

Table 12 reconstructs depth and width comparisons (§5). Report the standard median rate over model-length cells at $n \in \{16, 32\}$ with peak prefix completion at least .85. Prefix positions require batch accuracy at least .90. The gate uses peak completion within $T_{\text{obs}} = 40, 64$, not the training allowance. Depth and scale cohorts use two seeds per cell. At demand one, the eight-layer and large-model rows use separate lower-rate cohorts. Their default-rate counterparts pass 0/4 cells. The one-layer demand-four cell includes an extra lower-rate configuration.

Table 12: Frontier speed by block depth and width on S_4 (median v [range] over seeds and test lengths $n \in \{16, 32\}$; 2 seeds per cell plus learning-rate controls where noted). At demand four, the one-layer cohort has one eligible $n = 16$ cell and no eligible $n = 32$ cells; deeper blocks advance farther. The wider four-layer cohort has a lower median here but differs in other training settings. The marker n/a denotes cells outside these dedicated cohorts.

Block (width $\times$ layers)	demand 1	demand 4
128×1	0.98 [0.93, 1.00]	1/6 cells pass; $n = 32$. 8/32 to 13/32
128×4	1.03 [1.00, 1.07]	4.67 [4.00, 5.33]
128×8	1.00 [1.00, 1.03] [†]	4.95 [4.00, 5.33]
256×2	1.00 [1.00, 1.03]	n/a
512×4	1.02 [1.00, 1.07] [†]	3.60 [3.11, 5.33]

[†]at $\text{lr } 10^{-4}$ (128×8) and 7×10^{-5} (512×4); at the default lr these cells train to high mean accuracy but the frontier is ragged or training diverges late. On A_5 at demand 1 the 128×4 cell fails to learn at every lr tried (0/5 models across 3 learning rates); the 128×1 and 128×8 cohorts succeed in 2/2 and 1/2 models, respectively. These counts reuse seed values across learning rates.

Easy-to-hard protocol. Official 32-bit training split (10,000 examples) with random prefix crops $n \in [4, 32]$ for the length curriculum; evaluation on the official 512-bit test split and on generator-matched intermediate lengths; NoPE, scale s , 12k steps.

Metrics. Exact match = all supervised positions correct; per-token accuracy over supervised positions. Generalization outcome in §7. Mean per-token accuracy at $n \in \{64, 128\}$ under the schedule-matched loop budget, recomputed uniformly from checkpoints. Statistics. Tie-aware Spearman; partial correlations by residual rank regression on adjacent-loop cosine.

H DISCUSSION AND SCOPE

Frontiers describe readable progress over measured lengths. Stable outputs need not imply fixed hidden states. Timing depends on the window and correctness criterion. The budget relation is clearest for measurable S_4 models. Training, curriculum costs, and trajectory coverage limit universal or uniquely mechanistic interpretations. Matched budgets and independent populations would test transfer. Rescue values are rounded aggregates. Original rescue evaluation draws, full checkpoints, and external datasets are not bundled.