WHEN DO TWO HOPS SUFFICE? LEARNING REUSABLE COMPOSITION IN LOOPED TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning individual operations does not guarantee reliable composition. We study a synthetic task in which each operation maps one entity to another. Training uses all single-step examples and selected two-step answers, together with state targets derived from the same shallow labels, random input prefixes, and a self-supervised loss encouraging waiting-state preservation on constructed states. With one loop per operation, the resulting full-causal looped Transformer achieves 100% accuracy at tested depths of 256, 512, and 1024 and 99% at depth 2048. State interventions show that intermediate states can decode correctly yet fail as inputs to the next operation. These results show how learning to preserve usable states supports deep composition from shallow answer supervision.

1 INTRODUCTION

Compositional generalization makes learning reusable: a learner acquires atomic skills once and applies them in combinations it has never encountered. Learning each combination separately becomes impractical as the number of operations and their composition depth grow. People instead transfer newly learned operations to novel inputs and compositions, motivating a longstanding goal of systematic generalization (Fodor & Pylyshyn, 1988; Lake et al., 2017; 2019).

Neural models have made progress toward this goal, but reliable reuse remains a central difficulty. Meta-learning can induce human-like systematicity (Lake & Baroni, 2023), while controlled implicit multi-hop tasks can require rapidly increasing training data (Yao et al., 2025). Recurrent-depth Transformers improve composition, but recent experiments on compositional generalization to longer chains progressively add deeper answer supervision (Kohli et al., 2026). Circuit-specific Transformers provide shallow-to-deep solutions with structured routing (Ito et al., 2026). These results motivate a more specific question: *how can a full-causal looped Transformer learn reusable execution from answers that require at most two atomic operations?*

Our starting point is that reuse requires more than a correct local answer. A relation may be forgotten before its input arrives, or its result may be altered before the next operation consumes it. We express the desired computation through three state roles: *wait*, *execute*, and *hold*. Waiting positions preserve the relation needed for a future step, and completed positions retain the entity computed at an earlier step. We use these roles to define training objectives and to examine learned execution.

Trajectory-supervised chain-of-thought trains on correct length-L execution traces. Our answer labels stop at depth two. Anchors reuse them as embedding targets, providing soft shallow execution schedules and encouraging outputs to match subsequent operations' input interface. Constructed-state consistency (CS) trains preservation on random synthetic states representing deeper compositions; random entity prefixes provide positional coverage. Inference retains full-causal attention and continuous states. In a world with N_E entities and N_R atomic relations, the answer-label inventory is $N_E N_R(1+k)$ for k trained successors per relation: linear in N_R at fixed N_E, k, while test programs grow as N_R^d.

Our contributions are:

- **State preservation as a training objective.** Constructed-state consistency trains a shared loop to preserve waiting and completed states around an unconstrained execution frontier.

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.

Role ablations identify waiting-state preservation as the most effective variant in the main relational world.

- **Deep composition from shallow answers.** With waiting-state consistency, random prefixes, and state anchors reusing the shallow labels, a full-causal loop achieves 100% accuracy at depths 128, 256, 512, and 1024 and 99% at depth 2048, using answers of depth at most two (Table 1; Figure 2).
- **A mechanism separating atomic competence from reliable reuse.** Correctly decoded intermediate states can fail when consumed by the next operation. State interventions recover long-chain execution before natural generalization, while reduced drift accompanies its later emergence (Figures 4–7; Table 3).

2 SETTING

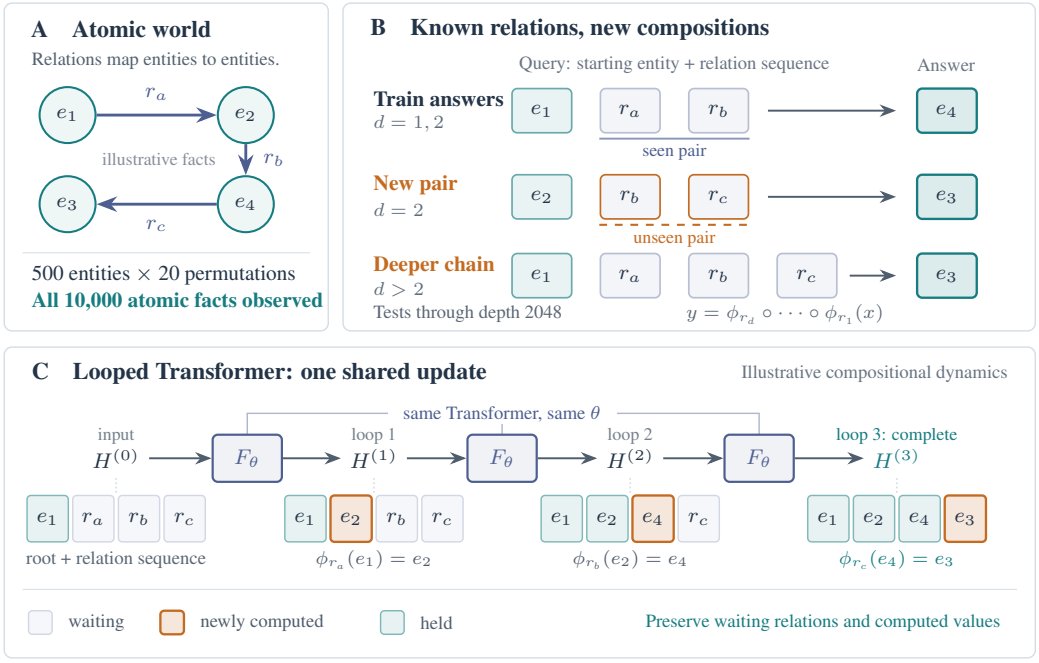


Figure 1: **Known skills, new compositions.** Atomic relations define a fixed world (A); tests recombine them and increase the number of successive applications (B). A shared Transformer can reuse the same wait–execute–hold update across loops (C). Intermediate entities are computed at inference.

2.1 COMPOSITIONAL GENERALIZATION AS REUSE OF ATOMIC OPERATIONS

Compositional generalization is the ability to evaluate new compositions of known atomic operations whose query–answer pairs were absent from training. We test new combinations and longer compositions of these operations; the focus is compositional generalization, rather than an isolated test of length generalization.

Let $\mathcal{E}$ be an entity set and each relation $r \in \mathcal{R}$ define an atomic map $\phi_r : \mathcal{E} \to \mathcal{E}$. A query $(x, r_1, \ldots, r_d)$ requires

$$v_0 = x, \qquad v_i = \phi_{r_i}(v_{i-1}), \qquad y = v_d. \tag{1}$$

The controlled world contains $N_E = 500$ entities and $N_R = 20$ random permutation maps. Training reveals all atomic facts and two-hop answers for $k = 8$ successors per relation: 10,000 atomic and 80,000 composed labels. Of the 400 ordered relation pairs, 160 are used for training, 20 for validation, and 220 for testing. Atomic knowledge is therefore fixed and fully observed; evaluation changes how it is composed (Figure 1).

Positional coverage is deliberate: random prefixes extend shallow inputs to 131 tokens, covering depth-128 queries (129 tokens). Evaluation extends compositional execution to depths 1024 and 2048 while answer labels remain at depth ≤ 2.

The main comparison uses a 1:1 mixture of training-only adjacencies (`all_seen`) and uniformly sampled relations excluding validation adjacencies (`random`). Each depth and seed uses 5,000 questions through depth 128; long-chain wait-CS uses 200 per distribution (400 pooled). Independent ablations also test exactly one held-out adjacency (`one_new`), as captioned. APIs use 40 mixed queries per depth. Looped inference uses $R = d$ (Appendix C.4).

2.2 Frontier models do not sustain reliable composition

We also give API models the complete atomic fact table in context. End-to-end accuracy declines on long chains (Figure 2). At depth 2048, Gemini 3.8 Flash (Google DeepMind, 2026) scores 25%, Claude Opus 5.5 (Anthropic, 2026) scores 5%, and GPT-6 Sol (OpenAI, 2026), DeepSeek-V4.1-Flash (DeepSeek-AI, 2026), and Muse Spark 1.3 (Meta, 2026) score 0%, each on 40 queries. Only Claude Opus 5.5 is affected by safety refusals; every non-refused Claude Opus 5.5 answer is correct. Our wait-CS loop reaches 100% at depths 256, 512, and 1024 and 99% at depth 2048. These results expose a practical reliability gap under different knowledge-access and inference protocols (Appendix C.5).

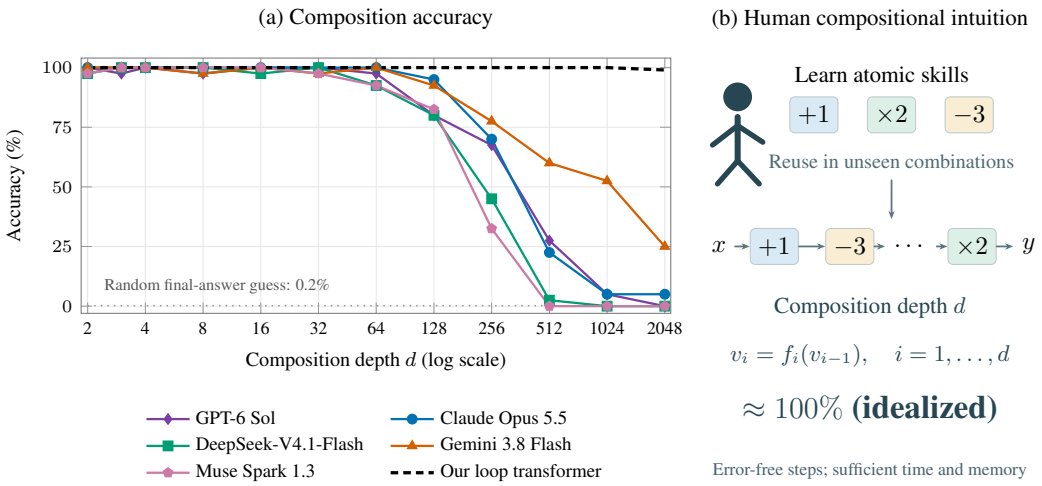


Figure 2: **API evaluation and the compositional ideal.** Left: five APIs and our wait-CS loop evaluated through depth 2048 (protocols in Appendix C.5). Accuracy scores the final entity; uniform guessing gives $1/500 = 0.2\%$ at every depth. Only Claude Opus 5.5 is affected by safety refusals, counted as failures. Right: human skill reuse as an error-free conceptual ideal, not measured performance.

3 Method

3.1 A shared loop as a reusable atomic computation

Human compositional learning motivates acquiring operations that can be invoked in new combinations (Lake et al., 2017; 2019). A looped Transformer offers a natural implementation: it repeatedly applies the same parameterized transformation (Dehghani et al., 2019; Kohli et al., 2026),

$$H^0 = \mathrm{Embed}(x, r_1, \ldots, r_d), \qquad H^{t+1} = F_\theta(H^t). \tag{2}$$

One loop can therefore reuse a learned atomic operation at the next position. Weight sharing makes this possible but does not enforce it: the model must preserve an operation before execution and its result afterward. We train this preservation behavior explicitly.

3.2 Waiting, execution, and holding

We distinguish three roles for a relation position: *wait* while its predecessor value is unavailable, *execute* its atomic relation when that value arrives, and *hold* the resulting entity for later positions.

These roles describe the desired behavior of continuous hidden states; they are not extra vocabulary tokens. The training objectives below encourage waiting relations and computed values to remain usable across loops. Role ablations, operand interventions, drift probes, and extended rollouts test the resulting learned behavior.

An architectural realization. A first learned implementation enforces predecessor-only routing and softly re-embeds the predicted entity distribution. Resetting each position's own input to its original relation embedding c_i gives the stateless recurrence

$$s_i^{t+1} = G_\theta(c_i, s_{i-1}^t), \qquad s_0^t = c_0. \tag{3}$$

This interface succeeds with shallow answer training in the tested depth range (Appendix D.2). Architectural and training details appear in Appendix B.1.

3.3 Learning state preservation

Can a standard full-causal loop learn related dynamics without enforcing that interface? Just looping with shallow answer cross-entropy (CE) stays near chance beyond two hops. The tested deeper CE and reward-training recipes also retain a depth boundary (Table 1). We therefore add direct training objectives for state preservation.

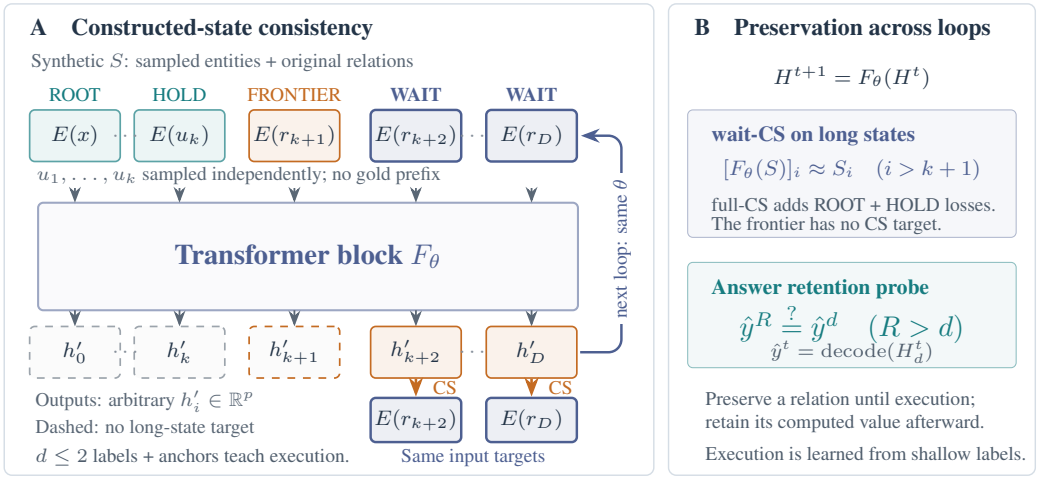


Figure 3: **Self-supervised constructed-state consistency (CS).** In A, E denotes the learned token embedding map. Synthetic input states serve as their own preservation targets. The frontier output remains free. Wait-CS constrains waiting states; full-CS also constrains root and hold states. Execution is learned from depth-one/two answers; anchors map those labels into the model's own embeddings.

For a synthetic chain, construct a detached state S with a root entity, independently sampled entity embeddings at completed positions, and relation embeddings at the remaining positions (Figure 3). The sampled entities are independent of the chain's correct prefix values. The executing frontier receives no CS target. Using a role mask $M_{b,i}$ and the same normalization N_{CS} across role ablations, minimize

$$\mathcal{L}_{\text{CS}} = \frac{1}{N_{\text{CS}}} \sum_{b,i \neq k_b+1} M_{b,i} D(F_\theta(S_b)_i, S_{b,i}) \,. \tag{4}$$

Here b indexes synthetic chains, k_b is the number of completed positions in chain b, and $k_b + 1$ is its frontier. D is normalized squared displacement (Appendix B.3), and N_{CS} counts all nonfrontier positions in the batch. Full-CS selects all nonfrontier roles; wait-CS selects positions after it. CS requires no answer labels: its preservation targets are copied from the synthetic input states, without relation-table lookups or computed intermediate values. Denoising extends this preservation objective to perturbed inputs, and walkback practices recovery over multiple loops; both retain the same shallow answer data (Appendix E).

Where the training targets come from. The objective is $\mathcal{L}_{\mathrm{CE}} + \lambda_A \mathcal{L}_{\mathrm{anchor}} + \lambda_C \mathcal{L}_{\mathrm{CS}}$. Answer CE uses labels of depth at most two. Anchors reuse v_1, v_2 as targets from the model's own detached embeddings. For relation i after the root, the target is its input embedding before loop i and its known entity embedding from that loop onward; roots and prefixes self-copy. These soft state and timing constraints encourage alignment with subsequent hops' input interface. They add no answer labels or forward-state replacements.

Training component	Answer labels used	Target or role
Answer CE	Depth ≤ 2	Final-position answer readout
State anchors	Reuse the same labels	Embedding targets for shallow states and timing
CS	None	Preserve selected synthetic-state positions
Random prefixes	None	Shift positions; self-copy loss belongs to anchors

Unlabeled positional augmentation. We prepend 0–128 random entity tokens to shallow queries to provide the positional coverage described in Section 2.1, while leaving each answer unchanged. Prefix self-copy targets are part of the anchor loss. All role ablations retain the same shallow labels, anchors, and prefix distribution. Standard input lengths and loop schedules are specified in Appendix B.

4 RESULTS

Two-hop labels support much deeper composition. With answer labels of depth at most two, our full-causal loop achieves 100% accuracy at depths 128, 256, 512, and 1024 and 99% at depth 2048—1024 times the maximum labeled depth (Table 1; Figure 2). CS and random prefixes supply self-supervised preservation and positional coverage, while anchors map the same shallow labels into the model's own embeddings. Inference uses frozen weights, full causal attention, and continuous states at every depth. Architectural alternatives and consistency variants are compared in Appendix D.2.

Comparison with API models and trained baselines. The API comparison in Section 2.2 shows declining contextual execution accuracy, with different training and inference budgets from the loop models.

The shallow answer-only loop and vanilla Transformer fit two-hop composition but fail on deeper chains. Adding shallow anchors and prefixes extends accuracy to 78.35% at depth eight, but only 0.09% at depth 128; wait-CS reaches 100% there under the same 200k-update budget. The three-seed anchor-only and four-seed wait-CS results are separate cohorts; within these tested recipes, shallow state and timing targets alone do not sustain deep composition. Final-answer supervision through depth 32 reaches 99.74% there, yet only 0.25% at depth 64. Reward training through depth eight extends competence to depth 16. At depth 256, Vanilla and both RL baselines score at most 0.33% on the balanced long-chain metric. Wait-CS demonstrates compositional generalization through depth 2048 with shallower answer labels than either of these deeper-training baselines. Baseline training and inference protocols are given in Appendix D.3.

5 ABLATION: WHICH STATES NEED CONSISTENCY?

Waiting is the simplest sufficient *additional* CS constraint in our experiments (Table 2). Wait-CS reaches 100% on both held-out-adjacency and random depth-128 chains in all four seeds; three seeds make zero errors across the full evaluation matrix and one makes nine. Removing the wait term is less reliable: root+hold fails, while root-only succeeds in two of four seeds. Waiting consistency is therefore useful without being necessary for every successful trajectory. These ablations remove long-state CS terms, not the root/wait/hold information in shallow anchors.

Stateless re-embedding and full-CS are compared with wait-CS in Appendix D.2.

Paired state interventions help locate these failures. Among historically correct executions with correctly decoded intermediate outputs, replacing the written state with one produced from canonical inputs rescues $19/45$ natural read failures at the 120k wait-CS checkpoint; cleaning only the reading context rescues $0/45$. In a root-only checkpoint, cleaning the reading context rescues 251 of 300 natural read failures. Across all 4,634 eligible events, supplying a correct predecessor leaves

(a)	Composition depth d									
Model / training	2		3		4		8		16	
API models: atomic facts in context										
GPT-6 Sol (OpenAI, 2026)	**100.00**		97.50		**100.00**		97.50		**100.00**	
Claude Opus 5.5 (Anthropic, 2026)	**100.00**		**100.00**		**100.00**		**100.00**		**100.00**	
Gemini 3.8 Flash (Google DeepMind, 2026)	**100.00**		**100.00**		**100.00**		97.50		**100.00**	
DeepSeek-V4.1-Flash (DeepSeek-AI, 2026)	97.50		**100.00**		**100.00**		**100.00**		97.50	
Muse Spark 1.3 (Meta, 2026)	97.50		**100.00**		**100.00**		**100.00**		**100.00**	
Trained models:										
Vanilla-4 + CE	99.91	(<0.01)	0.30	(0.01)	0.20	(0.01)	0.20	(<0.01)	0.22	(<0.01)
Loop + CE	**100.00**	(0.00)	0.20	(0.00)	0.17	(0.01)	0.17	(<0.01)	0.12	(0.00)
Loop + CE + anchor + prefix	**100.00**	(0.00)	**100.00**	(0.00)	**100.00**	(0.00)	78.35	(1185.72)	53.54	(2386.14)
Loop + CE ($d \leq 32$)	**100.00**	(0.00)	99.99	(<0.01)	99.95	(<0.01)	99.88	(<0.01)	99.92	(<0.01)
Loop + CE/RL ($d \leq 8$)	90.13	(0.04)	92.30	(3.55)	93.84	(4.23)	96.00	(6.64)	85.78	(123.87)
Loop + SFT→RL ($d \leq 8$)	89.89	(0.05)	91.99	(2.87)	94.03	(1.32)	96.66	(0.66)	84.28	(59.19)
Published loop objectives										
Recall + prog. loss (Bansal et al., 2022)	77.32	(4.39)	0.26	(<0.01)	0.14	(<0.01)	0.27	(<0.01)	0.16	(0.01)
NDR (Csordás et al., 2022b)	84.91	(0.73)	16.15	(54.61)	1.66	(2.62)	0.21	(<0.01)	0.23	(0.01)
Input injection (Fan et al., 2025)	85.89	(18.35)	0.76	(0.67)	0.17	(<0.01)	0.32	(<0.01)	0.23	(<0.01)
Dynamic recurrence (Kohli et al., 2026)	81.43	(24.84)	0.21	(<0.01)	0.19	(<0.01)	0.25	(0.01)	0.19	(0.01)
FPR mixing (Movahedi et al., 2026)	89.92	(0.33)	76.65	(1.90)	62.35	(2.57)	31.43	(56.67)	8.79	(30.35)
Our loop transformer	**100.00**	(0.00)	**100.00**	(0.00)	**100.00**	(0.00)	**100.00**	(<0.01)	**100.00**	(0.00)

(b)	Composition depth d									
Model / training	32		64		128		256		512	
API models: atomic facts in context										
GPT-6 Sol	**100.00**		97.50		80.00		67.50		27.50	
Claude Opus 5.5	**100.00**		**100.00**		95.00		70.00		22.50	
Gemini 3.8 Flash	97.50		**100.00**		92.50		77.50		60.00	
DeepSeek-V4.1-Flash	**100.00**		92.50		80.00		45.00		2.50	
Muse Spark 1.3	97.50		92.50		82.50		32.50		0.00	
Trained models										
Vanilla-4 + CE	0.19	(<0.01)	0.19	(<0.01)	0.17	(0.01)	0.33	(0.02)	0.00	(0.00)
Loop + CE	0.21	(0.02)	0.21	(0.02)	0.18	(<0.01)	0.15	(0.02)	0.3	(0.09)
Loop + CE + anchor + prefix	19.07	(489.33)	0.25	(0.05)	0.09	(<0.01)	0.00	(0.00)	0.00	(0.00)
Loop + CE ($d \leq 32$)	99.74	(0.01)	0.25	(<0.01)	0.25	(<0.01)	0.23	(<0.01)	0.00	(0.00)
Loop + CE/RL ($d \leq 8$)	0.15	(<0.01)	0.19	(<0.01)	0.15	(<0.01)	0.08	(0.02)	0.33	(0.15)
Loop + SFT→RL ($d \leq 8$)	0.00	(0.00)	0.00	(0.00)	0.00	(0.00)	0.00	(0.00)	0.00	(0.00)
Published loop objectives										
Recall + prog. loss (Bansal et al., 2022)	0.19	(<0.01)	0.19	(<0.01)	0.21	(<0.01)	0.42	(0.08)	0.75	(0.06)
NDR (Csordás et al., 2022b)	0.20	(0.01)	0.17	(0.01)	0.15	(<0.01)	0.08	(0.02)	0.33	(0.02)
Input injection (Fan et al., 2025)	0.18	(<0.01)	0.22	(<0.01)	0.25	(<0.01)	0.00	(0.00)	0.25	(0.06)
Dynamic recurrence (Kohli et al., 2026)	0.22	(<0.01)	0.18	(0.01)	0.21	(<0.01)	0.08	(0.02)	0.17	(0.02)
FPR mixing (Movahedi et al., 2026)	1.06	(1.20)	0.03	(<0.01)	0.03	(<0.01)	0.00	(0.00)	0.25	(0.19)
Our loop transformer	99.98	(<0.01)	**100.00**	(<0.01)	**100.00**	(0.00)	**100.00**	(0.00)	**100.00**	(0.00)

Table 1: **Main comparison across composition depths.** Accuracy (%). Trained rows report mean (variance across seeds): test set with 5,000 queries per seed through $d = 128$, and an equal mixture of `all_seen`/`random` at longer depths. Long-chain cells use 200 queries per distribution per seed (CE at $d = 256$: 1,000). All trained rows use 200k updates. APIs use 40 mixed queries per depth. Details: Appendix C.4.

257 failures, all removed by restoring the next relation operand. These are within-checkpoint diagnoses: the root-only and wait-CS probes use different model seeds, so their contrast is not a matched training ablation (Appendix F.4).

CS-constrained roles	Mean (%) $d = 128$	Sample variance
Root	50.11	3319.35
Root + hold	0.24	0.12
Root + wait	100.00	0.00
Root + wait + hold	100.00	0.00
Wait-CS	100.00	0.00

Table 2: **Waiting-state consistency.** Depth-128 `one_new` accuracy at 200k updates. All rows retain the full shallow anchor, backbone, prefixes, and CS normalization. Variance is in percentage points squared.

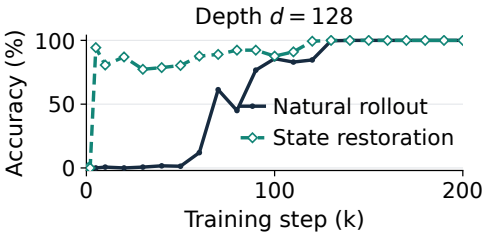


Figure 4: **State restoration.** Depth-128 accuracy on 300 chains. Between loops, reset waiting relations to input embeddings and project new predecessors to their nearest entity embeddings, without gold intermediate answers (Appendix F.6.1).

6 Analysis: From Atomic Competence to Reliable Composition

6.1 Does the learned loop reuse an atomic operation?

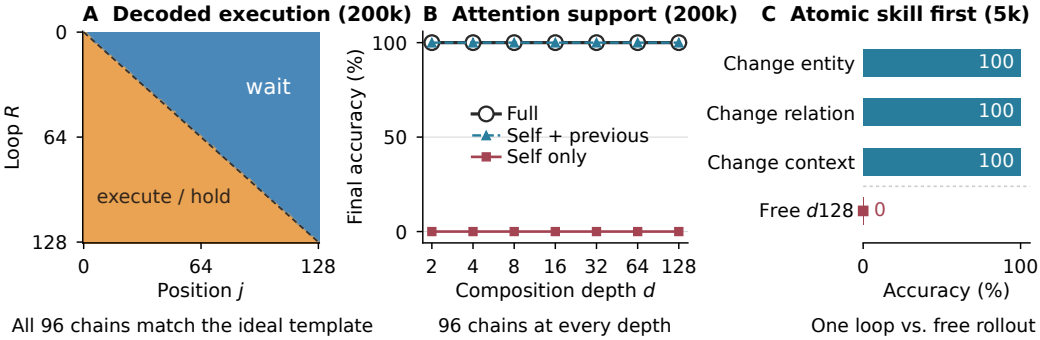


Figure 5: **Learned local composition.** Wait-CS. **A:** At 200k, all 96 depth-128 chains follow the decoded wait–execute–hold template. **B:** Self-plus-predecessor attention preserves 96/96 correct answers at every tested depth; self-only gives 0/96. **C:** At 5k, operand replacements already yield correct counterfactual one-loop outputs, while free depth-128 execution is 0/96. Protocol: Appendix F.2.

Figure 5A shows an advancing execution frontier with preserved waiting and completed positions in the learned full-causal model: all 96 tested depth-128 trajectories match the decoded template. Restricting every attention block to the current position and its predecessor retains perfect accuracy across tested depths, whereas self-only attention fails (panel B). The learned computation can therefore operate through a local communication path.

Starting from canonical correct internal states, we then intervene on the operands of one-loop execution. Replacing the predecessor entity or the frontier relation changes the output to the correct counterfactual result; replacing distant context preserves it (panel C). This supports causal dependence on the two operands of the atomic map. These interventions already succeed at 5k updates, when unconstrained depth-128 rollout answers none of the same 96 chains correctly. Knowing the local operation is insufficient: the model must repeatedly supply it with a usable state.

Execution also responds to operand readiness. At 120k, inserting a correct entity embedding at a future predecessor position advances execution from $0/300$ to $298/300$; reverting the just-computed predecessor to its original relation embedding suppresses the next execution from $288/300$ to $0/300$. At 200k, updates for the same entity–relation pair are nearly parallel across seven positions (mean cosine 0.9995). These probes support state-dependent execution and reuse across positions (Appendix F.5).

Correct decoding does not guarantee reuse. We write an intermediate state from canonical inputs, then supply it to the next operation in an otherwise canonical context. At 200k, CE-only correctly decodes all 4,800 written states, yet 2,085 fail when reused; supplying the correct entity

embedding instead succeeds in every case. Wait-CS succeeds on both tests (Table 3). Thus an answer can be semantically correct while its hidden representation is unsuitable for the next computation. This comparison tests the combined training recipe, since CE-only omits both anchors and CS.

Model	State decoded correctly	Next-step reuse	Gold-embedding control
CE-only	100.00	56.56	100.00
Wait-CS	100.00	100.00	100.00

Table 3: **Decoding versus reuse.** Accuracy (%) at 200k updates on 4,800 canonical write–read events (300 chains, 16 positions). The control replaces only the written state with the correct entity embedding. These fixed-run diagnostics are separate from the main accuracy evaluation; protocol in Appendix F.3.

6.2 Why does compositional generalization emerge late?

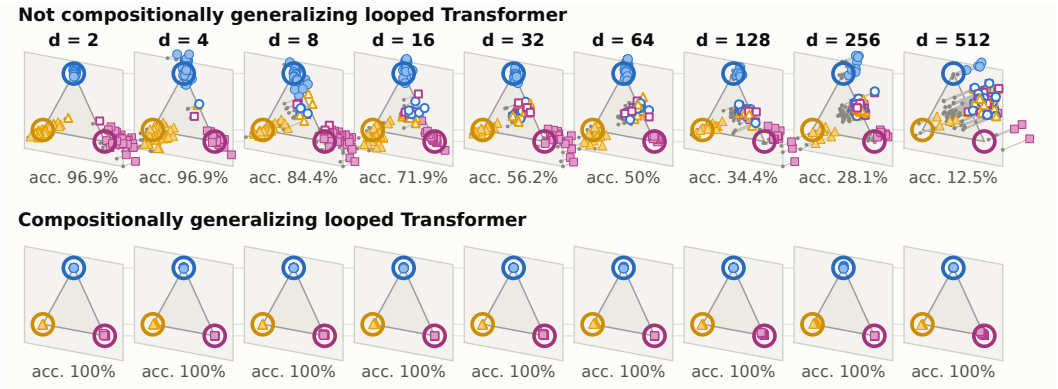


Figure 6: **Output states across composition depths.** The same wait-CS run at 60k (top) and 200k (bottom) training steps. Each card overlays 32 three-cycle chains: rings mark entity embeddings, and colored markers show the last three states after $R = d$ loops. Filled/hollow markers indicate correct/incorrect nearest-embedding decoding. Off-plane distance is compressed approximately $10\times$. Numbers report answer accuracy on these chains. Early states drift from their target embeddings as depth grows; late states remain close, with 100% accuracy through $d = 512$. Visualization details: Appendix F.1.

Figure 6 makes the contrast visible in a controlled cyclic-chain probe: at 60k, repeated execution takes states away from their target embeddings; at 200k, outputs remain close across all nine depths. This geometric contrast accompanies the emergence of reliable composition in the same run.

Figure 7 connects the training-step and loop-step axes. Shallow fitting and clean one-step execution precede reliable long rollouts. Between 120k and 130k updates, depth-128 monitoring rises from 84.67% to 99.67%, while waiting-state displacement at loop 64 falls from 0.961 to 0.285.[1] Tracking the same waiting positions throughout rollout also shows a $3.2\times$ decrease in drift-growth slope (Appendix F.6).

State restoration recovers long-chain execution before natural generalization (Figure 4). At 5k, projecting each newly computed predecessor to its nearest entity embedding and restoring waiting relations after every loop raises depth-128 accuracy from $0/300$ to $283/300$ (94.3%). At 120k, predecessor projection alone raises it from $254/300$ to $298/300$. These interventions use model-derived entity choices and the supplied relations, without gold intermediate answers; the repair locations follow an external schedule (Appendix F.6.1).

These measurements link delayed compositional generalization to reduced state drift across repeated updates. Correct atomic execution from clean inputs precedes the ability to preserve usable inputs

[1] Panel B measures raw displacement. In a separate probe, removing each token's coordinate mean leaves a decrease in the one-step hold residual from 0.1442 to 0.0156 over the same interval. This centered measure has a different reference and cohort (Appendix F.6.2).

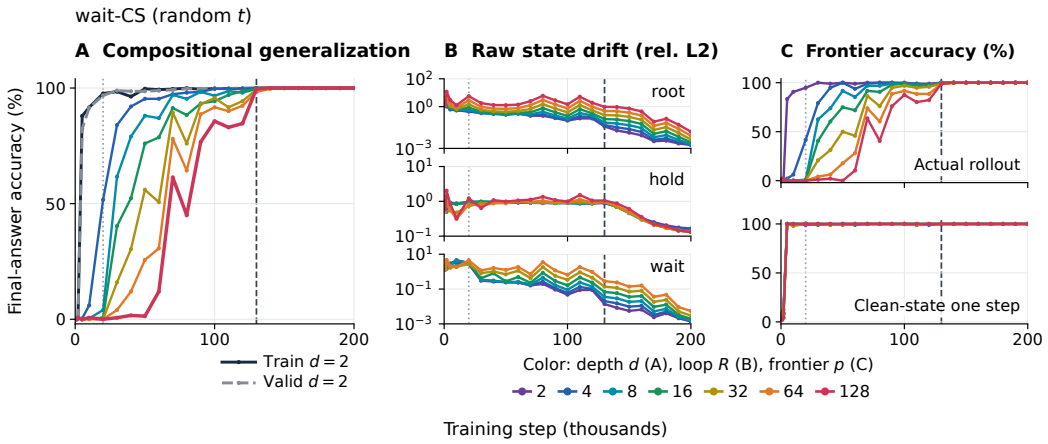


Figure 7: **Delayed compositional generalization and state drift.** Wait-CS. **A:** Natural accuracy across training checkpoints. **B:** Cumulative raw relative state displacement at different loop counts. **C:** Frontier accuracy from clean states versus continuous rollout. Vertical dotted: CS begins; vertical dashed: natural depth-128 monitoring stays above 99%. Definitions and probe counts: Appendix F.6.

throughout a long rollout. Training progress is not necessarily monotonic: multistep recovery can also deteriorate while a one-step residual improves (Appendix F.8.1).

Reaching an answer versus retaining it. Successful CS models preserve depth-128 answers through 256 loops on the tested trajectories. Completed reward-training runs can answer depth-16 queries accurately at 16 loops and then fall to nearly zero at 32 loops (Appendix D.3). Larger-world failures can also retain correct local execution while losing multistep recovery. Targeted denoising or multi-loop preservation training repairs several such failures (Appendix E). Together, these findings connect long composition to maintaining the conditions for repeated atomic execution.

7 Discussion and Conclusion

Correctly decoded states can still fail at the next operation. Interface repairs recover long-chain execution before natural generalization, connecting reusable computation to the preservation of usable representations.

Relation to prior work. Shared computation is central to Universal Transformers (Dehghani et al., 2019), and recurrent-depth models improve relation composition (Kohli et al., 2026). The Neural Data Router uses copy gates to preserve states until execution (Csordás et al., 2022b), while circuit evaluators use structured routing for shallow-to-deep execution (Ito et al., 2026). Our full-causal loop learns task-specific state preservation from shallow answers and synthetic states. Role ablations and operand interventions connect this learned preservation to reusable execution. Further comparisons appear in Appendix A.

Scope and limitations. Our experiments measure learned performance through depth 2048 in a controlled relational world. At fixed width, the full-causal model uses an externally supplied $R = d$: d loops through four blocks, with $O(d^2)$ dense attention cost per loop and $O(d^3)$ total cost. World-change results use full-CS and show that stronger denoising or walkback can recover deep composition with unchanged answer-training data, but preservation hyperparameters are not uniformly robust across worlds. Unchanged wait-CS has not been evaluated on every changed world. All reported accuracies use fixed training endpoints; selecting checkpoints from shallow validation alone remains an open practical question. Anchors reuse shallow answers while supplying a prescribed execution schedule as soft state targets. Extending this interface to tasks with non-chain dependencies remains open. The mechanistic probes measure preservation and local operand dependence on sampled trajectories; broader tasks and contexts remain to be tested.

ETHICS STATEMENT

This work studies compositional generalization in looped Transformers using a fully synthetic relational world of abstract entities (`E000–E499`) and relations (`R00–R19`). It involves no human subjects, personal or sensitive data, or web-scraped content, and it does not release deployed systems. The API-model evaluations send only this synthetic fact table and queries to commercial models under their standard terms of use. We do not foresee direct harmful applications; the findings concern how learned computation preserves and reuses intermediate states. We have read and adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

The task, relational world, and query distributions are defined in Section 2 and Appendix C.1. Architectures and training objectives, including the state-consistency losses, are specified in Section 3 and Appendix B; training schedules, seeds, and evaluation protocols appear in Appendices D, C.2, and C.4. The API-model prompts, settings, and budgets are documented in Appendix C.5, and checkpoint provenance in Appendix C.3. Multi-run comparisons report means and unbiased variances where available, with seed counts and evaluation protocols specified in the appendix. Anonymized code for generating the world, training the checkpoints, and running the evaluations, together with the API-evaluation code and all recorded model responses, is available at `https://anonymous.4open.science/r/loop-transformer-compositional-generalization-4C35`.

AI USE STATEMENT

The research ideas, methods, experimental design, and interpretation of results are the authors' own. We used large language models (LLMs) for two supporting purposes: polishing the language of the manuscript and assisting with code implementation. All authors reviewed, tested, and verified all AI-assisted text and code, and take full responsibility for the content of this paper. The API models evaluated in Section 4 are objects of study, not research assistants.

REFERENCES

Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J. Zico Kolter, and Roger B Grosse. Path independent equilibrium models can better exploit test-time computation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 7796–7809, 2022. doi: 10.52202/068431-0566. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/331c41353b053683e17f7c88a797701d-Abstract-Conference.html.

Anthropic. System card: Claude Opus 5.5. Technical report, Anthropic, September 2026. URL https://www.anthropic.com/claude-opus-5-5-system-card.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, pp. 690–701, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/01386bd6d8e091c2ab4c7c7de644d37b-Abstract.html.

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Extrapolation without overthinking. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20232–20242, 2022. doi: 10.52202/068431-1471. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/7f70331dbe58ad59d83941dfa7d975aa-Abstract-Conference.html.

Jay Bear, Adam Prügel-Bennett, and Jonathon Hare. Rethinking deep thinking: Stable learning of algorithms using Lipschitz constraints. In *Advances in Neural Information Processing Systems*, volume 37, pp. 97027–97052, 2024. doi: 10.52202/079017-3076. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/afbe068bd0469f4cd778c0f8106181b6-Abstract-Conference.html.

Jonathon Cai, Richard Shin, and Dawn Song. Making neural programming architectures generalize via recursion. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=BkbY4psgg.

Róbert Csordás, Kazuki Irie, and Juergen Schmidhuber. The devil is in the detail: Simple tricks improve systematic generalization of transformers. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 619–634, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.49. URL https://aclanthology.org/2021.emnlp-main.49/.

Róbert Csordás, Kazuki Irie, and Juergen Schmidhuber. CTL++: Evaluating generalization on never-seen compositional patterns of known functions, and compatibility of neural representations. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9758–9767, Abu Dhabi, United Arab Emirates, December 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.662. URL https://aclanthology.org/2022.emnlp-main.662/.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. The neural data router: Adaptive control flow in transformers improves systematic generalization. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=KBQP4A_J1K.

DeepSeek-AI. DeepSeek-V4.1-Flash: Pushing the limits of KV cache compression. Technical report, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4.1-Flash/blob/main/DeepSeek_V41_Tech_Report.pdf.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2edigk8yoU.

Jerry A. Fodor and Zenon W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1–2):3–71, 1988. doi: 10.1016/0010-0277(88)90031-5. URL https://www.sciencedirect.com/science/article/abs/pii/0010027788900315.

Google DeepMind. Gemini 3.8 Flash — model card. Technical report, Google DeepMind, September 2026. URL https://deepmind.google/models/model-cards/gemini-3-8-flash/.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning, 2026. URL https://arxiv.org/abs/2605.21488.

Takuya Ito, Ruchir Puri, Murray Campbell, and Parikshit Ram. Universal transformers for circuit computations: Perfect length generalization in tiny transformers, 2026. URL https://arxiv.org/abs/2608.31067.

Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9087–9105, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.731. URL https://aclanthology.org/2020.emnlp-main.731/.

Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. In *Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=8fz7WRThKL.

Asher Labovich. Stability and generalization in looped transformers, 2026. URL https://arxiv.org/abs/2604.15259.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2873–2882. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/lake18a.html.

Brenden M. Lake and Marco Baroni. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623:115–121, 2023. doi: 10.1038/s41586-023-06668-3. URL https://www.nature.com/articles/s41586-023-06668-3.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253, 2017. doi: 10.1017/S0140525X16001837. URL https://doi.org/10.1017/S0140525X16001837.

Brenden M. Lake, Tal Linzen, and Marco Baroni. Human few-shot learning of compositional instructions. In *Proceedings of the 41st Annual Conference of the Cognitive Science Society*, pp. 611–617. Cognitive Science Society, 2019. URL https://escholarship.org/uc/item/5vr9p4ks.

Adam Liška, Germán Kruszewski, and Marco Baroni. Memorize or generalize? searching for a compositional RNN in a haystack, 2018. URL https://arxiv.org/abs/1802.06467.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Meta. Muse Spark 1.3 evaluation methodology. Technical report, Meta, September 2026. URL https://research.meta.ai/static/muse-spark-1-3-multimodal-evaluation-methodology.

Sajad Movahedi, Vera Milovanović, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers, 2026. URL https://arxiv.org/abs/2606.18206.

OpenAI. GPT-6 Astra System Card. Technical report, OpenAI, September 2026. URL https://deploymentsafety.openai.com/gpt-6-astra. Section 11: GPT-6 Sol, GPT-6 Luna. Updated September 22, 2026.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 833–840, 2011. URL https://icml.cc/2011/papers/455_icmlpaper.pdf.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding, 2021. URL https://arxiv.org/abs/2104.09864v1.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1096–1103, 2008. doi: 10.1145/1390156.1390294. URL https://dl.acm.org/doi/10.1145/1390156.1390294.

Yuekun Yao, Yupei Du, Dawei Zhu, Michael Hahn, and Alexander Koller. Language models can learn implicit multi-hop reasoning, but only if they have lots of training data. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9684–9702, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.490. URL https://aclanthology.org/2025.emnlp-main.490/.

GLOSSARY OF ARCHITECTURES, TRAINING RECIPES, AND DIAGNOSTICS

The terms below distinguish architectural modes, training recipes, enlarged worlds, and evaluation interventions.

Term	Meaning	Details
Mode 2	Stateless predecessor-only computation with a fixed own-token input; communicates the original relation embedding plus the expected entity embedding.	App. B.1
Mode 3	The same stateless computation as mode 2, but communicates only the expected entity embedding.	App. B.1
R40, E1000	Enlarged worlds with 500 entities and 40 relations, or 1,000 entities and 20 relations, respectively.	Apps. E, F.8
State restoration	Diagnostic that projects completed model states to their nearest entity embeddings and restores root and waiting embeddings under an externally supplied execution schedule.	App. F.8.2

A EXTENDED RELATED WORK

Compositional generalization. SCAN and COGS evaluate systematic recombination through controlled distribution shifts, revealing generalization gaps that standard IID evaluation can miss (Lake & Baroni, 2018; Kim & Linzen, 2020). Performance on these benchmarks is sensitive to positional representations, embedding scaling, and training choices such as early stopping (Csordás et al., 2021). A closer task family is Compositional Table Lookup, which studies compositions of symbolic functions (Liška et al., 2018). CTL++ tests unseen combinations of known function groups and analyzes whether their learned representations remain compatible across compositions (Csordás et al., 2022a). For implicit relational reasoning, Yao et al. (2025) demonstrate multi-hop composition, with training-data and model-depth requirements increasing with hop count in their controlled setting. Our task separately tests unseen relation adjacencies and greater composition depth while keeping atomic knowledge fixed. Random entity prefixes diversify token positions during training.

Recurrence and control flow. Universal Transformers combine transformations shared across recurrent steps with adaptive computation at individual positions (Dehghani et al., 2019). Fan et al. (2025) show that looped Transformers can learn iterative solutions that generalize to longer algorithmic inputs. Kohli et al. (2026) demonstrate systematic compositional generalization, including to longer relation chains, using logit-lens probes and activation patching to analyze the role of intermediate entities. They also show that excessive recurrence can degrade predictions. The Neural Data Router addresses execution order through copy gates and geometric attention, allowing states to remain unchanged until they are ready to be processed (Csordás et al., 2022b). Our full-causal loop learns state preservation through shallow state anchors and constructed-state consistency (CS). It retains standard causal attention without dedicated copy gates or enforced predecessor routing.

Reusable local computation. Cai et al. (2017) use recursive decomposition to make verification of learned neural programs tractable. Ito et al. (2026) demonstrate compositional generalization from shallow to deeper arithmetic, Boolean, and ListOps expressions with autonomous halting. Their approach uses depth-sensitive masks, hard attention, discrete states, and gated updates that preserve inactive positions. Our architectural experiments learn a predecessor-only interface with stateless re-embedding. The full-causal experiments instead learn preservation through state objectives, retaining ordinary causal attention and continuous hidden states at inference.

Stability and recovery during computation. Deep equilibrium models define representations through input-conditioned fixed-point equations and train via implicit differentiation (Bai et al., 2019). In recurrent algorithm learning, Bansal et al. (2022) combine input recall and progressive training to avoid overthinking, and examine recovery from perturbed or replaced internal states. Anil et al. (2022) connect generalization to harder instances with path independence: convergence to the same limiting behavior from different initial states for a fixed input. Labovich (2026) analyzes fixed-point reachability, input dependence, and parameter-space geometry in looped Transformers.

Equilibrium Reasoners distinguish solution-aligned from spurious attractors and use randomized initialization and noise injection to make correct solutions easier to reach (Huang et al., 2026). Fixed-Point Reasoners combine architectural stabilization with convergence-based adaptive computation (Movahedi et al., 2026). In full-causal models, correct one-step execution from clean states can coexist with failed long-chain computation. One-step consistency can also improve while multistep recovery and long-chain accuracy deteriorate.

Consistency and neighborhood robustness. Denoising autoencoders reconstruct clean inputs from corrupted observations (Vincent et al., 2008), whereas contractive autoencoders penalize the encoder's input Jacobian to encourage locally insensitive representations (Rifai et al., 2011). For recurrent algorithm learning, Bear et al. (2024) use Lipschitz constraints to stabilize iterative computation. CS trains selected positions to preserve their input states while leaving the executing frontier free to update. Shallow answer and state-anchor losses teach execution, while CS supplies the preservation objective. Targeted denoising and multistep recovery training extend this approach and improve compositional generalization on long chains in the larger-world experiments (Appendix E).

B Architectures and Training Objectives

This appendix specifies the learned architectures and losses summarized in Section 3.

B.1 Predecessor-only stateless re-embedding

Modes 2 and 3 make each local computation stateless with respect to its own history. Let $c_i = E(r_i)$ be the original relation embedding, $c_0 = E(x)$, and s_i^t the state communicated after loop t. At every loop, the position's query and residual input are reset to c_i; keys and values come from the previous communication states. A strict singleton mask allows position $i > 0$ to read only $i - 1$, with the root reading itself. In the implementation this mask overrides the ordinary window option. Thus "one-hop routing" here is stronger than a self-plus-predecessor attention window of one. The model uses a single shared, pre-layer-normalized Transformer block with GELU, width 768, 12 heads, and no positional encoding. Its attention and feed-forward weights, entity embeddings, and relation embeddings are learned.

For a relation position, denote the resulting block output by

$$\begin{aligned} o_i^{t+1} &= B_\theta(c_i; s_{i-1}^t), \\ p_i^{t+1} &= \mathrm{softmax}\big(E_{\mathcal{E}}^\top \mathrm{LN}_f(o_i^{t+1})\big), \qquad u_i^{t+1} = E_{\mathcal{E}} p_i^{t+1}. \end{aligned} \tag{5}$$

where the columns of $E_{\mathcal{E}}$ are the learned entity embeddings and the softmax is restricted to entities. The communication rules are

$$s_i^{t+1} = \begin{cases} c_i + u_i^{t+1}, & \text{mode 2}, \\ u_i^{t+1}, & \text{mode 3}, \end{cases} \qquad s_0^t = c_0. \tag{6}$$

The final answer is decoded from the current block output. Re-embedding is a differentiable expectation, not an argmax, nearest-embedding projection, or insertion of a gold intermediate value. Mode 3 gives a computed predecessor value the same representational form as a raw entity input, up to its soft probability mixture. Mode 2 retains the predecessor's original relation embedding in that message, although the receiving position already receives its own relation separately.

Training the stateless models. Both stateless modes train with final-answer CE on atomic and two-hop queries. Each update uses 32 direct atomic queries, 16 width-two sources rendered as 32 additional atomic queries, and 64 two-hop queries. Answers are supervised at loop 1 for atomic queries and loop 2 for two-hop queries, without intermediate-state or anchor targets. Both use 200,000 AdamW (Loshchilov & Hutter, 2019) updates, 2,000 warm-up updates, linear learning-rate decay, and final-checkpoint evaluation. Inference uses the externally supplied budget $R = d$. Results and tested answer retention appear in Appendix D.1.

B.2 Standard full-causal loop and shallow answer labels

The full-causal model repeatedly applies four Transformer blocks with residual connections, GELU (Hendrycks & Gimpel, 2016), layer normalization (Ba et al., 2016), and rotary position embeddings (Su et al., 2021). It reads the final position after $R = d$ loops without test-time projection,

state resets, a hard predecessor mask, or an explicit readiness channel. Numerical architecture and optimization settings appear in Appendix C.2.

Shallow answers and state anchors. Answer-supervised training uses atomic queries and depth-two compositions. The shallow computation runs for four loops; a depth-d answer contributes cross-entropy at $t = d, d+1, d+2$. For the j-th relation after the root, the anchor target is its original relation embedding when $t < j$ and the embedding of its true prefix value v_j when $t \geq j$. The index j excludes prepended entity tokens; the root retains its original entity target. In the loss below, $\mathcal{V}$ indexes all valid, nonpadding token locations across the batch, with each i identifying both a sample and its token position. The target A_i^t is aligned with H_i^t at that location:

$$\mathcal{L}_{\text{anchor}} = \frac{1}{4} \sum_{t=1}^{4} \frac{\sum_{i \in \mathcal{V}} \|H_i^t - \text{sg}(A_i^t)\|_2^2}{\sum_{i \in \mathcal{V}} \|\text{sg}(A_i^t)\|_2^2}, \tag{7}$$

where sg stops gradients through targets drawn from the model's current input embeddings. Completed-value targets reuse the known atomic value v_1 and two-hop answer v_2; all other targets copy their input token embeddings. These soft state and timing constraints encourage computed outputs to match the entity input format used by subsequent operations. Targets enter the loss only and are never substituted into the forward trajectory.

Entity prefixes. We prepend 0–128 random entity tokens to shallow queries, sampling the prefix length uniformly. Including the root and at most two relations, this branch has at most 131 input tokens. The chain's local execution clock starts at its root, while its physical positions shift. Prefix tokens are anchored to their own detached input embeddings. This augmentation exposes shallow computation to later positions and entity-rich contexts resembling completed prefixes, without additional answer labels.

B.3 Constructed-state consistency and role masks

The standard CS branch uses depth $D = 32$ (33 token positions), no additional entity prefix, and one application of F_θ. For each synthetic chain, sample the number $k < D$ of completed positions. The detached input S contains an entity embedding at the root, independently sampled entity embeddings at positions $1, \ldots, k$, and relation embeddings at positions $k+1, \ldots, D$. These sampled entities need not equal any true prefix values. No target is supplied for the executing frontier $k + 1$; preservation targets copy the sampled input embeddings.

The normalized squared displacement in Equation (4) is

$$D(a, b) = \frac{\|a - b\|_2^2}{\max\{\|a\|_2^2 + \|b\|_2^2, \epsilon_{\text{num}}\}}, \tag{8}$$

with a positive numerical safeguard ϵ_{num}. Full-CS selects every nonfrontier position, including the root; wait-CS selects only $i > k_b + 1$. Every role ablation uses $N_{\text{CS}} = \sum_{b,i} \mathbb{I}[i \neq k_b + 1]$, without renormalizing by the number of selected roles. The output norm in the residual denominator remains differentiable, while S is detached.

In the original eight-model ablation, every arm retains root CS and the complete shallow anchor: "root+wait" and "root+hold" denote additional long-state terms. The subsequent wait-CS condition removes both root and hold CS, retains the same wait coefficient and denominator, and still uses the full shallow anchor. Root and hold outputs can affect the selected waiting outputs through attention even when they have no direct CS penalty.

CS supplies a state format and frontier role. Identity dynamics can achieve zero CS loss without executing a relation; conversely, correct atomic outputs do not ensure reusable natural trajectories.

Denoising and walkback extend this standard one-loop objective to perturbed inputs and repeated updates; their constructed depths and rollout schedules are specified separately in Appendix E.

C Training and Evaluation Protocols

C.1 Dataset and query distributions

The primary world contains 500 entities and 20 independently sampled permutation relations (world seed 42). An 8-regular directed graph (graph seed 1) defines 160 supervised relation pairs. Training includes all 10,000 atomic facts and all 500 starting entities for each supervised pair, giving 80,000 depth-two questions. Validation uses 20 disjoint pairs and 50 starts per pair; the remaining 220 pairs form the unseen-adjacency pool. Test generation uses a separate random seed and removes duplicate inputs within each depth–distribution cell.

In `all_seen`, every adjacent relation pair is a training edge; depth-two queries therefore check fit on training support. In `one_new`, exactly one adjacency comes from the test pool, at a varying position, and all others are training edges. In `random`, relation sequences are sampled uniformly and rejected if they contain a validation adjacency. Entities and relations may repeat, including in cycles. These distributions share one atomic world. Constructed CS states can contain relation contexts outside the answer-training graph. Formal appendix accuracy comparisons pool `all_seen` and `random` equally within each seed before calculating the across-seed mean and variance. The supplementary comparisons use their existing 5,000 questions per distribution (10,000 pooled per depth and seed); the main table uses the counts in Appendix C.4. Mechanistic interventions and training monitors retain the explicitly stated diagnostic cohorts.

C.2 Full-causal training

The main full-causal backbone has four Transformer blocks, hidden width 768, 12 heads, GELU, tied input/output embeddings, full causal attention, RoPE base 10,000, and no dropout. Its 28,756,225 parameters include an unused 769-parameter stopping head. AdamW uses learning rate 10^{-4}, weight decay 0.01, 2,000 warm-up updates, linear decay to zero at 200,000 updates, and gradient clipping at 1. Training uses FP32 tensors with TF32 matrix multiplication; evaluation uses strict FP32. The separate root-only mechanism diagnostic uses the configuration in Appendix F.3.

Each update contains 128 shallow answer queries: 32 atomic queries, 32 independently rendered atomic queries from 16 paired source rows, and 64 depth-two queries. The model performs four loops, with atomic answer CE at loops 1–3 and depth-two CE at loops 2–4. The anchor has weight 1, and random entity prefixes have length 0–128. Anchor and CS definitions are given in Appendix B.

CS starts at update 20,000 with weight 1, using 32 constructed sequences of depth 32 per update. Role ablations retain the common normalization and shallow anchor. The matched CE-only control retains the CE schedule, prefixes, optimizer, and endpoint but omits anchor and CS. Baseline-specific schedules appear in Appendix D.3.

Optimization seeds. Four-seed experiments use $\{1, 7, 43, 2024\}$; three-seed experiments use $\{1, 7, 43\}$, and two-seed comparisons use $\{1, 7\}$. Available computation determines each cohort's size, particularly for long chains. All runs within each reported cohort contribute to its statistics. Optimization seeds are distinct from world and graph seeds; single-run mechanistic probes are separate from multi-seed accuracy comparisons. Root-only and wait-CS use four seeds, root+wait, root+hold, and full-CS use two, and the anchor-only comparison uses three. The latter cohort and the four-seed wait-CS cohort both train for 200,000 updates.

C.3 Evaluation provenance

Formal accuracies use configured training endpoints, not maxima selected on deep tests. The 16 controlled full-causal runs and the stateless mode 3 comparison use update 200,000. World-change endpoints follow their training configurations and logs where evaluation files omit checkpoint identifiers. Mechanistic probes and training monitors are reported separately from formal evaluations. API settings and token usage are limited to the records described below.

C.4 Main comparison and statistics

Table 1 uses an equal mixture of `all_seen` and `random` for trained models at every reported depth, retaining the exclusion of validation adjacencies from `random`. Through depth 128, each

depth and optimization seed has 5,000 questions in total. At depths 256–2048, wait-CS uses 200 questions per distribution per seed (400 pooled). At depth 256, Vanilla and both RL baselines use the same counts; shallow and dense CE use 1,000 per distribution (2,000 pooled). Questions are shared across optimization seeds, which are not independent world replications.

The main-table seed counts are three for Vanilla and each RL recipe, two for each CE-only recipe, three for anchor-only, three for each published loop objective, and four for wait-CS through depth 128. Long-chain wait-CS uses the first three seeds in Appendix C.2; seed 2024 is omitted because of computational cost. All three evaluated seeds reach 100% at depths 256, 512, and 1024, giving zero seed variance. The across-seed variance at depth 2048 is not reported.

Trained models use frozen weights and no evaluation prefix. All looped rows, including RL, are read at forced $R = d$; Vanilla applies its four distinct blocks once. The long-chain evaluations use new depth-d chains, whereas answer-persistence probes continue an already completed depth-128 query. Architecture and training schedules are specified in Appendices B and D.3.

API rows pool 40 requests per depth in the same balanced mixture, with no seed-variance estimate. They receive the fact table in context, whereas trained models learn the fixed world. The comparison therefore uses different knowledge access and compute budgets. Prompts, model settings, and scoring are specified in Appendix C.5.

Mean and variance. For N seed accuracies a_s expressed as percentages, tables report $\bar{a} = N^{-1}\sum_s a_s$ and the unbiased sample variance $(N-1)^{-1}\sum_s(a_s-\bar{a})^2$ in parentheses when $N \geq 2$. Variance is measured in percentage points squared and describes optimization-seed spread, not query-sampling uncertainty. Display precision is shortened for readability; exact statistics remain in the accompanying data files. A dash denotes an unavailable or undisplayed comparison.

C.5 API EVALUATION

Information supplied to the models. Every API request includes the complete relational world as a fixed text prefix, followed by a user message describing the task and giving one query. The table contains all 10,000 relation–entity facts, plus one header line (170,105 characters), over entities `E000`–`E499` and relations `R00`–`R19`. Facts are grouped by relation and sorted by input entity. The fact table uses the same relation map as the trained-model world. A line such as `E190 R05 -> E304` states one atomic transition. Every relation is a permutation, so the table supplies exactly one output for each entity–relation input.

The fact prefix and instruction template are the same across all reported high-reasoning runs. Only the query line changes. There are no few-shot examples, task tools, or fallback models. No sampling parameters are explicitly supplied. The API models must retrieve and compose facts from context; the loop models instead acquire the fixed world through training. The labels `all_seen` and `random` refer to the loop training graph's relation adjacencies, not to known or unknown API pretraining examples. The pooled API score cannot isolate the effect of held-out adjacencies.

User-message template. The evaluation protocol provides the following text, with only the final query substituted. The identifier `E999` in the format instruction is deliberately outside the entity set, so copying it does not yield an accidentally correct answer.

> The table above is complete: every relation is a function defined on every entity, and for every entity and every relation exactly one line exists.
> Below is a query: a start entity followed by a sequence of relations. Apply the relations one after another, starting from the start entity: the first relation to the start entity, the second relation to the result, and so on.
> Answer with a single line of the form "`ANSWER: E999`" containing the final entity. Do not write anything else.
> `QUERY: <start entity> <relation 1> <relation 2> ... <relation d>`

The instruction requests only a visible final answer; it does not disable hidden reasoning. For example, `QUERY: E190 R05 R08 R09 R02` has answer `ANSWER: E054`. The intermediate path is not included in the query. Longer chains add relation names while leaving the fact prefix unchanged.

API settings and budgets. GPT-6 Sol and Muse Spark 1.3 use OpenRouter with high reasoning effort. Table 4 reports the available interface settings, output limits, and depth-four prompt lengths. Output ceilings cover hidden reasoning and the visible answer; they do not equalize inference budgets.

Model identifier	Interface	Requested output cap	Prompt tokens at $d = 4$
`gpt-6-sol`	OpenRouter	—	—
`claude-opus-5-5`	Messages (streaming)	128,000	110,214
`gemini-3.8-flash`	GenerateContent	65,536	130,154
`deepseek-flash`	Chat Completions	393,216	80,168
`muse-spark-1.3`	OpenRouter	—	—

Table 4: Recorded API settings. All five runs request high reasoning effort. Dashes denote unavailable token limits or counts; displayed caps are not matched compute budgets.

Claude Opus 5.5 receives the table in the system text, with ephemeral prefix caching and `output_config.effort="high"`. Gemini 3.8 Flash uses `system_instruction` and `thinking_level=HIGH`, with automatic function calling disabled. DeepSeek-V4.1-Flash uses the system message, `reasoning_effort="high"`, and enabled thinking. Gemini 3.8 Flash and DeepSeek-V4.1-Flash use provider prefix caching. The first request populates the cache, followed by concurrency four for Claude Opus 5.5/Gemini 3.8 Flash and 16 for DeepSeek-V4.1-Flash. Caching supplies no intermediate answers.

Sampling and scoring. The API evaluation contains 40 first-pass requests for each of five models at each of twelve depths $2, 3, 4, 8, \ldots, 2048$: 480 requests per model. Every reported API cell is complete. Each API cell pools 20 `all_seen` and 20 `random` queries, giving a balanced 1:1 mixture. Figure 2 and Table 1 convert the supplied integer counts to percentages; no uncertainty bands or API variance estimates are reported. The evaluator takes the last case-insensitive `ANSWER: E<number>` match, allowing formatting characters after the colon. If absent, it takes the last entity identifier in visible text and flags the missing answer line. Accuracy therefore measures the parsed final entity. Refusals and output-limit stops remain in the denominator and count as incorrect.

Only Claude Opus 5.5 is affected by safety refusals in this cohort, and every non-refused Claude Opus 5.5 answer is correct. Refusals therefore explain its accuracy decline; selected retries of refused queries are excluded. The trained-model evaluation protocol appears in Appendix C.4.

Earlier low-reasoning, trace-output, four-digit-entity, and shuffled-table controls are excluded from the displayed cohort.

D Supporting Comparisons

The comparisons below complement the main tables. The stateless architecture and the full-causal loop use different structural priors but the same 200,000-update endpoint; each recipe is specified separately.

D.1 Architectural re-embedding from shallow answers

The stateless models use the learned predecessor interface in Appendix B.1: one shared block, width 768, 12 heads, NoPE, and approximately 7.49M parameters. Modes 2 and 3 both train for 200,000 updates on atomic and two-hop answers. For each mode we test both $k = 8$ and $k = 1$, where k is the number of trained relation successors per relation. Each condition has three optimization seeds and evaluates 5,000 queries at every combination of eight depths $\{2, 3, 4, 8, 16, 32, 64, 128\}$ and each of `all_seen` and `random`. Pooling these equally gives 10,000 questions per depth and 80,000 per seed. All twelve runs have zero errors on this matrix. Appendix D.2 compares the architectural and full-causal approaches.

For the six $k = 8$ stateless runs, the first 500 questions per cell are also evaluated at budgets $R \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$. Each run has zero errors over the 42,000 query–budget evaluations with $R \geq d$. This tests retention of completed answers on repeated questions, with the loop budget supplied externally. The stateless architecture builds preservation and predecessor routing into its

update; full-causal CS instead learns preservation through shallow anchors and synthetic states. Their different architectures and training recipes demonstrate two successful design routes rather than a matched single-factor comparison.

D.2 Architectural and consistency variants

Stateless re-embedding, full-CS, and wait-CS support repeated composition from atomic and two-hop answer labels. Stateless mode 3 supplies a fixed own-token input and predecessor-only re-embedded messages; its balanced-mixture evaluation is reported in Appendix D.1. Full-CS and wait-CS use full-causal continuous inference, differing in which synthetic-state roles receive preservation targets; their depth-128 comparison appears in Appendix F.7. All three recipes train for 200,000 updates; stateless results use three seeds, full-CS two, and wait-CS four. These are comparisons of complete training recipes, not equal-compute ablations.

D.3 Baseline training and evaluation

The trained baselines use the same relational world as the proposed model and 200,000 updates. Table 1 reports final endpoints with the sampling and seed counts specified in Appendix C.4. Looped models use forced $R = d$ at evaluation; Vanilla-4 applies its four distinct blocks once.

Shallow-answer controls. Vanilla-4 has width 768 and approximately 28.8M parameters and learns atomic and two-hop answers with CE. The matched answer-only loop retains the four shared blocks, RoPE, random entity prefixes up to 128, optimizer, and four-loop shallow answer schedule, with CE at $t = d, d + 1, d + 2$, while removing anchor and CS. The anchor-only condition restores the shallow state targets but omits CS. It uses three seeds, while the matched CE loop uses two and wait-CS uses four. All use the same 200k-update endpoint; the anchor-only and wait-CS results are separate cohorts.

Dense answer supervision. Dense CE retains RoPE and entity prefixes but uses answer loss alone. It adds 10,000 distinct chains at every depth 3–32, for 300,000 additional chains. Each update contains 128 shallow and 30 deep rows; CE is applied at loops $d, d + 1, d + 2$. This costs approximately 1,097 row-loops per update, versus 512 for the shallow CE reference, and is not an equal-compute comparison. The longest training input has 131 tokens, compared with 129 for a depth-128 query. Two seeds are evaluated at 200k updates.

Reward supervision. Both main-table RL conditions use four shared blocks, NoPE, three seeds, and no anchors, CS, re-embedding, or entity prefixes. They add 16 reward queries at each depth 3–8 from pools of 10,000 distinct chains per depth. Joint CE/RL retains shallow CE and begins deeper correctness rewards at update 10k. SFT→RL uses 20k updates of shallow SFT, then reward-only training with a gated computation cost of 0.02. Each query supplies eight sampled stopping-time/answer pairs for one on-policy gradient update with group-normalized correctness rewards and no reference-policy KL term. The recipes differ in schedule and cost as well as whether CE continues; they are not a single-factor loss ablation relative to CS.

Published loop objectives. Five rows of Table 1 port published recurrent-depth objectives to the same world and the same shallow training rows: atomic, width-2, and depth-2 answers only. These objectives are recall with progressive loss (Bansal et al., 2022), the Neural Data Router's copy gates and geometric attention (Csordás et al., 2022b), input injection at every loop (Fan et al., 2025), dynamic recurrence (Kohli et al., 2026), and the input mixing between loops of Fixed-Point Reasoners (Movahedi et al., 2026). Each uses 512 queries per update for 200,000 updates, learning rate 2×10^{-4} with 500 warm-up updates and linear decay, and seeds 1, 7, and 123. The final checkpoint is evaluated at forced $R = d$ with the answer read at the last position.

Table 5 distinguishes reaching a correct answer from retaining it after further recurrence. Its marginal counts do not estimate querywise retention conditioned on initial correctness. The main comparison uses the common forced budget $R = d$ for every looped model.

Reward recipe	$R = 16$ mean (variance)	$R = 32$ mean (variance)
Joint CE + RL through $d = 8$	87.73 (100.25)	0.20 (0.04)
SFT $\rightarrow$ RL through $d = 8$	85.07 (64.69)	0.00 (0.00)

Table 5: **Answer persistence under further recurrence.** Accuracy (%), mean (variance) across three seeds on the same 500 depth-16 `one_new` queries per seed, read at two budgets along one continuous rollout. The stop head is bypassed. These subsets are separate from the formal 5,000-query cells.

E State Preservation in More Demanding Worlds

Wait-CS and full-CS share the objective of preserving states needed by later operations; full-CS additionally constrains root and completed positions. The world-change experiments use full-CS with the same shallow-answer training protocol within each world. Table 6 compares the original recipe with denoising in R40 and walkback in E1000. Both refinements restore depth-128 accuracy above 99.99% on the balanced `all_seen`/`random` mixture without adding answer-labeled data. These are retraining experiments with world-specific preservation settings.

Entities / relations	n	$d = 64$	$d = 128$
500 / 20, original	3	100.00 (0)	100.00 (0)
500 / 20, world seed 43	3	99.990 (0.0003)	99.993 (0.00013)
500 / 40	3	1.327 (0.932)	0.20 (0.0021)
500 / 40 + denoising	3	100.00 (0)	99.993 (0.00013)
1,000 / 20	3	54.38 (2551.25)	0.093 (0.00043)
1,000 / 20 + walkback	3	100.00 (0)	100.00 (0)

Table 6: **World changes and state-preservation refinements.** Accuracy (%) on an equal `all_seen`/`random` mixture: mean (sample variance) across three seeds, with 5,000 questions per distribution, depth, and seed (10,000 pooled). The original full-CS recipe uses constructed depth 32, 32 rows, prefixes through 128, and 200k updates. R40 denoising and E1000 walkback retain the same answer-training data and update budget within their respective world.

R40: denoising waiting and completed states. Gaussian perturbations are applied to constructed waiting and completed positions, with a row-level relative scale up to 0.1 and the clean states as targets. The constructed depth remains 32, with the original shallow anchors and prefixes. This recipe makes five errors across the three balanced evaluation matrices (240,000 questions) and reaches mean accuracy 99.993% at depth 128.

E1000: longer constructed contexts and walkback. The constructed depth increases to 128. Walkback perturbs waiting states toward nearby relations with scale up to 0.2, unrolls up to 16 loops, and differentiates through the last four. During this training loss, an external schedule replaces frontier positions with independently sampled entity embeddings; inference uses natural continuous states without replacement. The resulting models make zero errors across all three balanced 80,000-question evaluation matrices.

Both refinements attain 100% depth-128 accuracy at loops 128 and 256 on the same fixed balanced subsets of 1,000 questions per seed (500 per distribution). This tests retention of completed answers. The successful settings differ across worlds; these results do not evaluate one unchanged wait-CS recipe or shared recovery hyperparameters across all worlds.

F Mechanistic Evidence

F.1 Output-state geometry across composition depths

Example test chain. The following depth-128 input repeats the relation cycle (r_9, r_6, r_3) from root entity e_{169}. Its gold intermediate entities cycle through $e_{169} \rightarrow e_{417} \rightarrow e_{472} \rightarrow e_{169}$:

Position p	0	1	2	3	4	5	6	$\cdots$	127	128
Input token	`<e_169>`	`<r_9>`	`<r_6>`	`<r_3>`	`<r_9>`	`<r_6>`	`<r_3>`	$\cdots$	`<r_9>`	`<r_6>`
Gold entity	e_{169}	e_{417}	e_{472}	e_{169}	e_{417}	e_{472}	e_{169}	$\cdots$	e_{417}	e_{472}

Figure 6 uses 32 disjoint three-cycle chains as a controlled visualization cohort, separate from the main mixed-distribution benchmark. Each cycle's entity-embedding plane is mapped to a common equilateral triangle. At $R = d$, we plot positions $d-2, d-1, d$ (96 states per card; the root is included at $d = 2$). Color identifies the expected entity; filled markers select it as the nearest of 520 token embeddings, whereas the printed answer accuracies use the LM head. The third coordinate summarizes the unsigned residual norm outside the plane, drawn approximately $10\times$ smaller than in-plane distances. Grey dots mark projections; projections outside a card are placed at its edge (at most 11/96, all at 60k). Figures 8–10 show the same states from three orthographic views, with the same colors and marker conventions.

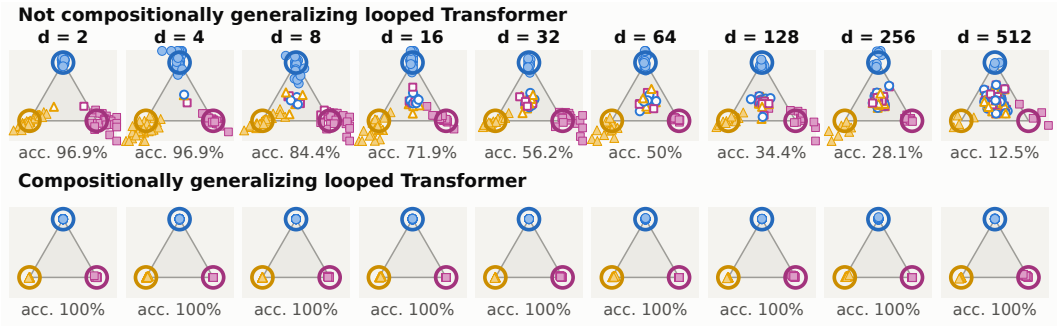


Figure 8: **Front view of output states.** Face-on projections of Figure 6. At 60k (top), more states project toward the triangle center as depth grows; at 200k (bottom), projections remain near their target embeddings. Off-plane distances are not visible in this view.

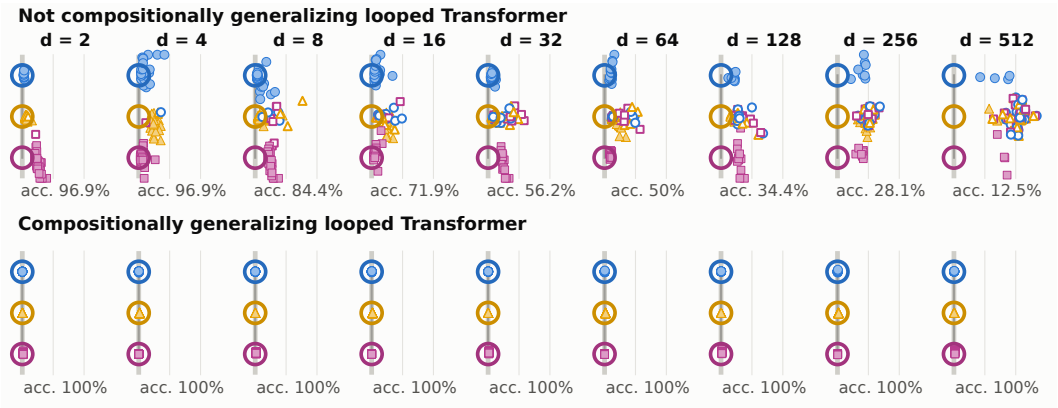


Figure 9: **Side view of output states.** Horizontal displacement shows compressed distance from the entity-embedding plane; vertical position separates the three cycle entities. At 60k (top), states increasingly depart from the plane with depth; at 200k (bottom), they remain close to their target embeddings.

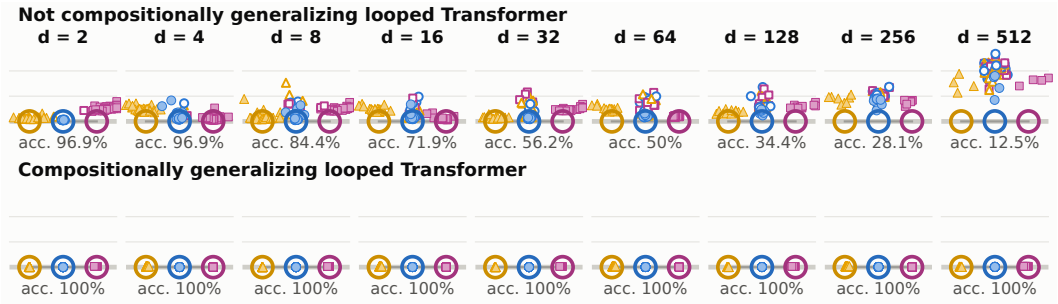


Figure 10: **Top view of output states.** Horizontal position shows one in-plane coordinate; height shows compressed distance from the plane. The 60k states (top) move farther off-plane at greater depths, whereas the 200k states (bottom) remain close to the entity embeddings.

F.2 LOCAL COMPOSITION INTERVENTIONS

Figure 5 uses frozen wait-CS checkpoints and the first 32 prespecified chains from each of `all_seen`, `one_new`, and `random`: 96 chains at each depth in $\{2, 4, 8, 16, 32, 64, 128\}$. Chains and replacement draws are deterministic. Inference uses FP32 with TF32 disabled, no prefix, and no added noise. These probes are separate from the formal balanced-mixture evaluation.

Decoded trajectories. For depth 128, we record the full-vocabulary argmax at every position and loop, including loop zero, without state replacement or teacher forcing. The displayed example is the first `all_seen` chain (sample ID 105000). The template labels positions $j \leq R$ by their correct intermediate entities and positions $j > R$ by their original relations. The dashed diagonal is this prescribed frontier. Early execution counts as a template mismatch; semantic agreement does not require stationary hidden vectors.

Attention support. The self-plus-predecessor and self-only restrictions apply to every attention block and loop, starting from the original embeddings and running $R = d$. All other model operations remain unchanged. An independently implemented unrestricted forward is bitwise equal to the original at every probed checkpoint. Four blocks per loop allow information to travel up to four positions under the local restriction, so it does not impose a one-hop schedule. Success under this restriction shows sufficiency of the retained connections on the probe cohort, without establishing exclusive use by the original model.

Operand replacement. At frontiers $p \in \{2, 4, 8, 16, 32, 64, 128\}$ in the same 96 depth-128 chains, a canonical state contains correct entity embeddings before p and original relations from p onward. One unmodified loop is then applied. Replacing the predecessor entity tests target $\phi_{r_p}(x')$; replacing the frontier relation tests $\phi_{r'}(v_{p-1})$. Alternatives must change the true target. A third intervention changes entities at positions $0, \ldots, p-2$, preserves the local operand pair, and averages three fixed distant-context draws per chain. Accuracy uses the true relation table. Changed prefixes need not form a globally consistent chain: these are local internal-state interventions. Panel C averages over seven frontier positions; its first three bars measure one-loop operation accuracy, while the last measures end-task accuracy from an uninterrupted rollout.

F.3 DECODING AND NEXT-STEP STATE REUSE

The additional diagnostics below use the first 100 depth-128 chains from each of `all_seen`, `one_new`, and `random`, for 300 chains. We retain this recorded three-category cohort separately from the formal balanced-mixture metric and the 96-chain Figure 5 cohort. The wait-CS and CE-only models use RoPE base 10,000 and model seed 7. The root-only diagnostic uses base 10,000 and seed 42. Checkpoint labels indicate the number of training updates.

For the canonical write–read test, let F be one loop and C_t the canonical state with correct entity embeddings through position t and original relations thereafter. For $t \in \{4, 12, \ldots, 124\}$, the writer produces $f = F(C_t)_{t+1}$. We decode f with the model's final LayerNorm and full-vocabulary LM head, then insert it at position $t+1$ of C_{t+1} and apply one loop to test the correct entity at $t+2$. The gold control inserts the correct entity embedding instead. There are $300 \times 16 = 4{,}800$ events per checkpoint; reuse is conditional on correct decoding of f.

Table 3 reports the 200k comparison: all 4,800 outputs decode correctly in each model, while CE-only has 2,085 next-step reuse failures and wait-CS has none. Both gold controls succeed on all events.

F.4 PAIRED WRITER AND READER INTERVENTIONS

The paired writer–reader probe scans $t = 1, \ldots, 126$ and retains events with historical survival $\bigcap_{i=1}^{t} \{\mathrm{LM}(H_i^i) = v_i\}$: each position decoded correctly at its own execution loop. This does not require every earlier position to remain correct in the current state. At $p = t+1$, the natural writer gives $f_N = F(H^t)_p$ and the canonical writer gives $f_C = F(C_t)_p$. The natural reader is H^{t+1} and the canonical reader is C_{t+1}; replacing position p with either writer output and applying one loop yields four tests at $p+1$. In NN, NC, CN, CC, the first letter identifies the writer and the second the reader. Both writer outputs must decode correctly to enter the table.

Checkpoint	Eligible events	Writer rescue $NN \to CN$	Reader rescue $NN \to NC$	Gold/N failures	Gold/N+relation failures
Wait-CS 90k	33,117	34/67	6/67	19	17
Wait-CS 120k	34,885	19/45	0/45	3	2
Root-only 200k	4,634	43/300	251/300	257	0

Table 7: Paired rescues among the same natural NN failures. Gold-control failure counts use all eligible events.

Gold/N replaces the predecessor with its correct embedding in the natural reader. Gold/N+relation additionally restores the original relation embedding at $p{+}1$. In root-only, all 257 gold/N failures are rescued by this additional edit, locating an operand defect given a correct predecessor. The stronger writer rescue in wait-CS and reader rescue in root-only are within-checkpoint diagnostic effects. The different model seeds preclude interpreting their contrast as a controlled wait-CS ablation. Event counts pool positions within chains and are not independent chain-level accuracy estimates.

F.5 Readiness and reuse across positions

After natural loop $t = 32$, readiness interventions change one predecessor and apply one further loop. In the early-insertion test, position 33 receives its gold entity embedding and position 34 is read. In the revert test, position 32 is reset to its original relation embedding and position 33 is read. The original probe decodes by nearest embedding among the 500 entity and 20 relation tokens, so its readout differs from the LM-head tests above.

Wait-CS 120k intervention	Read position	Unmodified correct	Edited correct
Insert ready predecessor	34	0/300	298/300
Revert predecessor to relation	33	288/300	0/300

Table 8: Readiness responses on the 300-chain cohort after loop 32. Correct means the target entity for the read position.

These external state manipulations demonstrate a response to predecessor readiness on this cohort; they do not rule out implicit timing information under arbitrary inputs.

Cross-position update directions. We sample 300 entity–relation pairs (probe seed 0), one per chain context, and place each fixed pair at predecessor/frontier positions $p-1, p$ for $p \in \{2, 8, 16, 32, 64, 96, 128\}$. Earlier positions contain gold entity embeddings and later positions retain original relations. For each position pair, cosine similarity between the updates $F(C)_p - C_p$ is averaged over the 300 operand pairs. For wait-CS at 200k, the mean over the 42 off-diagonal position pairs is 0.999516 and the minimum of these position-pair means is 0.998665; cyclically mismatching operand pairs gives mean 0.152952. This measures the directional similarity of sampled local updates across positions, without asserting exact functional equality.

F.6 Training dynamics and state drift

Figure 7 follows one wait-CS run with full shallow anchors, random training prefixes of length 0–128, and four shared Transformer blocks. It includes all 23 recorded checkpoints: 1k, 2k, 5k, then every 10k through 200k. The diagnostic adds no training or selection by test performance.

Panel A uses a fixed 8,000-example depth-two training subset, 1,000 validation examples, and 100 questions from each evaluation distribution at every greater depth (300 queries per deep accuracy point). "Sustained" means all later recorded points reach at least 99%, without interpolating between checkpoints.

Panels B/C use FP32 inference with TF32 disabled, no evaluation prefix, and no added noise. At each depth in $\{2, 4, 8, 16, 32, 64, 128\}$, the first 32 fixed questions per distribution give 96 chains. Natural trajectories run uninterrupted for 128 loops without gold-state insertion. A canonical control instead supplies correct entity embeddings at completed positions and original relations elsewhere, then executes one loop. This restores several roles simultaneously, rather than isolating waiting states. Frontier accuracy uses full-vocabulary argmax. Panel C displays positions $\{2, 4, 8, 16, 32, 64, 128\}$ separately; positions within a chain are correlated.

Checkpoint	Natural $d = 128$	Fresh centered	Hold centered	Reuse failures
120k	$254/300$	1.203	0.1442	$9/4800$
130k	$299/300$	0.846	0.0156	$0/4800$
200k	$300/300$	0.128	0.0014	$0/4800$

Table 9: Concurrent changes in the wait-CS run. Centered errors use the separate probe in Appendix F.6.2. Reuse failures come from a canonical writer–reader test: all 4,800 fresh states decode correctly at each listed checkpoint, but some fail when consumed by the next step.

Panel B averages cumulative relative displacement over the positions eligible for each role:

$$\frac{\|h_i^R - h_i^{\mathrm{ref}}\|_2}{\max\{\|h_i^{\mathrm{ref}}\|_2, \epsilon_{\mathrm{num}}\}}.$$

Root and wait reference their original embeddings; hold references its own output at expected execution loop $R = i$. Roles follow the prescribed schedule, with empty roles omitted. Membership therefore changes with loop count, and there is no wait measurement at $R = 128$ for a depth-128 chain. This statistic is neither squared error nor the CS loss.

An additional probe fixes wait positions 65–128 and measures $R \leq 64$, avoiding changing role membership. The fitted displacement growth slope is 0.013927 per loop at 120k, 0.004346 at 130k, and 0.00006288 at 200k. The $3.2\times$ decrease from 120k to 130k accompanies the depth-128 accuracy transition. These slopes are not monotonically decreasing throughout training and summarize only the measured positions and loop range. This single trajectory and its checkpoint spacing limit causal and transition-time conclusions.

F.6.1 Per-loop interface repair

The state-restoration curve in Figure 4 uses 300 fixed depth-128 chains, the first 100 from each of the three evaluation distributions, at all 23 recorded checkpoints. Starting from the original input embeddings, we execute one shared loop and, after each loop $t = 1, \ldots, 127$, replace the just-executed position t by its nearest entity embedding in raw Euclidean distance (among all 500 entities). We also reset every still-waiting position $i \geq t + 1$ to its own original relation embedding. The next loop then reads these repaired states. The final answer is the full-vocabulary LM-head argmax after loop 128, before any further edit. The entity choice uses the model state, not the gold intermediate entity; the relation reset uses the given input. The projection onto the model's entity embeddings and its prescribed timing are supplied externally.

At 5k, this combined repair raises accuracy from $0/300$ to $283/300$; at 120k, from $254/300$ to $298/300$; and at 130k, from $299/300$ to $300/300$. These results establish the effectiveness of this interface repair, without implying that natural execution performs the same projection.

F.6.2 Centering and concurrent representation changes

Raw displacement can include a direction invisible to LayerNorm. Write $Pz = z - \mathrm{mean}(z)\mathbf{1}$ for centering across channels. In this pre-LN model, a tokenwise constant shift passes through the residual stream while leaving normalized inputs and final logits unchanged. We therefore also examine centered errors in a separate 300-chain probe at $t \in \{16, 64, 112\}$. The fresh error is $h_t^t - E(v_t)$. For the conditional hold residual, we replace only position t of the natural state H^t by its correct entity embedding, execute one loop, and subtract that embedding from the resulting state at t. Each norm is divided by $\|E(v_t)\|_2$ and averaged over chains and the three positions. These are distinct measurements from Panel B's cumulative displacement over changing role memberships.

The corresponding raw conditional hold residual falls from 0.1458 to 0.0172 between 120k and 130k; the centered reduction in Table 9 shows that this change persists after removing the mean direction. The reuse test follows Appendix F.3; all gold-embedding predecessor controls succeed at the listed checkpoints. These improvements occur within the same 10k checkpoint interval; the measurements do not determine their causal ordering.

F.6.3 Continued execution and internal hold updates

A smaller probe uses 60 depth-128 chains (20 per distribution), runs the natural system for 128 loops, and then continues all positions for 256 additional loops, reaching $R = 384$. At the final transition,

we average $\|P(h_i^{384} - h_i^{383})\|_2/s_E$ and $\sqrt{D}|\operatorname{mean}(h_i^{384} - h_i^{383})|/s_E$ over all 129 positions and all chains. Here $D = 768$ and s_E is the mean nearest-neighbor distance between centered entity embeddings at that checkpoint. At 140k these centered and mean step sizes are 4.59×10^{-6} and 5.75×10^{-3}; at 200k they are 3.65×10^{-6} and 1.02×10^{-4}. Terminal answer accuracy is $60/60$ at 140k, 170k, and 200k, compared with $52/60$ at 120k and $59/60$ at 130k. Thus the late centered states change very little over the measured final step even while a mean component remains; this finite continuation does not certify an infinite-time limit.

Small net hold updates can coexist with large internal updates. In a canonical state at $t = 64$ on 300 chains, the mean MLP-update norms in blocks 0 and 3, divided by the held entity's embedding norm, are respectively 12.98 and 12.65 at 120k, 13.16 and 12.64 at 130k, and 10.36 and 10.93 at 200k. The corresponding complete-loop hold residuals are 0.1152, 0.0157, and 0.00139. Since the attention and MLP increments sum to the complete-loop update, preservation here involves cancellation among substantial sublayer updates.

F.7 Role-specific preservation and answer retention

Table 10 pairs depth-128 accuracy on an equal `all_seen`/`random` mixture with a canonical waiting-state probe. Every role-constrained model uses the same full shallow anchor, normalization, and 200k endpoint. The existing 5,000 questions from each distribution are pooled within each seed, giving 10,000 questions per accuracy cell. Canonical probes use 100 constructed states per depth: an entity prefix, an executing frontier relation, and waiting relations. All conditions have 100% frontier accuracy on these clean inputs, while their preservation and natural long-chain accuracies differ.

CS / objective	n	Mixed accuracy (%)	Wait displacement
Root	4	50.085 (3322.01)	0.15304 (0.00278)
Root+hold	2	0.26 (0.0968)	0.20715 (0.00019)
Root+wait	2	100.00 (0)	0.00142 (6.10×10^{-8})
Full	2	100.00 (0)	0.00154 (9.66×10^{-9})
Wait-CS	4	99.9975 (2.5×10^{-5})	0.00136 (7.76×10^{-9})
CE only	2	0.155 (5×10^{-5})	24.55941 (21.32570)

Table 10: **Depth-128 role comparisons.** Mean (sample variance) over n seeds. Waiting displacement is the relative ℓ_2 distance from each constructed target, distinct from the symmetric squared residual used for CS. CE only removes both anchor and CS.

Natural-rollout probes use 100 existing questions from each of `all_seen` and `random` per depth, giving 200 equally mixed questions. The ten high-accuracy constrained models retain every correct depth-128 answer through loop 256 on these subsets. This concerns continued execution of the same chain, not a new depth-256 query. The aggregate accuracy and displacement measurements support role-specific preservation; a mean embedding distance alone is not a criterion for correct composition.

F.8 Multistep recovery and state restoration

F.8.1 Controlled recovery probe

In a constructed chain of depth D, the first k_0 completed positions contain independently sampled entity embeddings and the remaining positions contain relation embeddings. Let S be this clean state and X^0 its optionally perturbed version. Waiting positions receive Gaussian noise with per-coordinate standard deviation $\sigma\|E(r)\|/\sqrt{p}$, where p is hidden width. After each model update, P_i replaces the prescribed frontier at position i with a pre-sampled entity embedding:

$$X^{j+1} = P_{k_0+j+1}\big(F_\theta(X^j)\big), \qquad 0 \le j < D - k_0. \tag{9}$$

Here j counts completed updates. The replacements supply a state format and execution schedule, without correct relation-chain answers. On the nonempty waiting mask $\mathcal{W}_j = \{i \in \{1, \dots, D\} : i > k_0 + j + 1\}$, which excludes the next prescribed frontier, we measure

$$R_j = \frac{1}{|\mathcal{W}_j|} \sum_{i \in \mathcal{W}_j} \frac{\|X_i^j - S_i\|_2^2}{\|X_i^j\|_2^2 + \|S_i\|_2^2}. \tag{10}$$

World	Update	Clean, 1 step	Clean, 16 steps	$\sigma = .1$, 1 step	$\sigma = .1$, 16 steps
Baseline	200k	8.71×10^{-7}	1.41×10^{-6}	9.40×10^{-5}	8.40×10^{-6}
R40	150k	1.57×10^{-5}	9.64×10^{-5}	1.34×10^{-4}	1.03×10^{-4}
R40	200k	1.33×10^{-6}	2.66×10^{-2}	5.22×10^{-4}	2.26×10^{-1}
E1000	200k	1.64×10^{-6}	1.44×10^{-6}	7.46×10^{-5}	8.07×10^{-6}

Table 11: **Controlled waiting-state recovery.** Seed 1, constructed depth 32, initial prefix bin $k_0 =$ 0–7. Values are normalized squared residuals under the prescribed frontier schedule.

World	Update	n	Natural	Restored
R40	150k	3	66.67 (3201.44)	100.00 (0.00)
R40	200k	3	0.44 (0.04)	100.00 (0.00)
E1000	200k	3	0.11 (0.04)	100.00 (0.00)

Table 12: **Natural execution and state restoration.** Percent accuracy, mean (variance across three seeds), on 300 shared `all_seen` depth-128 questions per seed at $R = d$. These diagnostic subsets are separate from the formal 5,000-question-per-cell evaluations.

Empty-mask means are undefined and excluded; the measured positions change with the horizon. This residual includes both baseline drift and response to noise.

The baseline and E1000 checkpoints reduce the noisy residual over 16 controlled steps (Table 11). In R40, training from 150k to 200k reduces the clean one-step residual while increasing the 16-step residual; natural depth-128 accuracy on the same diagnostic cohort falls from $299/300$ to $2/300$. One-step consistency therefore does not ensure sustained composition. Recovery also depends on context: E1000's clean 16-step residual at constructed depth 128 is 0.0527 for seed 1, compared with 1.44×10^{-6} in the depth-32 probe. These probes motivate multistep, context-dependent preservation objectives.

F.8.2 State restoration

State restoration projects completed positions to their nearest entity embeddings after each loop and restores the root and waiting relations to their original embeddings. An external schedule specifies completed positions; the intervention uses the model states and embedding table without gold intermediate answers. The final answer is scored before the last projection.

All six enlarged-world models at 200k recover to $300/300$ under state restoration (Table 12), showing that the learned parameters can compose under a repaired interface. Because state restoration combines root stabilization, waiting-state restoration, and completed-state projection, it does not isolate any one role. The denoising and walkback results in Appendix E improve natural execution through training with the same answer-training data.