WORK ON STACKS: TRANSFORMERS CAN DO PUSHDOWN UPDATES BUT STRUGGLE TO LEARN THEM

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers are remarkably capable sequence models, but reliably maintaining and updating state over long contexts remains challenging. In this paper, we show that while Transformers can represent exact pushdown state-update computations, learning and executing these computations reliably can be substantially more difficult. We study bounded pushdown updates with complete action–state histories retained in context, without any external symbolic memory or retrieval, and construct a Transformer that implements the update exactly. Empirically, we find that learning succeeds reliably when the required routing computation is provided, but becomes much less reliable when the model must acquire that computation itself. A separate end-to-end study finds increasing training requirements with stack depth and alphabet size, extending the acquisition difficulty beyond unary stacks. Finally, we evaluate pretrained language models and find that pushdown state updates become unreliable in long contexts even when the relevant state can still be recovered. Overall, these experiments show a fundamental gap between representing state and reliably using it for sequential computation.

1 INTRODUCTION

Transformers have become the dominant architecture for sequence modeling, supporting a wide range of language and reasoning tasks (Vaswani et al., 2017). At the heart of this success is the ability of self-attention to combine information from different parts of the input context. Yet increasing the amount of available context does not necessarily make this computation more reliable: language models can be distracted by irrelevant information, degrade as inputs become longer, and fail even when the relevant evidence can still be retrieved (Liu et al., 2024; Shi et al., 2023; Levy et al., 2024; Du et al., 2025). A better understanding of whether such failures arise from representational limits, from learning the required computation, or from using available state at inference time is therefore important for building reliable long-context models.

Pushdown memory provides a simple setting in which these possibilities can be separated. A stack is updated by a small recurrent rule, but the correct update may depend on state accumulated over an arbitrarily long history. Formal-language studies have used such algorithmic tasks to characterize the capabilities and limitations of neural sequence models, including Transformers (Hahn, 2020; Yao et al., 2021; Delétang et al., 2023; Strobl et al., 2024). Related work has shown that shallow Transformers can represent and sometimes learn shortcut implementations of recurrent automata, while constructive analyses show that attention can implement explicit retrieval algorithms such as copying (Liu et al., 2023a; Jelassi et al., 2024). These results clarify what Transformers can represent, but leave open a different question: when a correct state-update computation exists, which parts of that computation are actually learned, and will a trained model use the relevant state reliably when irrelevant history grows? Figure 1 summarizes our approach to separating these questions: we first construct an exact pushdown update, then vary which parts of its Transformer implementation are supplied or learned, and finally test whether pretrained language models reliably use an explicitly available state.

To study this question, we use bounded stacks with complete action–state histories retained in context. Supplying only the latest stack and action would externally isolate the information needed for the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

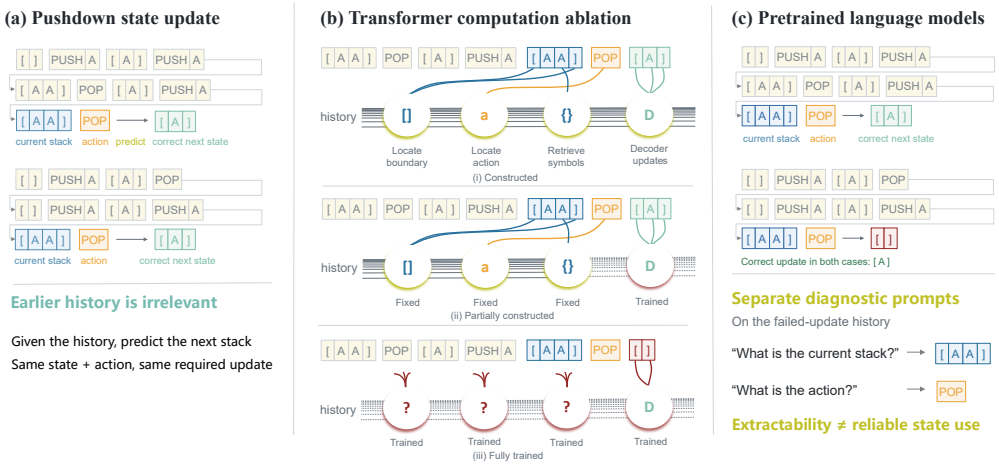

Figure 1: **From representing state to learning and using it reliably. a: Pushdown state update.** Different histories can lead to the same current stack and action and therefore require the same next-state update; earlier history is irrelevant once the sufficient state is available. **b: Learning the update computation.** Our construction separates attention stages that locate and route the relevant state information from a decoder that produces the next stack. This lets us compare supplied computation, partial learning, and end-to-end acquisition. **c: State use in pretrained language models.** Histories with the same current state and action can nevertheless produce different updates. On failed-update histories, separate prompts can still recover the current stack and action, distinguishing state accessibility from reliable state use.

update. Here, the model must identify and update the current stack while earlier snapshots remain visible. Explicit snapshots remove the need to reconstruct the stack from actions alone, but do not supply a runtime mechanism that accesses or updates it. The simplicity of the symbolic transition therefore does not establish the ease of learning its contextual execution. We construct a Transformer that implements the update exactly, providing a reference computation rather than only an existence argument. This construction separates the computation into attention stages that recover and route the relevant state information and a final decoder that maps this information to the next stack. Our analysis therefore studies representational capacity first, before asking whether the same computation can be acquired from data, following the broader use of compiled Transformers as controlled laboratories for algorithmic behavior (Weiss et al., 2021; Lindner et al., 2023; Shaw et al., 2025).

We then study how this computation is learned by selectively resetting different parts of the constructed Transformer. To address this question, we train models on the same stack-update task while varying which parts of the constructed computation are provided and which must be learned. We find a clear difference between learning the final state-update rule and learning the attention-based computation that supplies its inputs. These experiments suggest that the main difficulty is not simply representing pushdown memory, but acquiring a reliable computation for accessing and updating it.

Finally, we study whether pretrained language models satisfy the same task-level requirement: producing the correct update independently of obsolete history when the sufficient current state is explicitly supplied. We evaluate models on stack-update problems in which the current state is explicitly available in the context and vary the amount and content of preceding history. We find that the models become substantially less reliable as the context grows, and that histories leading to the same current state can nevertheless produce different updates. At the same time, the relevant state can often still be recovered when the model is asked to extract it directly. These results suggest that storing information in context and reliably using that information for state updates are distinct capabilities.

Our main findings are:

- **Representability does not imply easy acquisition.** A Transformer can implement the bounded pushdown update exactly, yet learning is substantially easier when the required

attention computation is provided. A separate end-to-end study finds increasing training costs as stack depth and alphabet size grow.

- **Available state does not ensure reliable updates.** Pretrained language models can receive the correct current state explicitly and still become sensitive to irrelevant history as the context grows.
- **State extraction and state use can separate.** Histories that lead to incorrect updates often still permit successful extraction of the relevant state under a separate prompt, motivating a distinction between storing, accessing, and executing on memory.

These results suggest that failures of pushdown memory in Transformers need not arise from an inability to represent the required state. They can instead emerge from the acquisition and execution of the computation that routes available state into the next update.

2 Representing a stack update

We first establish a correct Transformer implementation of the update. This provides a known solution for the learning experiments and allows us to intervene on individual components. We use this construction as a reference computation for these interventions.

2.1 An explicit-state task

The component-learning and pretrained-model experiments use a unary stack $S = A^d$, with $0 \le d \le D$, and the transition

$$T(A^d, \mathsf{PUSH}) = A^{d+1} \quad (d < D), \qquad T(A^d, \mathsf{POP}) = A^{\max(d-1,0)}. \tag{1}$$

For the end-to-end study in Section 3.3, we also consider stacks $S \in \Gamma^{\le D}$ with alphabet sizes $|\Gamma| \in \{1, 2, 4, 8\}$. The stack top is on the right: $T(S, \mathrm{PUSH}(x)) = Sx$ for $|S| < D$, and $T(Sx, \mathrm{POP}) = S$, with empty POP again a no-op. The input interleaves legal actions with complete bracketed stack snapshots and ends with the current state and a supplied next action. A PUSH includes its argument. The model generates the entire next stack, including brackets, by unconstrained greedy decoding. Empty POP is a no-op. Every earlier state is visible; the model receives neither a cropped context nor an external stack at inference time.

Write the input as (H, S, a), where H is the obsolete history. For two legal histories ending at the same state, the desired behavior is

$$f(H, S, a) = f(H', S, a) = T(S, a). \tag{2}$$

This condition combines two properties: correctness and history invariance. Equation 2 defines the behavioral requirement used throughout the paper. The construction provides one implementation that satisfies this requirement, and the learning experiments test whether trained components can compose to satisfy it. Section 4 evaluates the same requirement behaviorally in pretrained language models without assuming that they use the same internal computation. A model can satisfy the latter by returning the same wrong stack. Our evaluations therefore measure both. At fixed D, the unary setting has only $D+1$ semantic states, although many histories realize each state. At $D = 1$, even the state can be ignored: the action alone determines the answer. State-dependent learning conclusions use $D \ge 2$.

2.2 A constructive solution

The construction routes the latest source stack to a local output rule. Its three attention stages first identify the latest opening bracket, then locate the action and source boundary, and finally retrieve the source symbol, its lookahead, and the PUSH argument. A ReLU decoder decides whether to open, copy, append, or close the output:

$$E \longrightarrow \underbrace{A_1}_{\text{latest bracket}} \longrightarrow \underbrace{A_2}_{\text{action and boundary}} \longrightarrow \underbrace{A_3}_{\text{source retrieval}} \longrightarrow \underbrace{D}_{\text{next token}}. \tag{3}$$

Let s be the source opening position, a_{pos} the action position, and q the current prediction position. The source address is $j = s + q - a_{\mathrm{pos}} - \mathbf{1}_{\mathsf{PUSH}}$. Position and squared-position features implement

attention scores proportional to $-\beta(k-j)^2$. Thus retrieval follows the copy position rather than classifying the entire history into a next-state label.

Proposition 1 (Bounded exact decisions)**.** *For legal explicit-state traces with consecutive integer positions in* $[0, N]$*, the analytic three-stage Transformer produces the exact next stack under real arithmetic and greedy decoding whenever*

$$\rho = (N+1)e^{-\beta} < \tfrac{1}{2}, \quad \delta = (3N+1)\rho < \tfrac{1}{2}, \quad \max\{(N+1)\rho,\, 5(N+1)e^{-\beta(1-2\delta)} + 3\rho\} < 0.05.$$

The $0:8192$ *domain applies to the analytic representability result, while the stagewise learning experiments use the implemented* $0{:}512$ *positional domain.*

Finite softmax need not produce literal one-hot pointers. The proof bounds leakage at each selector, propagates the resulting address error through retrieval, and shows that it cannot reverse the decoder's positive margin. Autoregressive induction then gives the complete stack (Appendix A). The result concerns a bounded domain and the specified positional features; numerical checks are additional implementation evidence, not a universal floating-point guarantee. Most importantly, the construction establishes a solution without establishing that gradient training will find it.

3 Learning to Access and Update State

The previous section shows that a Transformer can implement the pushdown update exactly. However, this does not establish that the same computation can be learned from examples. We now study which parts of the constructed solution are reliably acquired from data before turning to pretrained language models in Section 4.

3.1 Routing computation and acquisition

We test whether a learned component can compose with the remaining stages of a known-correct implementation. A reset attention stage must learn to produce signals compatible with the fixed downstream computation. In contrast, a reset decoder only needs to learn a readout from the supplied analytic signals. The experiment therefore measures component acquisition under a specified interface, rather than the intrinsic learnability of attention and feed-forward computation. Results are shown in Figure 2a.

There is a clear difference between learning the final update rule and learning the attention computation that supplies its inputs. All four decoder-only runs reach exactness on both evaluation suites, whereas none of the isolated attention replacements, joint-attention models, or full models passes every query at a saved milestone. These models can nevertheless acquire highly accurate update behavior: the best A_3 model answers 401 of 402 complete-stack queries correctly. Thus, the difference is between acquiring a computation that works approximately and acquiring one that executes the update reliably on every tested case.

To test whether the frozen downstream interface alone accounts for this asymmetry, we jointly train each attention stage with the decoder while keeping the other attention stages analytic. In the selected runs, $A_1 + D$, $A_2 + D$, and $A_3 + D$ achieve 387/402, 392/402, and 401/402 correct final-suite queries, respectively (Table 4). Allowing the decoder to coadapt therefore does not yield an error-free solution in these runs. These are selected diagnostic runs rather than a replicated comparison, so they do not establish an intrinsic barrier to learning attention.

The previous experiment evaluates the learned replacements on a fixed finite suite. The primary matrix itself contains one run per component-initializer cell. The following confirmation instead uses five new initialization seeds and two paired data streams for selected conditions. We next test whether the same behavior persists on new models and under changes in input position. Specifically, we train selected conditions with new initialization seeds and data streams and evaluate the same legal histories at multiple position offsets. As shown in Figure 2b, decoder-only training remains exact in all ten runs at both tested depths, while no selected attention or full-model run passes the new panel.

The position intervention reveals a particularly sharp difference. At depth 2, moving exactly the same token sequence from offset zero to its maximum-safe position changes 282 of 600 A_3 model–history pairs from correct to incorrect, with no changes in the opposite direction. The decoder-only models

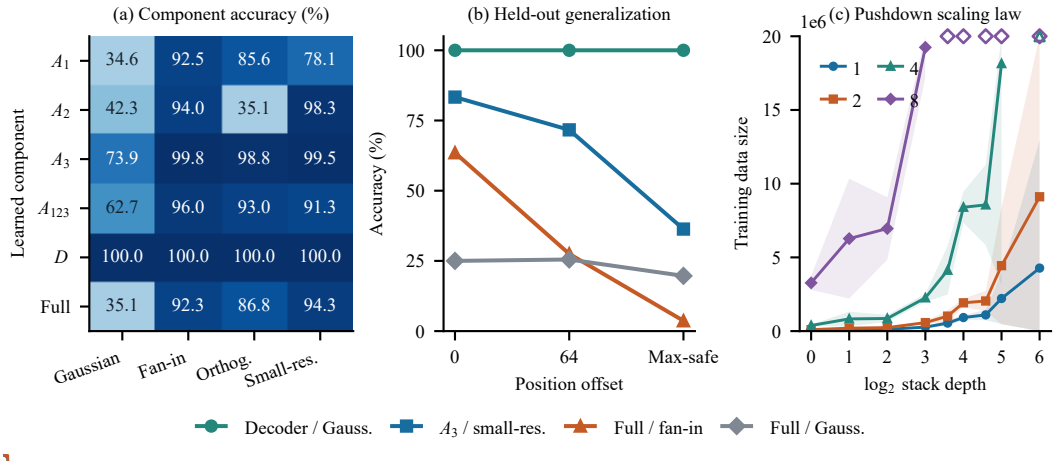

Figure 2: **Learning pushdown updates. a: Component acquisition.** Each primary matrix cell is one depth-2 run using initialization seed 11 and the common primary data stream. We reset individual parts of the analytic Transformer and train them on the stack-update task. Decoder-only training reaches exactness for all four initialization recipes, while the attention and full-model replacements remain imperfect. **b: Held-out generalization.** Selected models are evaluated on the same legal histories at different position offsets. Decoder performance remains unchanged, while the learned attention and full-model computations degrade under position shifts. **c: Depth and alphabet scaling.** A separate Transformer is trained from random initialization on bounded stack updates at depths up to 64 and alphabet sizes $|\Gamma| \in \{1, 2, 4, 8\}$. Filled markers show observed training traces to the first finite-suite pass; hollow markers indicate runs that have not passed by their training budget. Curves and bands show the mean and one sample standard deviation for settings in which all seeds pass. Full protocols and individual results are reported in Appendix C and Appendix G.

are unchanged by the same intervention. This shows that the learned attention computation can remain brittle to position even within the legal task domain.

Taken together, these experiments show that the final state-update rule is much more reliably learned when the required routing computation is already provided. This result is specific to the constructed interfaces and tested training procedures; it does not imply that attention computations are intrinsically unlearnable. Full architectural, initialization, training, and evaluation details are given in Appendix C.

3.2 Accurate local signals can still be used incorrectly

The stagewise experiments leave open why a learned attention component can recover much of the required information without producing a correct final update. We next test whether the problem can arise from how the learned component communicates with the fixed downstream computation. To do this, we directly supervise selected internal pointers and token categories in three diagnostic runs. The selected categorical probes become perfect, yet none of the resulting models produces exact stack updates. Accurate local readouts are therefore not sufficient for correct composition.

We test whether unintended residual writes account for part of this gap. Let Δ denote the learned stage increment, Δ^* its analytic counterpart, and P the projection onto the coordinates used by that stage in the analytic construction. We replace only the remaining coordinates:

$$\widetilde{\Delta} = P\Delta + (I - P)\Delta^*. \tag{4}$$

Because the analytic increment is zero outside these coordinates, this intervention preserves the learned canonical signal while removing its off-template writes.

The intervention substantially improves end-to-end correctness. For A_1, accuracy rises from 153/402 to 393/402 complete-stack queries. In contrast, replacing only the intended coordinate leaves performance at 153/402. Removing the off-template writes repairs 240 baseline failures without breaking any baseline success. The same intervention also improves A_2 and A_3, although by smaller amounts (Appendix E).

These results show that a component can contain a useful local signal and still interfere with the computation that consumes it. In these checkpoints, the off-template writes causally reduce correctness against the fixed analytic complement. The experiment does not imply that the compiler's coordinates are the unique or preferred representation of a fully learned network: the intervention uses three post-hoc checkpoints and the downstream computation is fixed. It instead explains how successful local prediction can coexist with an incorrect state update. In Section 4, we test the corresponding distinction behaviorally in pretrained models without assuming that they implement the same internal mechanism.

3.3 End-to-end acquisition becomes more costly with depth and alphabet size

The previous experiments study learned components within a supplied analytic computation. We next ask whether stack updates can be learned when the full representation is free to coadapt. We train a separate Transformer from random initialization with next-state supervision, varying stack depth from 1 to 64 and alphabet size over $\{1, 2, 4, 8\}$. Complete action–state histories remain in context, so the current stack is explicitly available, but the model must still identify and update it correctly.

Figure 2c shows that the observed training cost generally increases with depth and is higher for larger alphabets at comparable depths. The larger-alphabet conditions extend the task beyond unary depth tracking: producing the complete next stack requires preserving both symbol identity and symbol order. End-to-end acquisition is therefore possible in the tested setting, but reliable updating requires more training as stack depth and alphabet size increase. This pattern remains when the learned representation is allowed to coadapt rather than being constrained by a frozen analytic interface.

These measurements characterize the tested models, optimization procedures, and finite evaluation criterion. They should not be interpreted as an asymptotic sample-complexity law. Appendix F reports the full protocol, individual first-pass costs, and the outcomes of runs that do not meet the evaluation criterion within their training budget. Appendix G gives the complete training and selection protocol.

4 Using explicit state in pretrained language models

The previous experiments separate representation, acquisition, and composition in a constructed state-update circuit. We now ask whether a related distinction between state availability and state use appears in pretrained language models. We first test whether obsolete history can change an update when the sufficient current state is explicitly present, then ask whether that state remains accessible when updating fails, and finally examine related memory-object selection errors in structured histories. These experiments are behavioral and do not imply that pretrained models implement the internal computation of the constructed Transformer.

4.1 Obsolete histories change the next stack

We first test whether obsolete history can change a state update even when the sufficient current state is explicitly available. For each query, we construct matched groups of four histories with the same current state, requested action, recent suffix, suffix positions, and token length, while varying only the earlier history. The correct next stack is therefore identical within each group. We evaluate Qwen3-8B and Qwen3-32B at approximately 0.5K, 2K, 8K, and 32K tokens.

To avoid attributing failures to basic task competence, we restrict the main comparison to settings that pass the minimal-history and boundary controls: capacity 2 for Qwen3-8B, which passes 10/10 controls, and capacity 8 for Qwen3-32B, which passes 34/34. A separate history-extension panel varies length together with history content and is kept distinct from the matched-content comparison. Full inference settings and results for the other capacities are reported in Appendix H.

As shown in Figure 3a, both models are less accurate at the longest tested context than at the shortest. Qwen3-8B falls from 82.3% to 59.0%, while Qwen3-32B falls from 99.3% to 81.8%. More importantly, changing only obsolete history content can change the generated update. At 32K, 58/100 matched groups for Qwen3-8B and 74/340 for Qwen3-32B contain at least two different valid stack answers (Figure 3b). Because the correct update and input geometry are fixed within each group, these disagreements directly measure sensitivity to history that is irrelevant to the required transition.

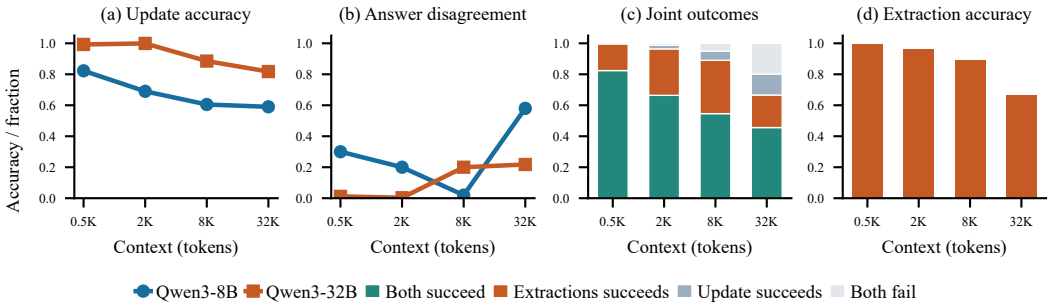

Figure 3: **Available state and reliable use can separate. a:** Complete next-stack accuracy across matched-history groups at increasing context lengths. Each band contains 400 queries in 100 groups for Qwen3-8B at capacity 2 and 1,360 queries in 340 groups for Qwen3-32B at capacity 8. The two models use different eligible capacities and therefore do not form a matched model-size comparison. **b:** Fraction of matched groups containing at least two different valid stack answers. Within each group, the correct update and input geometry are fixed while obsolete history content changes. **c:** Joint outcomes on 500 capacity-2 histories per band for Qwen3-8B, where extraction succeeds only when both the current stack and requested action are recovered under separate prompts. **d:** We report extraction success separately for histories with correct and incorrect updates, using the number of update outcomes in each group as the denominator. Extraction success is similar across the two groups through 8K. At 32K, extraction failures are more concentrated among histories with incorrect updates.

Accuracy and history sensitivity capture different failure modes. At 8K, Qwen3-8B answers only 60.5% of queries correctly, yet just 2/100 matched groups contain different answers; many of its errors are therefore stable across obsolete histories. Qwen3-32B also improves slightly from 0.5K to 2K before declining. The results therefore do not imply that either accuracy or history sensitivity must vary monotonically with context length. A fresh-seed Qwen3-8B panel shows similar descriptive behavior at 2K and 8K (Appendix H.3), but the two model sizes use different eligible capacities and do not establish a model-size law or generality across model families.

To characterize the incorrect updates, we annotate all failed responses in the eligible matched-history panels across the four context bands. Errors are predominantly small depth deviations: $503/517$ failures for Qwen3-8B and $360/415$ for Qwen3-32B are valid stacks whose depth differs from the correct next state by one. Of these same failure sets, 273 and 257 responses, respectively, leave the supplied current state unchanged. Matching the snapshot immediately before the current state is much rarer (11 and 0 responses). An explicit action-only baseline that returns $[A]$ for PUSH and $[\,]$ for POP matches 350 and 63 failures; this baseline is defined for this post-hoc comparison, not fitted to the responses. The indicators overlap and do not form a partition. The remaining 14 and 5 valid outputs match none of these indicators; an additional 0 and 6 outputs are invalid. Appendix J.2 gives the definitions and counts by context band. These categories describe output-level correspondences rather than internal routing mechanisms.

4.2 State extraction can succeed when updating fails

The previous experiment shows that obsolete history can change the generated update, but an incorrect update does not reveal whether the relevant state itself has become inaccessible. To separate these possibilities, we ask Qwen3-8B to extract the current stack and requested action in separate prompts on the *same histories* used for the update task. Exact query-level joins then distinguish failed updates from failed extraction. This analysis combines the matched and extension panels, giving 500 capacity-2 histories per context band; it is separate from the 400-query matched panel in Figure 3a–b.

Successful extraction often coexists with incorrect updating, but this comparison must account for the extraction rate in each outcome group. In Figure 3c–d, at 0.5K, extraction succeeds on $86/86$ failed updates and $412/414$ successful updates; at 2K, the corresponding counts are $150/156$ and $332/344$.

At 8K, extraction succeeds on $172/197$ failed updates (87.3%) and $273/303$ successful updates (90.1%). Through 8K, separate-prompt extraction is therefore similarly successful regardless of whether the original update succeeds. At 32K, this pattern changes: extraction succeeds on $105/204$ failed updates (51.5%), compared with $228/296$ successful updates (77.0%). Extraction and update failures thus co-occur more often at the longest context. Separate-prompt accessibility can coexist with update failure, particularly at moderate context lengths, but this dissociation is weaker at 32K.

Activation probes do not support a uniform account in which the current state remains equally accessible throughout the update computation. On a separate held-out panel at 2K, a fixed layer-26 probe correctly decodes current depth on $397/400$ queries and action on $400/400$. At 8K, depth decoding falls to $260/400$, while action decoding remains $400/400$. This decline is consistent with reduced state accessibility contributing to some update failures. However, unsuccessful linear decoding does not establish that the state information is absent, just as successful decoding does not establish that the model uses it causally. Full layerwise results and controls are therefore reported in Appendix I. The exploratory activation-patching analysis also identifies no confirmed repair mechanism. Our strongest evidence is therefore behavioral, based on the controlled matched-history comparison and exact query-level extraction joins.

4.3 Memory-object selection is sensitive to the interface

The explicit-state update task tests whether a model can use a sufficient state already provided in context. We next ask a related question: whether failures to select the correct memory object from a structured history persist when the relevant fields and operations are made explicit. Reference programs generate maze and Sudoku histories, and a separate Qwen3-32B diagnostic queries a single scalar field, such as a restored maze frame or a surviving ancestor assignment in Sudoku. The model therefore performs a targeted readout rather than an autonomous solution.

Several roles exhibit substantial errors under the original interface, but improve when the relevant fields and operations are stated explicitly. In the maze, nested restore increases from $2/10$ to $10/10$, single restore from $5/10$ to $10/10$, the still-suspended frame from $0/10$ to $7/10$, and the visible saved field from $5/10$ to $10/10$ by query ID. In Sudoku, the surviving-ancestor readout increases from $3/10$ to $8/10$. The other tested Sudoku roles are already correct on all query IDs under the original interface and remain so with explicit wording. These results make the original-interface errors difficult to attribute directly to an inability to select LIFO-defined memory objects: every role with a deficit under that interface improves when the relevant state and operations are made explicit. Appendix J reports the complete diagnostic, including distinct-prompt counts and oracle-assisted conditions.

5 Discussion and conclusion

Our results separate three questions that are often conflated in discussions of Transformer memory: if the required state-update computation can be represented, if it can be learned, and if available state is used reliably at inference time. We construct an exact bounded pushdown update, but find that learning its update rule is much more reliable when the required routing computation is already supplied. This difficulty is not an absolute learning barrier: end-to-end training can reach error-free finite evaluations, but its observed cost grows with stack depth and alphabet size. In pretrained language models, however, obsolete history changes the next update even when the sufficient current state is explicitly present, and separate extraction often succeeds on histories where updating fails. These results do not imply that Transformers cannot learn pushdown computation. Our experiments study bounded stacks, and the constructed and end-to-end studies use different architectures and training regimes. Likewise, the pretrained-model experiments cover two Qwen models, and the separate extraction prompts measure accessibility rather than the internal computation of the original update. The evidence therefore supports a distinction between representation, acquisition, and use, rather than a universal learning barrier or a single shared failure mechanism. This distinction suggests a different target for reliable memory systems. Making information available in the context, or even making it recoverable by the model, is not sufficient if the computation that selects and uses that information is brittle. Improving long-context memory may therefore require studying not only how models store or retrieve state, but also how retrieved state is routed into the computation that determines the next action or update.

AI USE STATEMENT

We used AI-assisted tools for literature search and related-work discovery. Large language models were also used as research subjects in experiments designed to evaluate the scope and generality of our method beyond the primary experimental setting. AI tools were not used to autonomously determine the scientific conclusions of this work. The authors reviewed and verified the final manuscript, experimental results, and cited references, and take full responsibility for the content of the paper.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide the implementation of the proposed method together with the code required to run the experiments. We also provide the benchmarks used in our evaluation, or instructions for obtaining publicly available benchmarks when redistribution is not permitted. The experimental configuration, including model architectures, training procedures, hyperparameters, random seeds, number of independent runs, evaluation protocols, and statistical procedures used to aggregate results and quantify uncertainty, is documented in the accompanying code and supplementary material. These materials include the configurations and scripts needed to reproduce the reported experiments and results.

REFERENCES

Lei Chen, Joan Bruna, and Alberto Bietti. Distributional associations vs in-context reasoning: A study of feed-forward and attention layers. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/533a89d7c2980218d82dbe3ea85f77ae-Paper-Conference.pdf.

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. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2207.02098.

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Babu Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. Context length alone hurts LLM performance despite perfect retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 23281–23298, 2025. doi: 10.18653/v1/2025.findings-emnlp.1264. URL https://aclanthology.org/2025.findings-emnlp.1264/.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10781.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020. doi: 10.1162/tacl_a_00306. URL https://aclanthology.org/2020.tacl-1.11/.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4475–4483, 2020. URL https://proceedings.mlr.press/v119/huang20f.html.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 21502–21521, 2024. URL https://proceedings.mlr.press/v235/jelassi24a.html.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. Same task, more tokens: The impact of input length on the reasoning performance of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15339–15353, 2024. URL https://aclanthology.org/2024.acl-long.818/.

David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: Compiled transformers as a laboratory for interpretability. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2301.05062.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *International Conference on Learning Representations*, 2023a. URL https://arxiv.org/abs/2210.10749.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Exposing attention glitches with flip-flop language modeling. In *Advances in Neural Information Processing Systems*, 2023b. URL https://arxiv.org/abs/2306.00946.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Dmytro Mishkin and Jiří Matas. All you need is a good init. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.06422.

Peter Shaw, James Cohan, Jacob Eisenstein, Kenton Lee, Jonathan Berant, and Kristina Toutanova. ALTA: Compiler-based analysis of transformers. *Transactions on Machine Learning Research*, 2025. URL https://arxiv.org/abs/2410.18077v2.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Sch"arli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227, 2023. URL https://proceedings.mlr.press/v202/shi23a.html.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024. doi: 10.1162/tacl_a_00663. URL https://aclanthology.org/2024.tacl-1.30/.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021. URL https://arxiv.org/abs/2104.09864.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 2021. URL https://proceedings.mlr.press/v139/weiss21a.html.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3770–3785, 2021. URL https://aclanthology.org/2021.acl-long.292/.

A TASK, CONSTRUCTION, AND PROOF

We study pushdown updates carried out through autoregressive context rather than an external runtime stack. The results separate representing a serialized state from learning the computation that accesses and updates it. A constructed Transformer implements the bounded update exactly. In controlled learning experiments, the final output rule is acquired reliably when analytic routing is provided, whereas learned components can remain imperfect or interfere with the downstream computation that uses their outputs.

End-to-end learning remains possible, but its observed training requirements increase with both stack depth and alphabet size. A separate Transformer trained from random initialization is evaluated through depth 64 with alphabet sizes $|\Gamma| \in \{1, 2, 4, 8\}$, extending the end-to-end study beyond unary stacks. Experiments with pretrained models provide a complementary result: under matched histories, obsolete context can alter an update even when the correct current state is explicitly available. Separate extraction often recovers the relevant state from histories in which the update failed, but this diagnostic does not reveal what information the model accessed during the original computation.

These learning results are restricted to bounded stacks, and the task-derived readouts do not establish autonomous search performance. Within this scope, they expose a concrete difficulty in using context as computational memory. Recording intermediate states in a transcript makes those states available, but does not enforce the operations needed to select and update them. Reliable in-context pushdown execution therefore depends not only on representing the required state, but also on learning how to access and transform it.

A.1 SERIALIZATION AND ARCHITECTURAL SCOPE

A legal trace begins with an empty stack and alternates supplied operations with complete states. For example,

$$\texttt{BOS [ ] PUSH A [A] PUSH A [A A] POP} \longmapsto \texttt{[A].}$$

Earlier states are reference states, not hidden corrections to generated answers. Every valid token remains visible under causal attention. The wrapper stops on the generated closing bracket and does not supply a correct output length or compute a stack boundary.

The construction uses constant, one-hot token, position, and squared-position coordinates. Three residual attention stages have respectively one, one, and three heads, with head dimension 16. A ReLU FFN and linear readout implement the output rule. The implemented width-64 graph has no LayerNorm. The empirical stagewise vocabulary has seven entries, including padding and BOS. Its position domain is $0, \ldots, 512$ and sharpness is $\beta = 64$; the larger construction domain in Proposition 1 is $0, \ldots, 8192$. The coordinate rescaling used for learning is invertible, with corresponding analytic-weight conjugation, rather than a normalization layer.

Table 1: The supplied interfaces in the stagewise graph. Roles describe the analytic solution, not all possible learned representations.

Group	Analytic role	Parameters
E	Token and constant coordinates	448
A_1	Latest opening position	4,096
A_2	Action position, source opening, operation indicators	4,096
A_3	Source, lookahead, PUSH argument	12,288
D	Open/copy/append/close decisions	8,704
Total		29,632

The source construction also permits a bounded alphabet that fits its coordinate and head dimensions. The stagewise acquisition experiments remain unary: every PUSH argument is A, so correct PUSH output does not by itself identify the argument-retrieval head. The separate end-to-end study in Appendix F evaluates heterogeneous stacks with $|\Gamma| \in \{1, 2, 4, 8\}$.

A.2 THE ANALYTIC OUTPUT RULE

The first attention stage scores a key at position p by $\beta(N+1)\mathbf{1}_{\text{open}} + \beta p$. The second uses the same form with an action indicator and reads the action position a, its saved opening position s, and operation indicators u (PUSH) and v (POP). Saving the opening *at the action* prevents a generated opening bracket from replacing the source boundary. The third stage reads positions $j = s+q-a-u$, $j+1$, and $a+1$, using scores $2\beta jk - \beta k^2$. Their equivalence to squared-distance scores follows after subtracting the query-only constant βj^2.

Let c_x, ℓ_x, and r_x be the retrieved source, lookahead, and argument coordinates for token x. The logit for the stack symbol A is the sum

$$\begin{aligned} z_A = \ & \mathrm{ReLU}(c_A + u - 1.5) + \mathrm{ReLU}(c_A + v - \ell_] - 1.5) \\ & + \mathrm{ReLU}(c_] + u + r_A - 2.5). \end{aligned} \tag{5}$$

The opening logit is $\mathrm{ReLU}(a+u-q+0.5)$ and the closing logit is 0.1. Other output logits are zero. In the ideal discrete selection limit, the correct opening or symbol has logit 0.5, while a required close wins at 0.1. Thus the minimum ideal winning margin is 0.1.

A.3 PROOF OF PROPOSITION 1

Marker selection. The latest eligible opening or action beats every competing key by at least β: the marker bonus dominates all positional differences, and distinct integer positions are at least one apart. With at most $N+1$ visible keys, nonselected softmax mass is at most $\rho = (N+1)e^{-\beta}$. The first stage's opening-position error is at most $N\rho$. Before any opening exists, its value is still a convex combination of positions in $[0, N]$, which bounds irrelevant values that the next imperfect selector may mix in.

The second stage therefore estimates the action position within $N\rho$, operation indicators within ρ, and source opening within $2N\rho$. The last bound combines first-stage error at the true action with second-stage leakage. Position and token coordinates remain intact. Consequently $|\widehat{j} - j| \le (3N+1)\rho = \delta$. The same bound applies to lookahead and to the PUSH argument address.

Address retrieval. For a required integer target j, a competing integer key k, and $e = \widehat{j} - j$, the desired-minus-competing score difference is

$$\beta\big[(k-j)^2 - 2e(k-j)\big] \ge \beta(1-2\delta).$$

Hence every retrieved token coordinate differs from its ideal indicator by at most $\tau = (N+1)e^{-\beta(1-2\delta)}$. Relevant source and lookahead targets are visible through the query that emits the close. At PUSH completion they reach the supplied action and argument. Empty POP closes correctly. The argument head need not retrieve a correct token on POP: the append branch is inactive since its input is at most $1 + \rho + 1 - 2.5 < 0$.

Decoding and induction. ReLU is 1-Lipschitz. Errors in the symbol's three branches sum to at most $5\tau + 3\rho$; the opening logit's error is at most $(N+1)\rho$. Thus every output coordinate differs from its ideal value by at most $E = \max\{(N+1)\rho, 5\tau + 3\rho\}$. Every winning margin remains at least $0.1 - 2E > 0$. The opening is therefore correct, and a correct generated prefix is followed by the correct next token. Induction proves the entire next stack. Another legal supplied action reestablishes the premise, provided the complete trace stays in the position domain. For $N = 8192$ and $\beta = 64$, $\rho < 1.315 \times 10^{-24}$, $\delta < 3.230 \times 10^{-20}$, and $E < 1.078 \times 10^{-20}$.

This argument proves real-arithmetic decisions. It does not bound backend roundoff, quadratic-score cancellation, or every floating-point operation. A sufficient implementation error bound would be a uniform logit error η satisfying $E + \eta < 0.05$; no such universal backend certificate is asserted. Numerical checks of the analytic implementation report 75,649 passing next-stack checks across 79 overlapping suites and 313 near-boundary next-token checks. Those checks are neither exhaustive enumeration nor independent learning trials.

A.4 History invariance and influence

Correctness on every legal equivalent history implies decision-level invariance, even when attention distributions or hidden activations differ. A sufficient structural alternative is to cut *every* path from obsolete history to the next-token logits in the unrolled computation graph. Cutting only direct output-to-history attention does not remove indirect paths through history-conditioned state, action, or earlier output representations.

A quantitative version follows from the margin. If each equivalent-history replacement changes every logit by at most ϵ and the original correct-token margin is $m > 2\epsilon$, greedy decisions are preserved. For one attention head, a useful decomposition is

$$\|o_{\text{full}} - o_{\text{clean}}\| \leq 2B\alpha + e_v + 2Be_s, \tag{6}$$

where α is attention mass on obsolete keys, values have norm at most B, local values change by at most e_v, and local scores by at most e_s in infinity norm. Remove the old-value mixture at cost $2B\alpha$, substitute clean local values at cost e_v, and use $\|\operatorname{softmax}(s) - \operatorname{softmax}(t)\|_1 \leq 2\|s - t\|_\infty$ for the final term. Multi-layer propagation must additionally account for projections, residuals, FFNs, and any normalization.

This explains why raw attention mass is insufficient: small direct attention leaves the other two terms uncontrolled, and a large mass can carry little harmful value. It is consistent with the distinction between key bias and value contribution in attention-sink work (Gu et al., 2025). These are sufficient mathematical conditions, not measured global robustness bounds for the learned models.

B Related work

Representing and learning algorithms. Formal-language analyses characterize Transformer expressivity under different assumptions on depth, precision, and attention (Hahn, 2020; Yao et al., 2021; Strobl et al., 2024). Empirical hierarchy studies and automata simulations further show that representational capacity does not by itself determine what is learned from data or how well the learned computation generalizes (Delétang et al., 2023; Liu et al., 2023a). Closest to our progression, Jelassi et al. (2024) connect a constructive copying algorithm to synthetic learning experiments and retrieval in pretrained models. We study an explicit state transition instead of copying, and use component resets to separate learning the routing computation from learning the final update rule.

Constructed computations and internal mechanisms. RASP, Tracr, and ALTA use constructed Transformer computations as controlled objects for analysis (Weiss et al., 2021; Lindner et al., 2023; Shaw et al., 2025). Our stagewise experiments use the same general idea for a different question: we replace selected parts of a supplied stack-update computation and test whether the learned component composes correctly with the fixed remainder. Related work also separates computations associated with attention and feed-forward layers. Chen et al. (2025) study distributional associations and in-context recall using controlled models and pretrained-model ablations, while Gu et al. (2025) show that attention scores alone need not reflect informative value contribution. Our analysis similarly distinguishes a locally useful signal from its downstream effect, using explicit interventions on the learned residual update.

Context availability and use. Long-context studies show that model performance can degrade when relevant information is displaced, surrounded by distractors, or embedded in longer inputs (Liu et al., 2024; Shi et al., 2023; Levy et al., 2024). Du et al. (2025) further distinguish successful retrieval from downstream task performance, and flip-flop language modeling exposes related failures when models must preserve state across intervening inputs (Liu et al., 2023b). Our pretrained-model contribution is a matched-history evaluation of in-context pushdown updating with an exact transition oracle. Within each group, we hold the current state, requested action, recent suffix, suffix positions, and total token length fixed, varying only the obsolete history. This design tests whether irrelevant history changes the predicted transition when the correct answer remains unchanged. We use separate extraction prompts as an additional measure of state accessibility, not as a causal readout of the original update computation. Together with the constructive and learning experiments, this evaluation studies whether a serialized stack functions as usable computational memory, rather than only as recoverable information.

C TRAINING, INITIALIZATION, AND EVALUATION PROTOCOL

We distinguish four evidence roles throughout this appendix. Hyperparameters for the primary stagewise matrix are selected using the development study described in Section B.1. The subsequent confirmation panel uses newly generated seeds and data streams after those recipes are fixed. The interface interventions are exploratory post-hoc analyses of selected checkpoints and are labeled accordingly. The depth- and alphabet-dependent end-to-end runs use a separate architecture and training protocol, described in A.G. No confirmation-panel outcome is used to retune the stagewise recipes.

C.1 SELECTION AND OPTIMIZATION

The pilot considers depth 2, conditions A_1, A_2, A_3, and Full, four initializer families, three learning rates, a single initialization seed, and 32 training queries generated with a shared data seed. Each of the 48 configurations receives 512 updates. For each initializer family, selection maximizes mean final development accuracy across the four conditions; ties are broken by mean worst development margin and then by the smaller learning rate. This procedure selects Gaussian 10^{-4} and 10^{-3} for the other families. The best pilot scores on the fixed 402-query evaluation suite, which are not used for method selection, are 175/402, 249/402, 261/402, and 231/402 for Gaussian, fan-in, orthogonal, and small-residual, respectively.

The primary experiment uses newly generated training batches with the selected family-level learning rates. All primary configurations retain the pilot initialization and data-generation seeds. We use AdamW with zero weight decay, batch size 32, gradient clipping at 1, and a 16-step warmup. Models are evaluated at updates 0, 512, 2,048, and 8,192 on the fixed training queries, development set, and fixed evaluation suite. At each checkpoint, we record complete model and optimizer states, predictions, reproducibility checks, diagnostics, and parameter-update path lengths. We report final exactness separately from the first scheduled evaluation at which exactness is attained.

The experimental protocol and accompanying code specify the random seeds and implementation details. We perform no per-run tuning based on the confirmation panel. Initializer calibration receives input-prefix tokens, positions, and padding masks, but not next-stack targets or teacher-forced focal output tokens. Earlier explicit states remain in the input because they define the task.

C.2 FOUR INITIALIZATION FAMILIES AND STRUCTURAL CONTROLS

The Gaussian and fan-in/input-RMS families retain the reference recipes from the earlier study. Orthogonal initialization combines orthogonal projection geometry with input-only scale calibration. We measure centered score dispersion across valid causal keys so that a softmax-invariant rowwise offset is not counted as useful score variation. The primary targets, dispersion 1 and residual/input RMS ratio 0.1, are fixed in advance. All six combinations of dispersion $\{0.5, 1, 2\}$ and ratio $\{0.1, 0.3\}$ are measured as input-only diagnostics and are not selected using target accuracy. The relation to LSUV is methodological inspiration rather than an identity of algorithms (Mishkin & Matas, 2016).

Small-residual initialization allocates a variance budget across the four residual additions. It is inspired by normalization-free optimization work (Huang et al., 2020), but does not introduce LayerNorm or inherit a theorem for the original T-Fixup architecture. These alternatives test initialization schemes beyond additional seeds of a single Gaussian recipe. Their failures do not exclude other scales, parameterizations, curricula, or optimizers.

We evaluate compiler-informed controls separately. Independently shuffled analytic entries preserve marginal weight statistics; support-based initialization preserves canonical nonzero locations; and support-and-sign initialization additionally supplies coefficient directions. The selected entries remain trainable, so these conditions modify initialization rather than impose permanently masked architectures. We also attempted a random-support control using up to 100 candidate initializations, screened using input prefixes without target labels. For each candidate, we differentiated the mean squared output logits with respect to the trainable parameters. Acceptance required every trainable parameter tensor with at least two dimensions to have a computed gradient containing at least one nonzero entry. None passed this check, so this condition is excluded from the completed training trials rather than counted as an unsuccessful training run.

C.3 SELECTIVE PROGRAM-TRACE OBJECTIVE

Conditions trained only on internal targets do not additionally optimize next-stack cross entropy. Their local objective combines defined pointer-attention cross entropy, token-unit pointer MSE divided by 16, relevant categorical-coordinate MSE, and 0.1 times the mean off-template-coordinate MSE. The off-template term uses the union of defined stage pointer masks. A_1 supervises action-token positions with a visible opening; A_2 supervises focal decision positions; and A_3 supervises branch-relevant source, lookahead, and argument retrieval. Undefined or irrelevant labels are excluded.

This objective does not reconstruct the full residual vector at every position. In particular, the off-template diagnostic reports all-valid-position RMS, whereas the loss applies only to a selective union of masks. A low or zero masked loss therefore does not certify unobserved positions. The analytic teacher and internal labels are used only in this explicitly labeled training arm. Teacher forwards and internal labels are counted separately from query output tokens. For conditions trained only on internal targets, the output-token counter refers to the associated next-stack queries and does not represent an additional output loss.

C.4 CONFIRMATION DESIGN

The selected recipes are decoder/Gaussian, A_3/small-residual, Full/fan-in, and Full/Gaussian. At depths 1 and 2, each recipe is trained with five new initialization seeds crossed with two new data-generation seeds. Data sequences are paired across recipes. Method selection is fixed before generating the new evaluation panel, using development accuracy, worst development margin, and a predetermined family order to break ties. Previously used inputs from the fixed evaluation suite, development set, and reserved evaluation sets are retained for consistency checks and are not treated as new independent holdouts.

There are three legal state/action pairs at depth 1 and five at depth 2. For each pair, four histories are generated at nominal walk budgets 32, 48, and 64, followed by a repair suffix that reaches the intended state. Each base history is reused at offsets 0, 64, and its maximum safe offset, producing 108 and 180 queries. The maximum safe offset reserves space for the full generation cap rather than only the known gold output. Query identifiers are checked for overlap with earlier evaluation suites. Table 2 reports measured rather than nominal lengths.

Table 2: New-panel scope. Maximal target and capped-generation positions are both 512. Ranges summarize observed inputs rather than exhaustive coverage of every intermediate value.

Property	Depth 1	Depth 2
Base histories	36	60
Queries including three offsets	108	180
Actual history actions	32–65	32–66
Current depths	0–1	0–2
Prefix tokens	118–250	132–288
Observed absolute offsets	0–393	0–378

The planned extension to depths 4 and 8 is conditional on obtaining an eligible depth-2 attention-stage or Full checkpoint with exact performance. No checkpoint meets this criterion. We therefore do not evaluate deeper stagewise acquisition or extrapolation without further training. The separate end-to-end cohort in Appendix G uses a different graph and protocol. Decoder-only and depth-1 successes do not satisfy the extension criterion.

C.5 RESOURCE ACCOUNTING

The internally supervised conditions additionally account for 262,594,485 internal labels and 24,576 analytic-teacher forward passes. Training uses PyTorch 2.11.0+cu128 on an NVIDIA RTX 5090 with FP64 arithmetic. Summed run times include initialization and evaluation and may overlap when runs execute concurrently, so they do not represent total elapsed study time. Two duplicate executions consume an additional 16,384 updates and 524,288 examples but are excluded from the independent trials. Their recorded weights, optimizer states, and predictions match those of the corresponding

Table 3: Stagewise-study resources. Every scheduled training batch produces an optimizer update. Output tokens in the diagnostic row include associated query targets, not an additional cross-entropy objective for internal-only runs.

Phase	Cells	Updates	Examples	Output tokens
Pilot	48	24,576	786,432	2,211,840
Primary	24	196,608	6,291,456	17,618,784
Diagnostic	7	57,344	1,835,008	5,138,812
Compiler controls	3	24,576	786,432	2,202,348
Continued training	3	24,576	786,432	2,202,348
Perturbation recovery	9	73,728	2,359,296	6,607,044
Confirmation	80	655,360	20,971,520	53,829,080

retained runs. Equivalence checks confirm that changes to diagnostic logging leave the optimization procedure unchanged.

D Additional results and interpretation controls

D.1 Primary margins and failure categories

The final minimum margins on the fixed evaluation suite for the four decoder recipes are 8.5875, 8.2292, 11.5809, and 12.4687 in Gaussian/fan-in/orthogonal/small-residual order. The corresponding A_3 margins are -2.8232, -1.4167, -17.1143, and -24.198. High query accuracy can therefore coexist with a strongly incorrect rare decision. The failure taxonomy distinguishes wrong opening, early close, extra symbols, invalid content, and exhaustion of the generation cap without closing. Malformed outputs are not pooled with successful updates through a permissive post-hoc parser.

Table 4: Diagnostic and supplied-structure controls at depth 2. Counts are the final numbers of correct predictions on the 402-query evaluation suite, not rates over independent seeds. Internal-only and output objectives are distinct.

Arm	Recipe / supervision	Correct
$A_1 + D$	Fan-in / output	387
$A_2 + D$	Small-residual / output	392
$A_3 + D$	Small-residual / output	401
$A_1 + A_3$	Small-residual / output	383
A_1	Fan-in / internal-only	153
A_2	Small-residual / internal-only	151
A_3	Small-residual / internal-only	97
Full	Analytic histogram shuffled / output	48
Full	Analytic support / output	109
Full	Analytic support and signs / output	57

D.2 Continued training and recovery from parameter perturbations

We test whether continued training preserves an already correct computation. Starting from checkpoints that pass every query in the fixed evaluation suite, we train one attention stage while keeping the remaining stages analytic. After continuation with the output loss, the A_1, A_2, and A_3 conditions score 157, 0, and 402 out of 402, respectively. The initial checkpoints are correct on the suite, but continued training does not preserve this performance in all three conditions.

To test recovery from parameter perturbations, we add per-tensor Gaussian noise with standard deviation equal to the specified multiplier times the larger of the analytic tensor RMS and 0.02. At multipliers 0.01, 0.1, and 1, the realized whole-stage relative distances are approximately 0.00976, 0.0976, and 0.976 for A_1/A_2, and 0.0101, 0.101, and 1.012 for A_3. Final numbers of correct predictions on the fixed evaluation suite are $(1, 175, 0)$ for A_1, $(0, 0, 140)$ for A_2, and $(214, 221, 166)$

for A_3. None reaches exactness at a saved milestone. These conditions probe neighborhoods of a supplied algorithm rather than oracle-free initialization.

D.3 Position and history factors

Table 5 reports the paired shift analysis. Each paired unit consists of a fixed checkpoint and a fixed tokenized query evaluated at two offsets. The output-change count can exceed the sum of correctness flips because two distinct incorrect outputs can also disagree.

Table 5: Offset-zero versus maximum-safe pairs. C→W and W→C denote changes in correctness. Counts are descriptive; observations share histories, initialization seeds, and training-data sequences.

Condition	Depth	Pairs	C→W	W→C	Output changes
A_3/small-residual	1	360	172	0	172
A_3/small-residual	2	600	282	0	333
D/Gaussian	2	600	0	0	0
Full/fan-in	2	600	368	9	550
Full/Gaussian	2	600	70	38	272

At depth 2, A_3 correctness across actual-history bands 32–34, 48–50, and 64–66 is 435/600, 381/600, and 332/600. Across current depths 0, 1, and 2, the corresponding counts are 571/720, 419/720, and 158/360. These summaries mix position with other query characteristics and therefore do not isolate causal effects of history length or state depth. The decoder-only condition is exact in every stratum. At depth 1, A_3 scores 356/360, 343/360, and 184/360 across the three offset conditions, indicating brittleness even when the semantic task admits an action-only shortcut.

D.4 Local gradient analysis

We measure local gradients at fixed checkpoints without additional training. Table 6 reports selected measurements. "Increment" denotes the gradient norm with respect to the stage residual increment. Pointer gradients are expressed per one-token change in the canonical address rather than per normalized-coordinate unit. A measured zero may reflect saturation, underflow, or the particular local path evaluated; it does not establish a symbolic zero everywhere or the absence of all live parameter paths.

Table 6: Selected local gradients on saved training examples. The later attention-score gradients for A_3 are all measured zero in the listed A_1/A_2 conditions; other paths remain live.

Stage	Update	Output CE	Increment norm	Action grad.	Source grad.
A_1/fan-in	0	59.33	88.1	—	0
A_1/fan-in	8,192	0.02675	0.042	—	0
A_2/small-residual	0	1.884	3.31	4.58×10^{-3}	0
A_2/small-residual	8,192	0.000758	0.000311	1.37×10^{-8}	0

For A_1, the source-address column denotes the latest-opening position. The initial downstream A_2 score-gradient norm is 7.43×10^{-63}, and its final value is measured zero. Determining whether these local patterns cause the training failures would require a controlled optimization intervention rather than additional gradient visualization alone.

E Details of the exploratory interface interventions

The interventions in Table 9 are exploratory post-hoc analyses of selected checkpoints from the stagewise study, rather than additional component-learning conditions. The analyses use final internally supervised checkpoints for A_1/fan-in and A_2, A_3/small-residual, each at depth 2 with the same initialization and data-generation seeds as the primary experiment. Evaluation uses the original

402-query evaluation suite. This comparison involves no optimizer updates, checkpoint reselection, or confirmation-panel outcomes.

The interventions modify the stage output increment before its residual addition. The canonical coordinate projection is specified by the analytic model and is not fitted to evaluation labels. The analytic increment is computed from the same token input and upstream computation, while the unmodified complement retains the analytic weights. Whole-stage replacement therefore recovers the analytic positive control. The accompanying analysis records identify the checkpoints and intended coordinates and report all four intervention outcomes.

The three final internal checkpoints contain useful but imperfect canonical signals. For A_1, the latest-opening mean absolute error is 0.1406 tokens, with a maximum of 0.2546. For A_2, the action-position and source-opening mean errors are 0.2804 and 0.2917 tokens. The categorical scores for A_3 are perfect at the positions evaluated by the diagnostic, but source and lookahead pointer top-1 accuracy is only 33/45 and 13/18. Moreover, averaging incorrect pointer locations can produce an apparently accurate mean address. Thus, "accurate local signals" does not imply a fully correct interface.

Subsequent diagnostic evaluation of these three fixed checkpoints uses the original 32-query training set. We record input-derived pointer indices and relevance masks, continuous analytic-teacher targets, learned increments, attention distributions, and logits. The pointer counts reproduce the original diagnostics. We also record full residual-increment and attention-score derivative tensors for the four cases in Table 6: the output-supervised A_1/fan-in and A_2/small-residual models at updates 0 and 8,192, without additional training. Their output cross-entropies differ from the originally recorded values by at most 7.2×10^{-15}. We distinguish absent automatic-differentiation paths from computed gradients that equal zero. These analyses cover selected checkpoints rather than repeating all 48 original gradient evaluations.

The subsequent intervention evaluation retains unrestricted generation over the seven-token vocabulary, the original generation caps and positions, and dense FP64 computation. It records margins against all competing tokens under the gold prefix, ties, and paired changes from incorrect to correct outputs and vice versa. The intended coordinate sets are fixed before evaluation and applied at every valid position; padding remains unchanged. Continuous reference increments are computed from the analytic block using the actual upstream state. Numerical agreement between CPU and CUDA implementations is checked before evaluation, without requiring any predetermined intervention outcome. This evaluation uses PyTorch 2.11.0+cu130 on a 24-GB RTX PRO 6000 MIG slice, distinct from the original 2.11.0+cu128 training and 2.10.0 CPU review environments. Its outputs are independently validated and retained separately from the original evaluation records.

Three caveats limit the causal conclusion. First, removing a block of off-template writes does not localize a unique harmful coordinate or interaction. Second, patching every valid position is broader than the training masks, so canonical repair conflates precision and coverage. Third, a coordinate that is harmful to a frozen analytic complement could be useful after downstream coadaptation. The specific supported claim is that these off-template increments reduce end-to-end correctness in these checkpoints under this complement. A stronger general mechanism claim requires additional independent models, held-out intervention evaluation, and narrower causal contrasts.

F Complete stagewise acquisition outcomes

Table 7: Primary depth-2 acquisition after 8,192 updates, as correct sequences out of 402. Each entry is one run. All four decoder runs also pass all 15 development queries; no other row is exact at any saved milestone.

Reset/trainable	Gaussian	Fan-in	Orthogonal	Small-residual
A_1	139	372	344	314
A_2	170	378	141	395
A_3	297	401	397	400
A_{123}	252	386	374	367
D	402	402	402	402
Full	141	371	349	379

Table 8: Confirmation across five new initialization seeds and two paired data streams per row. Exactness on the original suites requires correct predictions on every query in both the development set and the fixed evaluation suite. New-panel exactness requires every history at all three offsets. Previously checked earlier candidates do not improve new-panel success counts.

Component	Initializer	Depth	Original-suite exact	New-panel exact
A_3	small-residual	1	9/10	0/10
A_3	small-residual	2	0/10	0/10
D	gaussian	1	10/10	10/10
D	gaussian	2	10/10	10/10
Full	fan-in	1	0/10	0/10
Full	fan-in	2	0/10	0/10
Full	gaussian	1	0/10	0/10
Full	gaussian	2	0/10	0/10

The decoder's first recorded exact milestone is 512 updates for three initialization families and 2,048 for Gaussian. The best primary A_3 run misses one early-closing query; its small-residual counterpart misses two by failing to close within the generation cap. These errors matter because a single erroneous transition can break a longer computation, although this experiment does not estimate an autonomous long-run failure rate.

The new depth-2 position comparison starts from 500/600 correct A_3 queries at offset zero and ends at 218/600 at maximum-safe. Outputs change in 333 pairs, including changes between wrong answers. The strict old-panel content-matching analysis has only 30 depth-2 model-pairs per method and is insufficient for a general causal history-content claim. This limitation is why the pretrained matched-history design holds geometry fixed explicitly.

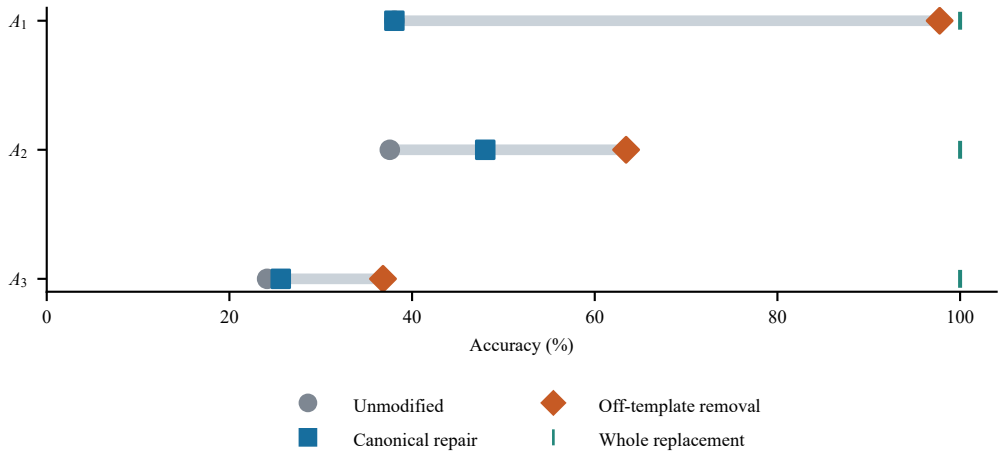

Figure 4: Exploratory interface interventions on three internally supervised final checkpoints. Off-template removal preserves the learned canonical coordinates and removes other writes. The analytic complement and the 402-query panel are fixed. Full replacement reinstates the supplied solution.

Table 9: Exact sequence counts for Figure 4, each out of 402. No retraining occurs.

Stage	Unmodified	Canonical repair	Off-template removal	Whole replacement
A_1	153	153	393	402
A_2	151	193	255	402
A_3	97	103	148	402

G DEPTH- AND ALPHABET-DEPENDENT END-TO-END ACQUISITION

G.1 EXPERIMENTAL GRID, MODEL, AND SUPERVISION

The end-to-end study uses a two-block, width-64, single-head rotary Transformer with FFN multiplier four, trained from random initialization in FP32 with TF32 disabled. We vary the maximum stack depth over

$$D \in \{1, 2, 4, 8, 16, 32, 64\}$$

and the stack alphabet size over

$$|\Gamma| \in \{1, 2, 4, 8\}.$$

For $|\Gamma| > 1$, different symbols can occur at different positions within the same stack. The larger-alphabet conditions therefore require the model to preserve symbol identity and order, rather than only the current stack depth.

For a stack $S \in \Gamma^{\leq D}$, the heterogeneous transition rule is

$$T(S, \mathrm{PUSH}(x)) = Sx \qquad (|S| < D,\ x \in \Gamma),$$

and

$$T(Sx, \mathrm{POP}) = S,$$

with POP on the empty stack treated as a no-op. The model generates the complete next stack rather than a depth label or an action.

The role-stratified generator produces legal 128-operation traces, and all 128 next-state spans receive causal, token-normalized cross entropy. Earlier actions and complete stack states remain in context throughout training and evaluation; the model is not supplied with a cropped latest state or an external stack. Earlier action tokens serve only as context rather than as an additional action-prediction objective. There is no margin term or history penalty in this cohort.

A trace, a supervised state transition, an output token, and an optimizer step are distinct accounting units. Each trace supplies 128 supervised state transitions. The logical batch contains 512 traces. Physical microbatches contain 256 traces at lower depths and 128 traces at higher depths, with accumulation preserving the same logical batch size. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and weight decay 0.01. The learning rate warms up to 10^{-3} over 32,768 traces and then remains constant unless a reported continuation activates the stabilization procedure described below. The EMA half-life is 4,096 traces.

We use three initialization/data seed pairs: $(794843687, 541169806)$, $(1034630086, 2561209270)$, and $(3362832204, 2472856037)$. The same seed identities are used across the depth–alphabet grid so that the reported curves compare corresponding training lineages rather than unrelated seed sets. GPU execution used RTX PRO 6000 24-GB slices; the records bind software and source versions separately from the older FP64 stagewise study.

Evaluation combines 2,048 random queries with role, boundary, and structured coverage, together with four generated rollouts of at most 128 operations. Complete generated stacks are scored by exact sequence match, so a prediction is correct only when its depth and every retained or appended stack symbol are correct and appear in the correct order.

The acquisition event is the first chronologically recorded EMA checkpoint that passes the *entire* finite measurement suite and all four rollouts. Because evaluation occurs on a checkpoint grid, this is a discretely observed first-pass time rather than the exact instant at which the model first becomes capable of passing the suite. Zero observed errors certify only the reported finite evaluation and do not establish correctness on every legal history in the corresponding $(D, |\Gamma|)$ domain.

G.2 ACQUISITION COST ACROSS DEPTH AND ALPHABET SIZE

Figure 2c reports the number of training traces consumed before the first complete measurement-suite pass. The observed training requirement generally increases with stack depth and becomes substantially larger as the alphabet grows. The alphabet intervention changes the semantic content of the task: in the unary condition, the stack is determined completely by its depth, whereas for $|\Gamma| > 1$ two stacks at the same depth can contain different ordered symbol sequences. A correct update must

therefore preserve and transform the stack contents rather than only produce the correct number of symbols.

This distinction is particularly important for POP. For example,

$$T([A, B], \mathrm{POP}) = [A], \qquad T([B, A], \mathrm{POP}) = [B].$$

The action and stack depth are identical in the two cases, but the required outputs differ. Similarly, $\mathrm{PUSH}(x)$ must preserve the entire existing prefix while appending the supplied argument. The larger-alphabet conditions therefore extend the end-to-end experiment beyond unary depth tracking.

For conditions in which all three seeds reach the criterion, Figure 2c shows the mean first-pass cost with ± 1 sample standard deviation. The shading describes variation across the three training runs and is not a confidence interval. Runs that do not reach the criterion within their allocated budget are treated as budget-limited observations rather than as if their first-pass cost were equal to the budget.

The resulting curves should be interpreted as empirical acquisition-cost trends under the tested architecture, optimizer, data distribution, and finite exactness criterion. They do not establish an asymptotic sample-complexity law or a scaling exponent in either D or $|\Gamma|$.

G.3 Continuation protocol and accounting

Some higher-cost training lineages use the previously defined stabilization continuation rather than a restart from new weights. For those runs, continuation begins from a saved full training state and preserves model weights, EMA weights, optimizer moments, random-number generator state, sampler position, and cumulative training counters. After the first strict training-cross-entropy crossing below 0.01, the continuation activates global gradient clipping at 0.1 together with inverse-square-root learning-rate decay. Once activated, the flag remains enabled even if the training loss subsequently rises.

Figure 2c counts all training traces consumed by each reported run before its first finite-suite pass, including training before any continuation. Discarded exploratory runs are not included. The continuation procedure was not evaluated as a randomized intervention.

Where a run reaches its maximum allocated training budget without passing the complete measurement suite, the table reports the result as right-censored at that budget. Such a run is not assigned a numerical first-pass cost for computing an ordinary mean. This distinction prevents a training budget from being conflated with an observed acquisition event.

G.4 Finite-suite acquisition and optimization stability

First-pass acquisition, independent confirmation, and subsequent optimization stability are separate properties. Figure 2c uses the first complete measurement-suite pass as its acquisition event. A model can reach this criterion and later regress, particularly in runs that exhibit training-loss spikes. Such later behavior is retained in the run records and is not erased by stopping the acquisition-cost clock at the first passing checkpoint.

Independent confirmation on separately generated queries is likewise not used to select the first-pass checkpoint. Confirmation results from the depth-only experiments do not validate the added depth–alphabet conditions unless those conditions are independently evaluated using the same confirmation protocol.

Consequently, the expanded experiment establishes that end-to-end acquisition is possible on finite bounded stack domains, while the observed training cost becomes substantially larger as the required stack depth and alphabet size increase. It does not establish exhaustive correctness over all legal histories, uninterrupted stability after the first pass, or an asymptotic law for pushdown learning.

H Pretrained-model protocol and complete core results

H.1 Model and inference controls

The pretrained-model evaluation uses Qwen3-8B and Qwen3-32B with pinned tokenizers and thinking-disabled native chat templates. The saved configuration uses BF16, dense SDPA, batch one,

a fresh cache per query, greedy decoding, and a 128-token output cap within a 32,768-token context budget. Model generation defaults are overridden by the recorded greedy policy. Original prompts, exact token lengths, generated token IDs, termination flags, and gold-prefix margins are retained. No quantized model is substituted. The pretrained-model experiments ran on RTX PRO 6000 GPUs.

Token-boundary checks verify the relationship between rendered prompts and continuation tokenization. Short dense/cache and backend comparisons are engineering controls, not additional scientific samples. Some BF16 logit differences exceed numerical tolerance despite identical generated IDs. Likewise, uniform position shifts preserve the short control outputs without certifying bitwise or arbitrary-length invariance.

For fixed-frequency RoPE without additional absolute-position dependence, $(R_{i+c}q)^\top(R_{j+c}k) = q^\top R_{j-i}k$ in real arithmetic (Su et al., 2021). A global position shift is therefore distinct from replacing history tokens or changing relative distances. The stagewise model's engineered positional failure cannot be transferred to these LLMs by analogy alone.

H.2 Eligibility, group geometry, and scoring

Minimal and boundary controls are evaluated separately at each capacity. Eligible 8B capacity 2 passes 10/10 combined controls; eligible 32B capacity 8 passes 34/34. The 8B capacity-8 controls score 25/34, while 32B capacity 2 scores 9/10 and remains ineligible under the prespecified eligibility criterion. Those failures are retained rather than retrospectively changing eligibility to favor an outcome.

In the content panel, four histories share current depth, requested operation, recent suffix, token length, and suffix positions. A group is the natural unit for an invariance contrast; its four responses are correlated. The length panel uses progressively extended histories, so changing the band also changes content and relative geometry. The two panels are not pooled for a causal length-only claim. Exact same-query joins are used only for the separately prompted extraction comparison.

Correctness scores the complete next stack. Disagreement compares actual answers and differs from mixed correctness: two distinct wrong stacks count as disagreement even though both are incorrect. Figure 3 uses groups containing different *valid* stack answers, with all groups in the denominator. Table 10 additionally reports the original all-response metric, which can include malformed outputs. At 0.5K, the 32B panel has six invalid queries, and valid-only disagreement is 4/340 rather than the original 7/340. At 32K all answers in both eligible strata are syntactically valid, so formatting cannot explain the reported disagreements there.

Table 10: Eligible matched-history results. Correct and valid columns count queries; disagreement and mixed columns count groups. "Valid disagreement" requires at least two different valid stack answers. "All disagreement" retains the original parser outcome for malformed responses.

Stratum	Band	Correct	Valid disagreement	All disagreement	Mixed correctness	Valid
8B, $D = 2$	0.5K	329/400	30/100	30/100	30/100	400/400
8B, $D = 2$	2K	276/400	20/100	20/100	20/100	400/400
8B, $D = 2$	8K	242/400	2/100	2/100	2/100	400/400
8B, $D = 2$	32K	236/400	58/100	58/100	57/100	400/400
32B, $D = 8$	0.5K	1350/1360	4/340	7/340	7/340	1354/1360
32B, $D = 8$	2K	1359/1360	1/340	1/340	1/340	1360/1360
32B, $D = 8$	8K	1204/1360	68/340	68/340	67/340	1360/1360
32B, $D = 8$	32K	1112/1360	74/340	74/340	74/340	1360/1360

H.3 Extension and fresh-seed results

The eligible 8B length panel scores 85/100, 68/100, 61/100, and 60/100 at 0.5K, 2K, 8K, and 32K, respectively. The corresponding 32B results are 338/340, 340/340, 295/340, and 275/340. These results agree qualitatively with degradation at the longest context but do not distinguish added length from added content.

On the separately generated 8B mechanism-panel baseline, 2K and 8K accuracy is 279/400 and 243/400, with 15/100 and 3/100 disagreeing groups. The original corresponding counts are 276/400 and 242/400, with 20/100 and 2/100 disagreeing groups. This is descriptive replication on fresh

groups, not a statistical equivalence test or extra samples to pool into the original panel. An accuracy near 60% also does not itself prove an action-only strategy: direct agreement with an explicit shortcut is a different measurement.

Table 11: All saved controls and matched-history strata from the main pretrained-model evaluation, including those that fail eligibility. These are descriptive results, not an expansion of the primary eligible claim. "Disagree" is the original all-response group metric.

Model	Capacity	Panel	Token band	Correct	Disagree
32B	1	boundary control	control	3/3	0/3
32B	2	boundary control	control	4/5	0/5
32B	8	boundary control	control	17/17	0/17
32B	2	content	2048	397/400	3/100
32B	2	content	32000	307/400	14/100
32B	2	content	512	400/400	0/100
32B	2	content	8192	323/400	7/100
32B	8	content	2048	1359/1360	1/340
32B	8	content	32000	1112/1360	74/340
32B	8	content	512	1350/1360	7/340
32B	8	content	8192	1204/1360	68/340
32B	1	minimal control	control	3/3	0/3
32B	2	minimal control	control	5/5	0/5
32B	8	minimal control	control	17/17	0/17
8B	1	boundary control	control	3/3	0/3
8B	2	boundary control	control	5/5	0/5
8B	8	boundary control	control	9/17	0/17
8B	2	content	2048	276/400	20/100
8B	2	content	32000	236/400	58/100
8B	2	content	512	329/400	30/100
8B	2	content	8192	242/400	2/100
8B	8	content	2048	693/1360	139/340
8B	8	content	32000	153/1360	153/340
8B	8	content	512	932/1360	79/340
8B	8	content	8192	390/1360	134/340
8B	1	minimal control	control	3/3	0/3
8B	2	minimal control	control	5/5	0/5
8B	8	minimal control	control	16/17	0/17

H.4 Joint extraction and update outcomes

Separate extraction prompts ask for current stack and action. Table 12 gives their exact joint outcomes with the original update on the same capacity-2 query IDs. Each band has 400 matched-history queries and 100 extension queries. Joint outcomes are computed by matching the saved extraction and update responses for each query; no additional model inference is performed.

Table 12: Joint extraction/update counts underlying Figure 3c–d. Here extraction success requires both current-state and action extraction to be correct. Every row sums to 500.

	Extraction succeeds		Extraction fails	
Band	Update succeeds	Update fails	Update succeeds	Update fails
0.5K	412	86	2	0
2K	332	150	12	6
8K	273	172	30	25
32K	228	105	68	99

The 32K separation is weaker because extraction itself fails more often. Conversely, successful updates can coexist with extraction failure, which is possible because the prompts define different

computations. Neither direction establishes the content of the original update activations. The available-state claim is already grounded by the explicit input; the extraction result adds a separate measure of accessibility.

I ACTIVATION PROBES AND EXPLORATORY PATCHING

I.1 DECODABILITY UNDER FIXED PROBES

Activation features are collected on the ordinary update pass at layers 8, 17, and 26 of 8B. Fixed dual-ridge classifiers with regularization 1.0 predict current depth and action. Each band uses 40 training rows from ten groups and 400 held-out rows from 100 groups, with seed 26091524. The split is by group, not a held-out-length test. Majority, length-only, and group-shuffled-label decoders provide controls. Full candidate layer results are shown rather than selecting the best layer after seeing its score.

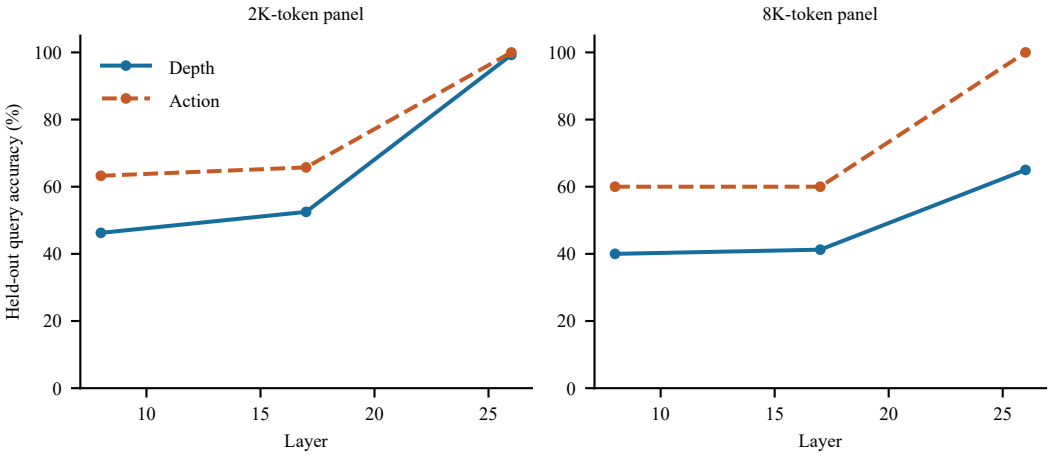

Figure 5: Decodability from activations on separate capacity-2 panels. Each point has 400 held-out rows sharing 100 groups. Depth becomes readily decodable at the last inspected layer at 2K, but substantially less so at 8K. An unsuccessful linear probe does not prove absent information; successful decoding does not prove causal use.

Table 13: All probe and control counts. These are fixed-model readouts rather than independent trained language models.

Band	Layer	Label	Activation ridge	Majority	Length only	Shuffled labels
2K	8	depth	185/400	160/400	160/400	145/400
2K	17	depth	210/400	160/400	160/400	154/400
2K	26	depth	397/400	160/400	160/400	133/400
2K	8	action	253/400	240/400	240/400	200/400
2K	17	action	263/400	240/400	240/400	192/400
2K	26	action	400/400	240/400	240/400	152/400
8K	8	depth	160/400	160/400	160/400	160/400
8K	17	depth	165/400	160/400	160/400	80/400
8K	26	depth	260/400	160/400	160/400	137/400
8K	8	action	240/400	240/400	240/400	239/400
8K	17	action	240/400	240/400	240/400	263/400
8K	26	action	400/400	240/400	240/400	254/400

The analysis uses saved activation features, held-out labels, and probe predictions. The fitted coefficient matrices were not saved, so the reported results summarize the saved predictions rather than a rerun of the original fitted probes. These experiments do not establish correct boundary localization or copying.

I.2 Directional patching does not establish selective repair

The exploratory activation-patching analysis contains 480 intervention records from 370 unique executions. Some executions appear under multiple intervention labels, including residual interventions assigned to different key/value (KV) groups. Rescue and reverse-corruption directions must be analyzed separately: averaging their signed net effects can cancel a useful or harmful intervention by construction.

The analysis of each patching direction finds no repair under the target matched-state patch condition. Its only nonzero repair records occur for a mismatched-state residual donor at layer 18, repeated across four KV-group intervention labels; corresponding reverse-corruption records also break a success. These repeated records do not constitute four independent repairs, and the mismatched donor effect does not establish selective restoration of the relevant state. These exploratory interventions establish neither a causal mechanism for routing the relevant state nor a validated method for correcting the updates. We did not conduct held-out confirmation of the patching results, test targeted repairs of attention and feed-forward components, or evaluate cross-task transfer. The matched-history results provide separate behavioral evidence of history-sensitive updating.

J Scalar readouts on task-derived memory histories

J.1 Readout semantics and scoring

Maze and Sudoku histories are generated by reference programs. The model does not select actions; it answers a scalar question at a designated checkpoint. In the maze, SAVE pushes a frame without replacing the last-returned register, while RESTORE pops a frame into that register. The still-suspended top frame is a separate object. Coordinates are zero-based. In Sudoku, ROLLBACK undoes writes since the latest outstanding savepoint without popping the savepoint, thereby preserving ancestor assignments.

For scalar evaluation, we use Qwen3-32B with the pinned revision above, BF16/SDPA, thinking disabled, and a `Value=` prefill. We score every candidate in the complete public domain of the queried field at a single prediction position after checking contextual tokenization. Expected answers are used only for evaluation. A query is correct only when the expected answer is the unique winner; ties count as unsuccessful. We record candidate logits, domain probability mass, and unrestricted vocabulary winners to distinguish ranking within the valid domain from unrestricted generation. C2 supplies the resolved oracle state and is therefore not a history-memory condition.

J.2 Explicit operations substantially reduce original-interface failures

The follow-up evaluates 40 maze and 40 Sudoku queries under three conditions: original wording (C0), explicit field and operation wording (C1), and oracle-resolved current state (C2). The resulting 240 readouts constitute a fixed diagnostic panel rather than independent model seeds. The C0-to-C1 comparison changes the interface as a package, including its representation and tokenization, rather than isolating the effect of any single wording change. It therefore tests whether failures under the original interface persist when the relevant state fields and operations are made explicit.

Table 14: Role-specific scalar correctness. Left: original query-ID counts. Right: distinct rendered prompts, retaining the same labels. Maze has six distinct prompts per role, so ten IDs do not constitute ten independent trials. C2 is oracle-assisted in both views.

		Query IDs			Unique prompts		
Task	Role	C0	C1	C2	C0	C1	C2
Maze	nested restore	2/10	10/10	10/10	1/6	6/6	6/6
Maze	single restore	5/10	10/10	10/10	3/6	6/6	6/6
Maze	still suspended	0/10	7/10	10/10	0/6	6/6	6/6
Maze	visible saved field	5/10	10/10	10/10	3/6	6/6	6/6
Sudoku	child after undo	10/10	10/10	8/10	10/10	7/10	8/10
Sudoku	child before undo	10/10	10/10	10/10	10/10	10/10	10/10
Sudoku	latest write	10/10	10/10	10/10	10/10	10/10	10/10
Sudoku	surviving ancestor	3/10	8/10	10/10	3/10	7/10	10/10

Explicit field and operation wording improves every role with a query-ID deficit under the original interface. In the maze, C1 increases nested restore from $2/10$ to $10/10$, single restore from $5/10$ to $10/10$, still suspended from $0/10$ to $7/10$, and visible saved field from $5/10$ to $10/10$. All corresponding distinct-prompt scores reach $6/6$. In Sudoku, surviving ancestor improves from $3/10$ to $8/10$ by query ID and from $3/10$ to $7/10$ over distinct prompts. Child before undo and latest write are already solved under C0 and remain at $10/10$ under C1, while child after undo remains at $10/10$ by query ID. Thus, every role with a query-ID deficit under C0 improves under C1, while roles already solved by this measure remain solved. The failures under the original interface are therefore sensitive to how the relevant fields and operations are expressed.

Providing oracle-resolved current state does not produce uniformly monotonic improvements. C2 reaches $10/10$ on all maze roles and on Sudoku surviving ancestor, child before undo, and latest write. Child after undo, however, decreases from $10/10$ under both C0 and C1 to $8/10$ under C2. This C2 error is therefore distinct from the corresponding original-interface failure: the role is already solved by query ID under C0 and C1, and the errors appear only under the oracle-assisted interface. Taken together, these controls weaken the interpretation of C0 failures as direct evidence of a native long-history memory deficit. Instead, much of the observed error depends on the interface used to express the relevant state and operations.