# CONTINUOUS STATEFUL INTELLIGENCE: BEYOND THE TURN-BASED AGENT

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

An agent's commitments outlive individual requests. Continuous Stateful Intelligence (CSI) makes the agent's lifetime the unit of execution: observations revise persistent state, reasoning advances through bounded model calls, and communication or action leaves the agent available to continue its work. We specify this architecture and evaluate how protected state, explicit revision, and retrieval preserve commitments across interaction boundaries. In 200 paired robot scenarios, persistent state achieves 93.0% physical success, compared with 49.5% after reset and 47.5% with a recent window, matching full replay. A four-key extension approaches replay success with approximately one forty-sixth of its visible memory. Language experiments identify a retrieval failure despite correct storage; a matched comparison favors exact lookup over the learned reader. During ongoing robot activity, scheduled corrections reliably update state, but we detect no improvement in physical completion.

## 1 INTRODUCTION

A person changes an instruction, a sensor reports an obstacle, or a tool completes a job while an agent is still working. Observations, commitments, and unfinished work have different lifetimes: an agent must revise what has changed while retaining what remains useful.

We introduce *Continuous Stateful Intelligence* (CSI), an architecture organized around a persistent execution loop: receive events, integrate evidence, advance reasoning, communicate or act, and continue. Producing a response preserves commitments and unfinished work. New observations can revise stored state and pending computation.

Durable state preserves commitments; lifecycle control updates or invalidates them; readout connects current state to behavior. Scheduling and dispatch determine whether a revision takes effect before the next action. We use *long-term memory* operationally to mean retention and revision beyond active-context or request boundaries, with measurable consequences for later behavior.

We specify an agent-level execution model with persistent state, temporal metadata, bounded reasoning, and action dispatch. We implement protected records with explicit creation, correction, and invalidation, then test their retrieval through learned and exact interfaces. Controlled language experiments isolate write interference, revision, and readout; paired robot experiments measure physical fulfillment and correction during execution; a capacity study measures accuracy under bounded storage (Table 1).

An independent fixed-sample robot confirmation yields 186/200 successes with persistence, 99/200 after reset, 95/200 with a recent window, and 186/200 with replay. A four-key follow-up tests joint use of revised records. Exact lookup outperforms learned readout on the matched language panel. During physical rollout, the runtime updates the stored preference on schedule, but we detect no increase in corrected-task completion.



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

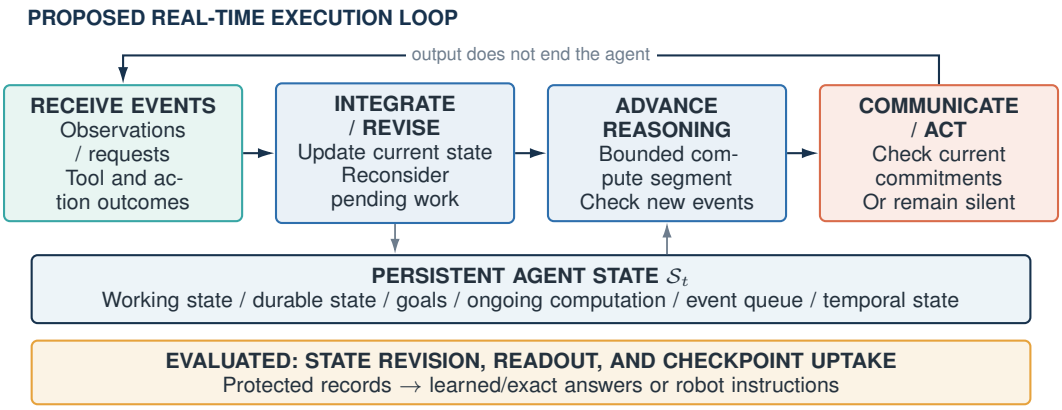


Figure 1: **CSI makes the agent's lifetime the unit of execution.** The proposed loop integrates arriving events, advances work, and dispatches effects while preserving agent state. The highlighted component is implemented and evaluated in separate language and robot systems. Checkpoint correction uptake is also evaluated. General asynchronous interruption and deadline-aware dispatch remain proposed components.

## 2 CSI: An Agent That Continues

### 2.1 The Persistent Execution Loop

Continuity in CSI is defined at the agent level: causal state persists and evolves across bounded model invocations and interaction boundaries. An output does not terminate the agent; no neural forward pass must run indefinitely. Retained, revised state must influence later behavior. Responsiveness additionally requires incorporating an event during ongoing execution; Section 6.7 tests correction uptake and physical completion.

The proposed runtime maintains commitments, unfinished work, and pending external effects in the logical state

$$\mathcal{S}_t = (W_t, P_t, G_t, C_t, Q_t, \Theta_t), \tag{1}$$

where $W_t$ is working state, $P_t$ durable state, $G_t$ active goals, $C_t$ ongoing computation, $Q_t$ pending events, and $\Theta_t$ temporal metadata. An event $e_t$ can be an observation, correction, tool result, action outcome, or timer. With computation allowance $\delta_t$, a scheduling step advances the agent:

$$(\mathcal{S}_{t+1}, u_t) = \mathcal{F}(\mathcal{S}_t, e_t, \delta_t), \qquad u_t \in \{\text{silent}, \text{communicate}, \text{act}\}. \tag{2}$$

A silent step performs internal work or waits without external output. Completed requests release temporary resources while preserving state needed by continuing goals.

### 2.2 Responsiveness, Interruption, and Action

The proposed scheduler advances reasoning in bounded segments with explicit points for integrating evidence. A correction can revise a goal, suspend an obsolete continuation, or invalidate a pending action. The design associates pending actions with state versions and dependencies to be checked before dispatch. Already executed physical effects constrain subsequent plans: correcting a destination, for example, cannot undo movement that has already occurred.

Temporal metadata distinguishes event occurrence, arrival, incorporation, and dispatch: late observations may describe earlier states, while newer corrections supersede pending plans. $\Theta_t$ records these relationships, deadlines, and validity intervals.

### 2.3 Long-Term Memory is Required for Continuity

Working state $W_t$ remains adaptive while durable records are protected from incidental overwrite. Explicit creation, correction, invalidation, temporary use, rejection, and no-change operations allow request-local computation to clear without erasing commitments.

Table 1: Architectural components and their experimental tests.

| Experiment | Observed finding | Architectural role |
|---|---|---|
| Write interference | Later writes damage retained associations | Protect durable state |
| Lifecycle counterfactuals | State changes need distinct legal operations | Make revision explicit |
| Reader-only intervention | Correct records can yield wrong answers | Connect state to behavior |
| Matched readers | Exact lookup exceeds learned readout | Choose the behavioral interface |
| Robot state tasks | Single- and multi-key state improve fulfillment | Establish action consequences |
| Correction during work | Timely uptake; no detected success gain | Separate revision from execution |
| Capacity/load sweep | Bounded state trades accuracy for storage | Characterize finite capacity |

## 3 Related Work

**Agent runtimes and continuity.** ReAct interleaves reasoning and environmental actions (Yao et al., 2023). Generative Agents retrieves experiences and reflections to plan behavior (Park et al., 2023), while Reflexion retains verbal feedback across trials (Shinn et al., 2023). MemGPT organizes context as an operating-system-style hierarchy (Packer et al., 2023); recent benchmarks study long-duration, real-clock tasks (Zhu et al., 2026) and long-term interactive memory (Wu et al., 2024). CSI gives current records explicit creation, correction, and invalidation operations and connects those transitions to subsequent answers and actions.

**Elapsed time and temporal state.** Chronometric Injection supplies elapsed seconds through gated residual-stream modulation while keeping visible tokens unchanged (Siavoshian et al., 2026a). ChronoState combines this hidden time input with visible symbolic task state for supervised temporal action selection (Siavoshian et al., 2026b). CSI's $\Theta_t$ specifies occurrence, arrival, and validity at the agent-runtime level; a learned temporal input could expose that metadata to bounded model calls.

**Recurrent and persistent state.** LiveMem places gated delta-rule recurrence (GDN2) beside frozen Qwen3-4B so information can survive attention and KV eviction (Liu et al., 2026; Yang et al., 2025; 2024). We retain this substrate and add explicit creation, correction, invalidation, and rejection of protected records.

**Selection and lifecycle.** Memory Caching studies selective recurrent memory (Behrouz et al., 2026); Memento uses query-related memory for streaming video (Jiang et al., 2026); online episodic systems measure quality/storage tradeoffs (Lando et al., 2025); and Titans uses surprise as a memory signal (Behrouz et al., 2024). Our factorial study varies write pressure, provenance, lifecycle state, and future demand independently to identify which factors change the preferred state transition.

**Evaluation.** RoboCasa365 and OpenWAM provide the simulation and frozen actor for our single-task drawer evaluation (Nasiriany et al., 2026; Wang et al., 2026).

## 4 Implementing the Agent's Long-Term Memory

The language implementation combines protected records and explicit state-update rules with learned or exact retrieval. LiveMem supplies adaptive GDN2 working state beside frozen Qwen3-4B. The separate robot implementation converts retained preferences into instructions for a frozen actor through a symbolic router.

The protected state is a bounded collection

$$P_t = \{(k_i, v_i, o_i, z_i)\}_{i=1}^{B}, \tag{3}$$

where $k_i$ is a canonical entity–relation key, $v_i$ its current value, $o_i$ occupancy, and $z_i$ validity. The lifecycle controller selects

$$a_t \in \{\textsc{Reject}, \textsc{Working}, \textsc{Create}, \textsc{Update}, \textsc{Invalidate}, \textsc{NoWrite}\}. \tag{4}$$

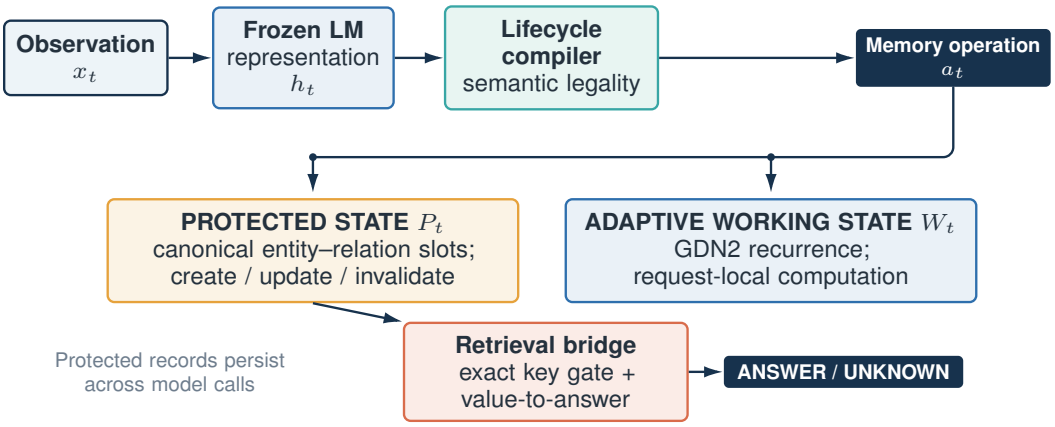


Figure 2: **Language implementation with the gated learned reader.** The lifecycle compiler governs protected records while GDN2 remains available for working computation. The learned bridge reads current values; an exact-key gate returns unknown for absent or invalid inferred keys.

### 4.1 Semantic lifecycle compiler

The front end canonicalizes entity, relation, current value, provenance, and persistence utility without seeing future probes or answers. Unreliable observations are rejected; transient observations route to working state; a durable fact with no matching key creates a slot; the same value causes no write; a changed value updates the matching slot; and an authenticated retraction invalidates it while retaining a tombstone. Below its confidence threshold, the compiler makes no protected-state mutation.

**Future utility.** We compare legal actions by executing each against the same future event sequence (Appendices J and N). A plan-conditioned estimator is compared with the same model receiving no task-plan input; Appendix O reports the results.

### 4.2 Learned and exact readout

The learned reader uses frozen embeddings and four bias-free rank-128 projections for queries, keys, values, and answer choices. Query/key similarities support differentiable soft slot attention during training; value/choice similarities produce differentiable answer scores. The interface is summarized schematically as

$$\alpha = \mathrm{Attention}_\theta(e_q, \{e_i^k\}), \qquad s = \mathrm{Readout}_\theta(\alpha, \{e_i^v\}, \{e_j^c\}), \tag{5}$$

where $\theta$ contains the four projections, key match scale and threshold, and key and value temperatures, with no answer-bias parameter. The final answer is selected by $\arg\max_j s_j$ at inference. A structural exact-key gate returns UNKNOWN if the inferred query key is absent or invalid. It uses no gold key or future answer.

An exact reader instead retrieves the record matching the same inferred entity–relation key. We test this with either the same learned answer decoder or deterministic normalized value matching. Both variants use the same records and inferred keys as the learned reader. Figure 2 depicts the learned implementation; exact readout substitutes for its retrieval bridge.

### 4.3 Supervision across state trajectories

Training includes insertion, distractors, request resets, updates, retractions, and retrieval. Answer, key, value, calibration, and margin losses have weights $1, .5, .5, .5, .25$, with margin 1. AdamW uses learning rate $3 \times 10^{-4}$ and one update per episode after averaging probe losses. The front end and embedding encoder remain frozen. Soft attention transmits gradients through query/key selection; symbolic writes and the inference legality gate remain discrete. Calibration uses

$$\mathcal{L}_{\mathrm{cal}} = \mathrm{BCE}(p_{\mathrm{match}} p_{\mathrm{live}}, y_{\mathrm{known}}), \tag{6}$$

with no separate prediction head. Its gradients reach query/key projections, key scale and threshold, and key-temperature-dependent attention; they do not reach value/choice projections. Competing occupied slots and non-target answer choices supply negatives without a separate negative sampler. Appendix I gives optimization details and gradient paths.

# 5 Evaluating the Agent's Durable State

Language experiments test retrieval after updates, retractions, and request resets; robot experiments test physical fulfillment through a separate symbolic router and frozen actor.

## 5.1 Learned state across language interactions

**Persistent state.** The initial LiveMem-RL bounded-memory comparisons use the same 79.09 MB allocation, including 49.16 kB reserved for the six-record reader, even when it is unused. This budget excludes model weights and encoder computation (exact allocations in Appendix C).

**Task generation.** The controlled NESS++ generator supplies executable lifecycle actions and produces facts, distractors, corrections, retractions, reliability conflicts, request boundaries, and later questions. For example, a cup preference changes from blue to green; after a request reset, the correct answer is green, and after retraction it is unknown. Answer-choice rotations control position effects; Appendix G defines the controls.

**Reader comparison.** Both readers use the same frozen lifecycle controller and working-state branch. The *original-reader baseline* retains the original neural readout; the learned-reader condition adds the trained retrieval bridge and key-validity gate.

**Evaluation protocol.** Architecture and checkpoint selection used development data. We then fixed three disjoint evaluation sets: six development seeds, 12 primary seeds, and 12 confirmation seeds, each with six final questions under a lexical-shift condition (Appendix C). The protocol was fixed before inspecting evaluation outcomes. Primary evaluation required all development criteria to pass; confirmation required all primary criteria to pass. Each 12-seed panel requires paired accuracy gains, at least ten improved seeds, closure of at least 75% of the control-to-oracle gap, unchanged working-state accuracy, matched memory, and no new key-gate errors (Appendix D).

**Matched exact readers.** A fresh 12-seed evaluation uses six probes per seed, identical episode fingerprints and targets, and three readers: learned retrieval, exact-key retrieval with the learned decoder, and exact-key retrieval with deterministic normalized value matching. Two bridge representations are tested: Qwen3-Embedding-0.6B and Qwen2.5-3B. This changes the reader representation, not the frozen Qwen3-4B working-state backbone. Seeds, checkpoints, and normalization rules are fixed before evaluation. Qwen3 results became visible before the Qwen2.5 arm finished; the latter resumed without tuning or changing its fixed sample (Appendix P).

**Paired analysis.** For experimental unit $g$ and comparator $c$, the success-rate contrast is

$$\widehat{\Delta}_c = \frac{1}{n}\sum_{g=1}^{n}(Y_{g,\text{persistent}} - Y_{g,c}), \tag{7}$$

where $Y$ indicates corrected-target physical success. Scenario pairing holds the assigned scene and sampling-noise prefix constant, while allowing observations and rollout lengths to diverge after actions differ. Action frames are not independent replicates. Language comparisons instead compute a probe-accuracy difference for each episode seed.

Language intervals resample whole seeds to preserve within-episode dependence; exact two-sided sign tests exclude ties. Robot intervals resample paired scenario groups, with exact McNemar tests. The 200-group confirmation uses 97.5% intervals and Holm adjustment for its two primary contrasts, and a prespecified $-10$-point non-inferiority margin against replay. Non-inferiority requires the 95% paired lower bound to exceed that margin; equal marginal success rates alone are insufficient. The two 97.5% primary intervals target at least 95% simultaneous coverage under the planned analysis.

### 5.2 Retained state and physical action

The frozen OpenWAM actor passed a 50-rollout RoboCasa365 `OpenDrawer` motor assessment with 47 successes. All memory conditions within a scenario share scene assignment and scenario-keyed sampling noise. Evaluator annotations are checked against the target and removed from actor observations. Success requires the environment-selected target drawer to be physically open, independently of the router instruction.

The original 105-group evaluation is followed by a separate fixed-sample 200-group confirmation. Each group supplies an initial preference, a correction or confirmation, distractors, a request boundary, and a follow-up task. Persistence retains the current record, reset clears it, the recent window loses it among distractors, and replay reparses the transcript. Missing state triggers a preassigned fallback. A separate 128-group study tests joint use of four state keys after multiple revisions. Scene and shared-action-noise-prefix checks validate pairing; the four-field specification remains incomplete (Appendix L).

For online correction, 120 eligible pairs balance early, middle, and late arrival over left/right preferences. A correction switches the side during rollout. One arm updates actionable state at a scheduled checkpoint; the other retains the event but applies it after the scored horizon. The actor, correction, and action budget are matched. Corrected-target physical success is the primary outcome; state uptake is a separate process measure. Appendices M and P give the protocols.

### 5.3 Capacity and temporal load

A deterministic protected-state runtime, without a learned reader or robot actor, is evaluated over 15,000 factorial cells and 60,000 episodes. Capacity, active keys, distractors, revisions, request boundaries, and invalidation rate vary jointly. Strict persistence, reset, a 256-token window, and replay receive paired event streams over the factorial grid

$$B \in \{2, 4, 8, 16, 32\}, \qquad K \in \{1, 2, 4, 8, 16, 32\},$$
$$D \in \{0, 8, 32, 128, 512\}, \qquad R \in \{0, 1, 2, 4, 8\},$$
$$H \in \{1, 2, 4, 8, 16\}, \qquad \rho \in \{0, .1, .25, .5\}.$$

Here $B$ is protected capacity, $K$ active keys, $D$ distractor count, $R$ revisions per key, $H$ request boundaries, and $\rho$ invalidation rate. Each cell has four generated replicates. Exact accuracy, stale reuse, and hard-capacity violations are recorded separately: a system can obey its storage bound while failing to retain a needed value (Appendix P).

## 6 Evidence for Coherent Behavior Across Time

### 6.1 Continued updates motivate protected state

Keeping 128 recurrent writes fixed while extending the stream from 128 to 2,048 steps preserves recall, whereas additional writes can damage retained associations (random-association sweeps in Appendix O).

### 6.2 Changing evidence requires distinct transitions

The factorial study executes alternative lifecycle actions across 480 decision blocks and 1,920 future task conditions: target retrieval, no-target retrieval, capacity competition, and later correction. Changing only the future condition changes the optimal action in every decision block; optima under neural answer loss agree with those under the constructed task costs on 85.3% of conditions. After correcting the evaluation, the prospective estimator did not outperform the same model without access to the task plan; Appendices N and O report the full comparison and interventions.

### 6.3 A reader-only intervention localizes the failure

On the development evaluation, the controller without memory truncation raises final accuracy from 0.694 to 0.750, but leaves nine errors. Holding its six-slot state fixed, replacing only the neural reader

Table 2: Language evaluation. The original-reader baseline was the strongest non-oracle control in the initial matched-budget evaluation. Pooled results are descriptive. W/T/L denotes wins, ties, and losses across seeds.

| Evaluation | Learned reader | Original reader | W/T/L | Sign $p$ |
|---|---|---|---|---|
| Development (6 seeds) | 32/36 | 23/36 | 6/0/0 | .03125 |
| Primary (12 seeds) | **72/72** | 56/72 | 10/2/0 | .001953 |
| Confirmation (12 seeds) | **68/72** | 58/72 | 8/4/0 | .007812 |
| Primary+confirmation | **140/144** | 114/144 | 18/6/0 | $7.63\times10^{-6}$ |

with an exact query raises final accuracy to 1.000 and persistent transition-probe accuracy from 0.903 to 1.000.

This development intervention uses an oracle query that directly accesses the correct stored record (Appendix O).

### 6.4 Held-out evaluation tests the learned reader

The learned reader answers 72/72 primary probes correctly versus 56/72 for the original reader, with ten improving seeds and two ties; the 95% seed-cluster interval for the gain is $[13.89, 31.94]$ points, passing all primary acceptance criteria (Table 2; Appendix D). Confirmation yields 68/72 versus 58/72, with a 95% interval for the gain of $[6.94, 20.83]$ points, eight improving seeds, and four ties. Confirmation fails two criteria: fewer than ten seeds improve, and the key gate introduces two errors while correcting one. Working-state accuracy is unchanged in both evaluations.

Replay reaches 94.44% on primary evaluation and 93.06% on confirmation, using 223.99 MB of active state versus 79.09 MB for the bounded conditions.

### 6.5 Exact lookup outperforms learned readout

On the fresh matched panel, deterministic exact lookup answers 64/72 probes correctly for both reader representations (Table 3). The Qwen3 bridge answers 54/72; its difference from deterministic lookup is $-13.89$ points (95% seed-cluster interval $[-20.83, -6.94]$; Holm-adjusted sign $p = .0156$). Exact retrieval with the same learned decoder answers 57/72; the bridge-minus-comparator difference is $-4.17$ points ($[-11.11, 0]$; adjusted $p = .50$).

Table 3: Matched readers on 72 probes per representation. All readers use the same inferred keys, state records, and targets.

| Reader | Qwen3 embedding | Qwen2.5-3B |
|---|---|---|
| Learned bridge | 75.00% | 31.94% |
| Exact key + learned decoder | 79.17% | 63.89% |
| Exact key + deterministic decoder | **88.89%** | **88.89%** |

Bridge accuracy is 31.94% with Qwen2.5 representations versus 75.00% with Qwen3; deterministic decoding remains at 88.89%. This panel uses seeds disjoint from the earlier primary and confirmation evaluations.

### 6.6 Persistent state improves target-drawer success

The fixed-sample confirmation completes all 200 groups without scene or shared-noise-prefix mismatches and yields 186 successes with persistence, 99 with reset, 95 with the recent window, and 186 with replay (Figure 3). Persistence gains 43.5 percentage points over reset (97.5% paired interval $[35.5, 51.5]$) and 45.5 over the window ($[37.5, 53.5]$). Holm-adjusted McNemar $p$-values are $3.38 \times 10^{-24}$ and $1.62 \times 10^{-27}$, respectively.

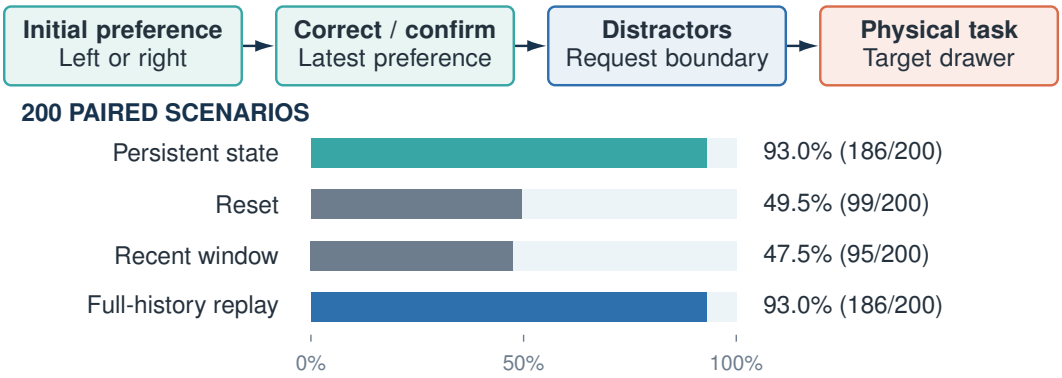


Figure 3: **Fixed-sample confirmation of embodied continuity.** Preference events precede physical rollout; success is scored against the environment-selected target drawer. Bars show observed rates; the text reports paired-effect intervals.

Against replay, the paired difference is zero (95% interval $[-3, 3]$ points), meeting the prespecified $-10$-point non-inferiority margin. Appendix M reports the earlier 105-group evaluation and its sampling procedure.

**Four-key state composition.** Persistence succeeds in 122/128 groups, reset in 64/128, the window in 68/128, and replay in 123/128. The paired gains over reset and window are 45.31 points (95% interval $[36.72, 53.91]$) and 42.19 ($[33.59, 50.78]$). Persistence uses 149 mean visible bytes versus 6,856 for replay, about one forty-sixth as much for this event sequence. Persistence and replay each route the correct instruction in 128/128 groups. The fixed sample completed after interruption and resumption, with scene and shared-action-noise-prefix checks passing.

### 6.7 Correction uptake during physical execution

Checkpoint incorporation updates state in 120/120 eligible pairs with zero logged control-step delay; wall-clock uptake was not measured. Deferred state retains the old preference until the post-horizon flush. Corrected-target physical success is 6/120 versus 8/120: a paired difference of $-1.67$ points (95% interval $[-4.17, 0]$; McNemar $p = .50$). Both arms fail in 112 pairs.

### 6.8 Accuracy under a fixed state capacity

Across the 60,000 generated episodes, strict persistence achieves 81.18% exact accuracy, reset 21.24%, the window 43.78%, and replay 100%. Persistence has zero hard-capacity violations and zero stale reuse (Appendix P).

## 7 From State Continuity to Responsive Action

**Current state as an execution constraint.** A transcript records that a preference changed; an executable state must also identify which preference governs the next action. CSI assigns that responsibility to lifecycle transitions and dispatch. In the proposed runtime, a correction changes the current record and prompts reconsideration of work that depends on it. For example, cancelling a delivery should invalidate the pending dispatch even when the earlier destination remains in history. This requires dependencies between commitments and unfinished work beyond storing and retrieving messages. The robot router implements the simpler case in which the current preference determines a new task's instruction; general dependency tracking remains to be evaluated.

**A correct record does not guarantee a correct action.** The two behavioral interfaces fail at different points. In the language diagnostic, replacing the reader resolves errors without changing stored records. In the online robot experiment, the record changes at the scheduled checkpoint, yet both arms fail in 112 of 120 pairs. Those observations do not identify whether the actor could recover

from the state reached when the correction arrived. A useful next control would give the actor the corrected instruction from matched intermediate simulator states and first establish that the remaining task is achievable within the action budget. The timing comparison could then isolate the cost of delayed incorporation from failure to execute an already available correction. Wall-clock uptake and completion of the obsolete target should be measured alongside corrected-target success.

**Choose retrieval by the information it must recover.** The matched panel provides canonical inferred keys and explicit current values, a setting in which deterministic lookup outperforms learned retrieval. CSI can use that simpler interface without changing its state lifecycle. A learned reader must justify its additional parameters on tasks where exact matching is inadequate, such as references that span several records or queries with incomplete keys. These tasks should hold the stored records and front end fixed so that gains can be attributed to retrieval. Representation sensitivity also makes a second embedding model a useful control rather than evidence of broader task coverage.

**Retention under finite capacity.** The capacity sweep separates obeying a storage limit from preserving enough information to answer correctly: strict persistence has no capacity violations or stale reuse, but its accuracy remains below replay. Selecting what to retain therefore remains a distinct problem even when updates are legal. Future capacity comparisons should report accuracy against active-key load and revision frequency separately; the pooled grid average cannot identify which condition causes loss. This would test whether a retention policy preserves commitments that continuing goals will actually require.

## 8 Scope and Limitations

All robot studies use OpenDrawer and one frozen actor, including the four-key extension. Wall-clock correction cost, old-target completion, and deadline-sensitive dispatch remain unmeasured.

Language comparisons cover one synthetic lexical-shift condition and show substantial reader-representation sensitivity. The capacity sweep evaluates a deterministic reference runtime with four replicates per cell; its pooled accuracy does not identify accuracy as a function of load. Visible-memory comparisons exclude actor weights and simulator state. Missing protocol and historical implementation details are listed in Appendix L.

## 9 Conclusion

Continuous Stateful Intelligence makes the agent's lifetime the unit of execution: state survives each response and changes with subsequent evidence. In the 200-scenario robot confirmation, retaining current preferences raises success from 49.5% after reset to 93.0%; the four-key study extends this benefit to joint state use. Deterministic exact lookup is the best reader on the matched language panel. During ongoing robot activity, all scheduled corrections update state, but we detect no improvement in physical completion. Adapting the actor's remaining trajectory after a correction is the next step toward responsive execution.

## AI Use Statement

Generative AI tools assisted literature discovery, experimental design, implementation, analysis, and writing. Human authors directed the research and remain responsible for its claims and submission.

## Reproducibility Statement

Exact historical trainer source and the original launch command were not recovered. Experiment code, checkpoints, and raw traces are not included in this submission; we plan to release them with the analysis scripts.

## REFERENCES


Zhichen Liu, Ruihan Sun, Hengjie Yang, Zipeng Wu, Zhaohan Chen, Xiaofan Zhang, and Yang Xu. LiveMem: Maintaining Memory State Continuity in Long-Running LLM Inference. *arXiv preprint arXiv:2608.02515*, 2026.

Hongxiang Jiang, Zengrui Ge, Guo Chen, Qixiong Wang, Jile Jiao, Xuetao Feng, Yuan Wang, and Yan Wang. Memento: Toward an All-Day Proactive Assistant for Ultra-Long Streaming Video. In *International Conference on Learning Representations*, 2026.

Ali Behrouz, Zeman Li, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Memory Caching: RNNs with Growing Memory. *arXiv preprint arXiv:2602.24281*, 2026.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to Memorize at Test Time. *arXiv preprint arXiv:2501.00663*, 2024.

Deyao Zhu et al. EdgeBench: Unveiling Scaling Laws of Learning from Real-World Environments. *arXiv preprint arXiv:2607.05155*, 2026.

Giuseppe Lando, Rosario Forte, Giovanni Maria Farinella, and Antonino Furnari. How Far Can Off-the-Shelf Multimodal Large Language Models Go in Online Episodic Memory Question Answering? *arXiv preprint arXiv:2506.16450*, 2025.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated Delta Networks: Improving Mamba2 with Delta Rule. *arXiv preprint arXiv:2412.06464*, 2024.

An Yang et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. *arXiv preprint arXiv:2410.10813*, 2024.

Charles Packer et al. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560*, 2023.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A Large-Scale Simulation Framework for Training and Benchmarking Generalist Robots. *arXiv preprint arXiv:2603.04356*, 2026.

Yuran Wang et al. OpenWAM: An Open, Modular Exploration Towards Systematic World–Action Model Pretraining. *arXiv preprint arXiv:2609.07398*, 2026.

Sam Siavoshian, Omar Ramadan, Amir Kashif Saeed, Benjamin A. Johnson, Amin Mohamed El-Amin Diab, and Benjamin M. Rodriguez. Chronometric Injection: A Residual-Stream Channel for Elapsed-Time Conditioning in Frozen Language Models. Preprint, 2026a. `https://doi.org/10.5281/zenodo.22715818`.

Sam Siavoshian, Omar Ramadan, Amir K. Saeed, Benjamin A. Johnson, Amin Mohamed El-Amin Diab, and Benjamin M. Rodriguez. ChronoState: Hidden Elapsed-Time Conditioning for Temporal-State Action Selection in Frozen-Backbone Language Models. *arXiv preprint arXiv:2608.09124*, 2026b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. *arXiv preprint arXiv:2304.03442*, 2023.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint arXiv:2303.11366*, 2023.

## A Measurement Controls

State-growth exponents are fitted to the saturated run tail, excluding fixed prompt offsets and restart discontinuities. Forced-choice scores are averaged over cyclic option rotations. Write-selection labels distinguish observable substantive events from filler; they do not expose the random choice of a future probe. Timestamp baselines receive the information needed to compute elapsed time. Matched baselines use the same generator version.

Uncertainty follows the experimental unit: language intervals resample whole seeds and robot intervals resample paired scenario groups. Pooling probe counts does not replace those paired analyses. Trainer checks include separable and pure-noise controls; recurrent-state checks compare warm and cold state under slow decay.

## B Factorial Variables and Actions

| Variable | Values |
|---|---|
| Entity correspondence | same, different |
| Relation correspondence | same, different |
| Value relation | same, changed, retraction |
| Provenance | reliable, unreliable |
| Utility | persistent, transient |
| Protected match | matching slot, no slot |
| GDN2 pressure | low, high |
| Future demand | target, no-target, capacity, correction |

Every valid state executes every legal action from

$$\{\textsc{Reject}, \textsc{Working}, \textsc{Create}, \textsc{Update}, \textsc{Invalidate}, \textsc{NoWrite}\}.$$

Before transition costs are constructed, validation checks matrix completeness, decision-snapshot agreement, and equal returns for transitions classified as physically equivalent. A failed check invalidates the return matrix. The equivalence criterion is not specified in the retained protocol (Appendix L).

## C Initial Language Evaluation Protocol

**Evaluation sets.** The six-seed development evaluation in Table 2 is distinct from the earlier reader-only diagnostic. All three final evaluation sets use the same lexical-shift condition; its exact lexical transformation was not retained (Appendix L).

**Exact state accounting.** The 36-layer recurrent allocation is 79,036,416 bytes. The reader reserves 49,164 bytes, giving every bounded condition 79,085,580 bytes, including conditions that do not use the reader. Full-history replay reaches 223,985,664 bytes. Main-text quantities use decimal MB and kB and are rounded; all allocations exclude model weights and encoder computation.

**Seeds and evaluation order.** Development seeds are 965000–965005, primary seeds are 965100–965111, and confirmation seeds are 965200–965211. The bridge checkpoint remains fixed across evaluations. Confirmation begins only after every primary acceptance criterion passes.

**Paired uncertainty.** For seed $s$, let $d_s = Q_s^{\text{reader}} - Q_s^{\text{control}}$. The sign test conditions on $n_+ = |\{s : d_s > 0\}|$ and $n_- = |\{s : d_s < 0\}|$, discards ties, and reports

$$p_{\text{sign}} = 2\min\{\Pr[\text{Bin}(n_+ + n_-, 1/2) \le n_+], \Pr[\text{Bin}(n_+ + n_-, 1/2) \ge n_+]\}, \tag{8}$$

clipped at one. Bootstrap intervals resample seeds rather than individual probes, preserving within-episode dependence.

## D PRESPECIFIED ACCEPTANCE CRITERIA

| Criterion | Primary | Confirmation |
|---|---|---|
| Beats initial matched non-oracle controls | pass | pass |
| $\geq 10/12$ positive paired seeds | pass (10) | fail (8) |
| Oracle-gap closure $\geq 0.75$ | pass | pass |
| Working GDN2 preserved | pass | pass |
| Identical bounded byte charge | pass | pass |
| Exact-key gate harms zero rows | pass (0) | fail (2) |
| All criteria | pass | fail |

The primary exact-key intervention changes five final predictions, corrects five, and harms zero. On confirmation it changes three, corrects one, and harms two. The full learned-reader system has no losing seed against the original reader on either panel; the key-gate intervention instead compares gated and ungated predictions from the same learned checkpoint.

## E PER-SEED PAIRED DIFFERENCES

Each entry below is the learned reader's final accuracy minus the strongest matched-budget non-oracle accuracy for the same episode seed. Six probes per seed make the unit $1/6$.

$$\text{Primary: } (1, 3, 2, 1, 0, 1, 1, 1, 3, 0, 1, 2)/6,$$
$$\text{Confirmation: } (0, 1, 1, 0, 1, 1, 1, 0, 2, 0, 2, 1)/6.$$

The pooled sign test conditions on the 18 non-tied seeds and yields $2/2^{18} = 7.62939453125 \times 10^{-6}$. This descriptive pooled result does not replace the prespecified confirmation decision.

## F LIFECYCLE TRANSITION SEMANTICS

Let $\kappa(x_t)$ be the canonical entity–relation key, $\nu(x_t)$ the canonical value (possibly null for a retraction), $p_t$ a calibrated provenance predicate, and $u_t$ a persistence-utility predicate. The protected branch implements

$$a_t = \textsc{Reject} \quad \text{if } \neg p_t, \tag{9}$$
$$a_t = \textsc{Working} \quad \text{if } p_t \wedge \neg u_t, \tag{10}$$
$$a_t = \textsc{Create} \quad \text{if } p_t u_t \wedge \neg\, \text{exists}(\kappa), \tag{11}$$
$$a_t = \textsc{NoWrite} \quad \text{if } p_t u_t \wedge \text{live}(\kappa) \wedge \nu = v_\kappa, \tag{12}$$
$$a_t = \textsc{Invalidate} \quad \text{if } p_t u_t \wedge \text{live}(\kappa) \wedge \nu = \varnothing, \tag{13}$$
$$a_t = \textsc{Update} \quad \text{otherwise when } p_t u_t \wedge \text{live}(\kappa). \tag{14}$$

A malformed CREATE or UPDATE carrying a null canonical value fails closed: it increments a diagnostic counter and performs no protected mutation. The rule list does not specify how an invalidated key becomes live again; that case remains an implementation detail to recover (Appendix L). Appendix H specifies replacement when a legal creation encounters a full six-slot table.

## G MATCHED-BUDGET CONTROL DEFINITIONS

- **Dense write-all.** Every eligible observation updates recurrent state.
- **Semantic heuristic.** A fixed surface-semantic rule selects durable events without learning from episode returns.
- **Semantic-only.** The semantic branch acts without pressure or prospective demand information.
- **Pressure-only.** GDN2 familiarity/update pressure controls writes without semantic lifecycle predicates.
- **Prospective hybrid.** Semantic state and observable demand are combined through expected suffix return.
- **Plan-hidden.** The prospective branch receives no plan signal.

- **Rate-matched random.** Sixteen deterministic random policies match the candidate write rate; their mean is reported.
- **Reset/window.** No protected state crosses the request reset.
- **Original-reader baseline.** The frozen lifecycle compiler uses the historical neural reader and the same working branch.
- **Bounded oracle.** Generator-known control decisions use the historical reader; because its reader differs from the learned reader, it is not a universal readout upper bound.
- **Direct-slot diagnostic.** An oracle query supplies the correct stored record, whose canonical value is read exactly. Unlike the matched exact reader, this diagnostic does not infer the query key.
- **Growing replay.** History remains active; this is a growing-state quality comparator rather than a matched bounded policy.

## H Protected and Working-State Update Equations

Write the protected table as $P_t = (K_t, V_t, O_t, Z_t)$ with capacity $B = 6$. For selected slot $i_t$, create sets

$$(K_{t+1,i_t}, V_{t+1,i_t}, O_{t+1,i_t}, Z_{t+1,i_t}) = (k_t, v_t, 1, 1), \tag{15}$$

update preserves the key and replaces the current value,

$$(K_{t+1,i_t}, V_{t+1,i_t}, Z_{t+1,i_t}) = (K_{t,i_t}, v_t, 1), \tag{16}$$

and invalidation retains identity while clearing validity,

$$(K_{t+1,i_t}, V_{t+1,i_t}, Z_{t+1,i_t}) = (K_{t,i_t}, V_{t,i_t}, 0). \tag{17}$$

Unselected slots remain unchanged. FIFO replacement is used only when a legal create encounters a full table in the initial six-slot language comparison; all its bounded arms use this policy.

The working branch retains the original recurrent update. The following schematic illustrates one head of gated delta recurrence; it is not a full implementation specification:

$$\widetilde{W}_t = d_t W_t, \tag{18}$$

$$W_{t+1} = \widetilde{W}_t + g_t(v_t - \widetilde{W}_t k_t)k_t^\top, \tag{19}$$

$$o_t = W_{t+1} q_t, \tag{20}$$

where $d_t$ controls decay and $g_t$ update strength. Protected lifecycle operations do not freeze these dynamics: episodic identity remains stable in $P_t$, while $W_t$ continues adapting to request-local computation.

## I Reader Training and Reconstruction

The evaluated gated reader uses a previously trained bridge; the structural exact-key/unknown rule is added at inference without retraining that bridge. The recovered checkpoint, configuration, and current trainer describe four bias-free rank-128 projections $(A_q, A_k, A_v, A_c)$ and four learned scalars: raw key scale, key threshold, log key temperature, and log value temperature. No answer bias or separate calibration head is present.

Training uses differentiable soft slot attention and soft choice logits. There is no hard slot argmax or straight-through estimator in training. Discrete answer selection occurs at inference. Thus the training objective can differentiate through slot attention, while symbolic lifecycle writes and the structural inference gate remain outside the gradient path.

The objective is

$$\mathcal{L} = \mathcal{L}_{\text{answer}} + .5\mathcal{L}_{\text{key}} + .5\mathcal{L}_{\text{value}} + .5\mathcal{L}_{\text{calibration}} + .25\mathcal{L}_{\text{margin}}. \tag{21}$$

Answer supervision uses the correct choice; key/value terms train matching representations. Competing occupied slots and non-target answer choices supply negatives; there is no separate negative sampler. The margin is 1. Calibration uses

$$\mathcal{L}_{\text{calibration}} = \text{BCE}(p_{\text{match}} p_{\text{live}}, y_{\text{known}}). \tag{22}$$

Its gradients reach query/key projections, key match scale and threshold, and key-temperature-dependent attention, but not value/choice projections. The embedding inputs are frozen; query/key and choice representations are normalized within the bridge.

AdamW uses learning rate $3 \times 10^{-4}$. Transition and final-probe losses are averaged within each generated episode, followed by one optimizer update. Episode order is deterministically shuffled each epoch. There is no learning-rate schedule, early stopping, or best-checkpoint selection; the final fixed epoch is saved. The batch size of 32 applies to frozen embedding extraction, not gradient minibatches. The epoch count and complete numerical definitions of the soft readout functions remain unavailable.

**Historical reconstruction.** The archived trainer identifier differs from the current checked-in trainer. The original source snapshot and literal launch command were not recovered. These implementation details therefore describe the recovered effective configuration and current code, rather than byte-identical historical reproduction.

## J Factorial Identification and Future Dependence

For semantic state $S$ and suffix family $\omega$, every legal action is executed to obtain $J(S, a, \omega)$. Exact- and uncertain-demand decisions are

$$a^*(S, \omega) = \arg\min_a J(S, a, \omega), \tag{23}$$

$$a^*(S, p) = \arg\min_a \sum_\omega p(\omega \mid S) J(S, a, \omega). \tag{24}$$

Variable $x_i$ is irrelevant to a branch only if changing it while holding the remaining valid state fixed leaves the minimizing action invariant. If two future-task conditions agree on all observed state but differ in $a^*$, the observed state alone cannot determine the optimal action. Exact demand changes the optimum in all 480 decision blocks across 1,920 state–future combinations. This establishes future dependence of the counterfactual optimum in the controlled construction; it does not establish accurate prediction of an unknown future.

## K Controller Development Ablations

Inference-only semantic masks were off-distribution for a dense-scan checkpoint. Scalar write-label supervision did not optimize answer retrieval. Appendix O reports the subsequent lifecycle and reader interventions.

## L Unresolved Reproduction Details

The following details were not recovered from the experimental records and are needed for an exact rerun:

- **Language generation and lifecycle control:** the exact lexical-shift transformation, calibrated confidence threshold, physical-equivalence criterion in the action matrix, and handling of a key that exists but is no longer live.
- **Reader training:** the original trainer snapshot and launch command, epoch count, and numerical definitions of soft attention and readout. Appendix I distinguishes recovered settings from the historical implementation.
- **Matched exact lookup:** normalization and tie-handling rules, and an error breakdown separating inferred keys, slot selection, and decoding.
- **Four-key robotics:** field definitions, revision sequences, and the rule composing the four current values into the actor instruction.
- **Capacity sweep:** the false-unknown denominator and cell-level results needed to estimate accuracy conditional on load. The reported pooled average cannot recover those conditional effects.

Appendix M lists additional omissions for the original robot protocol. No pooled test combines the separate language and robot studies.

# M ORIGINAL 105-GROUP EMBODIED EVALUATION

This appendix specifies the task, scenario construction, physical success criterion, pairing, and statistical analysis for the 105-group robot evaluation.

## M.1 TASK QUALIFICATION AND SCENARIO CONSTRUCTION

The actor is frozen throughout memory evaluation. Its task-competence screen used 50 rollouts: 47 succeeded overall, comprising 23/25 left-drawer and 24/25 right-drawer trials. The locked minimums were 40/50 overall and 18/25 per side. This screen establishes competence under the initial drawer-task conditions; checkpoint training overlap with the target split is unknown.

Scenario generation uses seed 20260926. Initial-episode seeds start at 4,100,000, follow-up seeds at 4,200,000, and the seed-search limit is 10,000. All arms share the initial episode and target follow-up scene assignment within a group. The manifest assigns an initial preference, a corrected or confirmed preference, and an independent fallback side. Configurations were generated and shuffled before evaluation. The analyzed prefix is the first 105 IDs in that schedule and contains all eight initial/corrected/fallback cells, with 7–18 groups per cell; these configurations were not selected post hoc from observed outcomes. Fallback balance is reported for the plan. In the analyzed prefix, the fallback matched the intended target in 47/105 groups and conflicted in 58/105. Exact counts for each cell, left/right marginals, and correction/confirmation totals are not reported, so complete cell coverage does not imply exact balance in the prefix.

The event sequence consists of an explicit kitchen-drawer preference, a correction to the opposite side or confirmation of the same side, and eight unrelated kitchen updates. The memory interface processes this finite prefix before an explicit request boundary and a fresh follow-up request. Persistent state stores the latest matching statement under the canonical key (`kitchen drawer`, `preference`); repeated confirmations retain the same value. Reset clears state. The four-event window contains only distractors. Replay retains and parses the complete ordered transcript. Reset and window use the preassigned fallback. No future probe, task annotation, reward, or success label enters the router.

The router recognizes complete explicit preference statements by symbolic logic. It does not learn semantic acceptance, a neural write gate, future utility, or the learned protected-state reader. The actor receives an instruction naming the selected side, then consumes camera/proprioceptive observations and emits actions in a closed-loop simulator rollout. Memory events are not concurrently injected into that rollout. State initialization and resets occur at physical episode boundaries; the persistent arm separately carries its preference record across the request boundary.

**Verified endpoint-to-target binding.** We audited the runner and RoboCasa evaluator to verify that success is scored against the environment's selected target drawer, independently of the instruction chosen by the memory router. The plan assigns each follow-up scene to the corrected side. The runner checks the scene's evaluator-only task annotation against that target and aborts on a mismatch. An explicit observation allowlist removes annotations, reward, and success fields before observations reach the actor. RoboCasa's physical success check tests whether the selected target drawer has joint position at least 0.95 in the evaluated environment; it does not score the prompt or the router's chosen side.

Across all 420 arm-specific follow-up rows in the 105 complete groups, we found zero mismatches between the evaluator annotation and the expected target. Table 4 further separates fallback agreement from physical execution. Reset and window succeed frequently when their instruction matches the intended target, and rarely when it conflicts.

Table 4: Row-level endpoint audit, stratified by whether the preassigned fallback instruction matches the intended target. Both strata together recover each arm's overall successes. These are descriptive diagnostics.

| Arm | Match (47 groups) | Conflict (58 groups) | Total |
|---|---|---|---|
| Reset | 46/47 (97.9%) | 3/58 (5.2%) | 49/105 |
| Four-event window | 45/47 (95.7%) | 3/58 (5.2%) | 48/105 |

### M.2 Pairing, complete groups, and process deviations

A common per-action sampling stream is keyed by scenario. The pairing audit reports zero physical-scene mismatches and zero common-action-noise-prefix mismatches among the 105 analyzed groups. Actor-observation hashes differ in 37/105 groups and action-noise trace lengths in 76/105. These are paired scenario assignments, not identical observation or action trajectories.

**Sample selection and protocol deviations.** The evaluated set comprises the first 105 complete groups of a pre-generated, shuffled schedule originally planned for 500 groups. We stopped the run at this point without a predeclared interim stopping rule. The 527-row manifest contains 525 rows from complete groups and two rows from a subsequent partial group, which was excluded. Monitoring accessed individual first/last-row success values and a distinct-value count before stopping, but no arm totals or paired efficacy contrasts had then been calculated. We report nominal paired intervals and tests for the evaluated set; they do not incorporate a stopping adjustment, and the study is not described as completion of the original fixed-sample protocol.

### M.3 Locked efficacy rules and observed paired results

The co-primary planned comparisons were persistence versus reset and versus the four-event window. The analysis lock specifies 100,000 paired scenario-bootstrap draws using Python RNG seed 20260924, two-sided 97.5% percentile intervals for each co-primary contrast, exact two-sided McNemar tests, and Holm adjustment across the two tests at familywise $\alpha = 0.05$. The interval levels target Bonferroni simultaneous coverage of at least 95% under the planned analysis. A planned task-specific win requires both point differences to be at least 10 percentage points, both interval lower bounds above zero, and both Holm-adjusted $p$-values below .05. Replay is a separate planned non-inferiority comparison with margin $-10$ points and a 95% paired-bootstrap lower bound above that margin. Both reset and window contrasts cross all three numerical efficacy thresholds in the 105 observed pairs: their gains exceed 10 points, their 97.5% interval lower bounds are positive, and both Holm-adjusted $p$-values are below .05. These are the prelocked numerical efficacy rules applied to the evaluated set.

For scenario $g$ and comparator $c$, the paired difference is

$$d_{g,c} = Y_{g,\text{persistent}} - Y_{g,c}, \qquad \widehat{\Delta}_c = \frac{1}{105}\sum_{g=1}^{105} d_{g,c}, \tag{25}$$

where $Y$ indicates that the environment-selected target drawer is physically open. Let $b_c$ count persistent wins and $c_c$ persistent losses. The exact two-sided McNemar value is

$$p_c = \min(1,\ 2\Pr\left[\text{Bin}(b_c + c_c, 1/2) \le \min(b_c, c_c)\right]). \tag{26}$$

Table 5 reports the paired effect estimates and nominal confidence intervals. The group, not an action frame or episode row, is the resampling unit.

Table 5: Paired physical-success contrasts across 105 complete scenario groups. Differences and nominal confidence intervals are in percentage points. W/T/L counts scenario-level wins/ties/losses.

| Comparator | $\widehat{\Delta}$ | W/T/L | Exact $p$ | Holm $p$ | Interval |
|---|---|---|---|---|---|
| Reset | +48.6 | 51/54/0 | $8.88\times10^{-16}$ | $8.88\times10^{-16}$ | 97.5% [38.1, 59.0] |
| Window | +49.5 | 52/53/0 | $4.44\times10^{-16}$ | $8.88\times10^{-16}$ | 97.5% [39.0, 60.0] |
| Replay | +1.0 | 1/104/0 | 1.000 | — | 95% [0.0, 2.9] |

The aggregate counts reproduce the success rates, risk differences, exact tests, and Holm adjustment. Because each comparison has zero observed losses, its empirical paired-bootstrap distribution is $\text{Bin}(105, b_c/105)/105$; its quantiles agree with the reported rounded intervals. These quantiles are calculated analytically from aggregate discordance counts; the fixed-seed resamples were not regenerated from raw records. For replay, one positive pair and 104 ties yield a nonnegative empirical bootstrap distribution. Its lower bound exceeds the planned $-10$-point margin, but the early-stopped sample does not support our confirmatory non-inferiority claim; that claim uses the independent 200-group evaluation.

### M.4 Diagnostics and resource interpretation

We recorded preference evidence recalled and the latest preference incorporated into the action instruction in 105/105 persistent and replay cases and 0/105 reset and window cases. The "corrected preference reused" physical-follow-up diagnostic is 100/105 for persistence and 99/105 for replay. These fields describe stages of retention and use; they are not independent replications of physical success. The aggregate does not separate changed-preference trials from same-side confirmations.

Mean visible memory at follow-up is 48.6 bytes for persistence, 0.0 for reset, 200.0 for the window, and 492.9 for replay. The ratio $48.6/492.9 = 0.0986$ concerns the reported visible representation for this event sequence. Serialization details and accounting boundaries require the implementation; these values exclude actor weights, simulator state, and other runtime costs. They do not establish a scaling law or share the language evaluation's active-state definition.

Mean actor-reported latency is 12.74, 23.51, 23.88, and 12.71 seconds for persistent, reset, window, and replay arms, respectively. Measurement boundaries, dispersion, and isolated router costs are not reported. Success-dependent termination and different rollout lengths can affect these values. They remain descriptive actor timings, not evidence of a memory lookup or end-to-end systems speedup.

### M.5 Software environment

The evaluation uses RoboCasa 1.0.1, robosuite 1.5.2, MuJoCo 3.3.1, Gymnasium 1.3.0, NumPy 2.2.5, and Python 3.11.15 on Linux aarch64. The actor was not trained or fine-tuned for this memory evaluation.

**Unavailable protocol details.** The retained protocol omits checkpoint access, observation preprocessing, cameras, rollout horizon, control rate, actor sampling, arm order, seed-search acceptance, the shared initial episode's role, correction/confirmation counts, byte serialization, latency boundaries, and diagnostic definitions. The one-success reset/window difference occurs in the target-matching stratum; its execution-level cause is not identified.

## N Counterfactual Identification of State Transitions

We vary state factors independently and execute alternative actions to identify which variables change the lowest-cost transition.

### N.1 Interventional state space

We represent each decision by

$$S = (E, R, V, P, U, M, G), \tag{27}$$

covering entity correspondence, relation correspondence, value relation (same, changed, or retracted), provenance, persistence utility, matching-slot existence, and GDN2 familiarity/update pressure. Invalid logical combinations are removed. The remaining variables are independently permuted so natural correlations—reliability with persistence, novelty with creation, or surprise with importance—cannot solve the task accidentally.

For every valid state and legal action, we physically execute the transition and evaluate four suffix families: target retrieval, no-target retrieval, capacity competition, and later correction. This yields the causal return table

$$\mathbf{J} = [J(S, a, \omega)]_{S,a,\omega}. \tag{28}$$

Controlled primary costs and neural answer losses are retained separately.

### N.2 From counterfactuals to a transition program

Intervening on one factor while holding all others constant reveals whether it belongs upstream of a decision. If changing recurrent pressure leaves the optimal semantic lifecycle action unchanged, pressure is excluded from that branch even if it remains useful for low-level write geometry. The measured return table determines the predicates retained in the transition program.

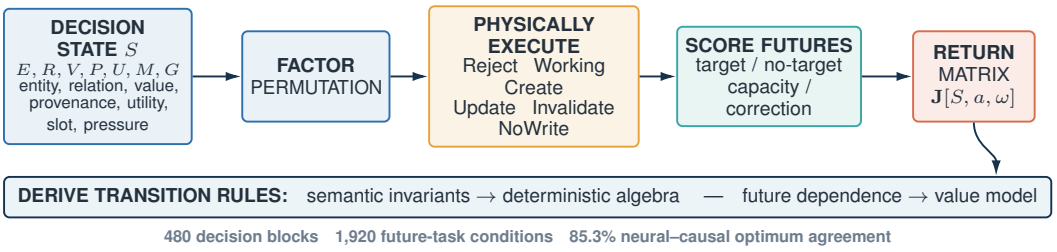


Figure 4: **Factorial memory-transition identification.** The protocol breaks natural correlations, executes every legal action, and scores its downstream consequences. The return matrix distinguishes state-dependent transition rules from actions whose ranking changes with the future task.

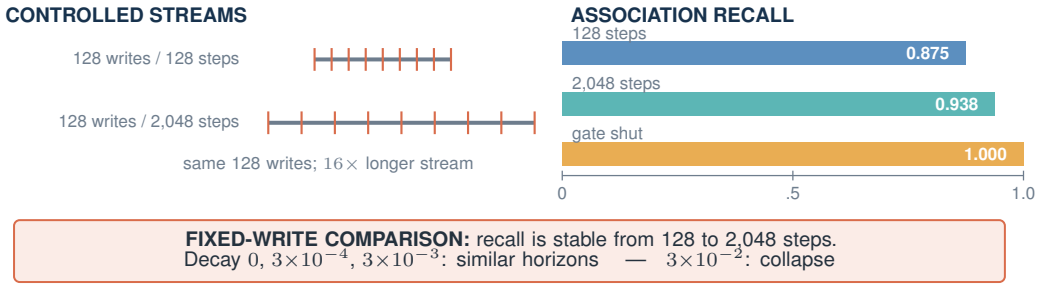


Figure 5: **Recall under fixed writes and varying sequence length.** With the number of writes fixed, extending the stream from 128 to 2,048 steps does not reduce recall; shutting the write gate preserves state exactly. Matched-step sweeps and decay ablations identify state modification, key correlation, and sufficiently large decay as the operative failure modes.

The experiment admits three outcomes. If $H(A^* \mid S) \approx 0$, a deterministic algebra is sufficient. If identical $S$ requires different actions, the state representation is missing a variable. If the difference is induced by future continuation, a distributional value model such as Eq. 24 is required.

# O Mechanism and Readout Diagnostics

## O.1 Write interference and sequence length

Keeping 128 writes fixed while extending the stream from 128 to 2,048 steps does not reduce recall (Figure 5). Closing the write gate preserves state exactly. At matched sequence length, write-rate sweeps and decay ablations show that later state modifications, key correlation, and sufficiently large decay reduce retention.

At eight heads with $64 \times 64$ state, the synthetic capacity ceiling is 128 random associations at recall at least 0.9, approximately 1.1 KB per retained association. At decay $10^{-2}$ the ceiling falls to 32. These are optimistic ceilings because random keys are less correlated than language keys.

## O.2 Substantive-event classification

Frozen representations expose a strong substantive-versus-filler signal (Table 6). A 3B linear head captures approximately 86% of the generator-derived Bayes-achievable F1. Similar fitted 3B and 7B performance suggests that the feature is already linearly accessible around 3B. Substantive-event classification does not specify whether an event should create, update, invalidate, remain temporary, or be rejected.

Table 6: Held-out write-selection diagnostic.

| Policy | Precision | Recall | F1 |
|---|---|---|---|
| Always write | 0.237 | 1.000 | 0.383 |
| Keyword rule | 0.277 | 0.525 | 0.363 |
| Learned head (3B) | 0.646 | 0.669 | **0.657** |
| Learned head (7B) | 0.633 | 0.655 | 0.644 |
| Bayes ceiling | 0.699 | 0.839 | 0.763 |

### O.3 Dependence of action choice on future demand

The factorial matrix contains 480 decision blocks and 1,920 future-task conditions. Changing only future demand changes the optimal action in every decision block. Actual neural-return optima overlap the controlled causal optimum on 85.3% of conditions. In this construction, identical present observations can require different actions under different future demand.

An initial cached-return analysis appeared to show that a quality-0.70 plan-conditioned controller closed 34.14% of the strongest-control gap on four replication seeds. An objective audit found that independently normalizing each demand branch changed 54 of 576 action argmins and did not preserve the joint expected-return decision. Under the corrected raw-preserving objective, the prospective branch did not beat its plan-hidden counterpart. A fresh 18-seed test then found zero quality-0.70 future-sensitive groups, although exact demand changed the optimum in all 18. Exact future demand changes the optimal action in this construction; the tested estimator does not reliably exploit uncertain demand.

### O.4 Retrieval from fixed protected state

The no-truncation compiler raises final accuracy from 0.694 to 0.750 on the earlier development set, but passes only three of five locked gates: four of six seeds improve and only 25% of the bounded-oracle gap is closed. Replacing the neural protected reader with an exact query of the compiler's frozen six-slot state raises final accuracy from 0.750 to 1.000 and persistent transition-probe accuracy from 0.903 to 1.000. This removes all nine final errors without changing the stored records.

The learned bridge reaches 0.944 final accuracy on held-out generator-driven engineering episodes. Frozen post-hoc evaluation on the earlier controller development set reaches 0.889, compared with 0.750 for the old reader and 0.917 for the bounded oracle. This closes 83.3% of the old-reader-to-oracle gap. Because only three seeds strictly improve while two tie and one regresses, the opened panel remains diagnostic rather than confirmatory.

## P Additional Evaluations and Protocol Details

### P.1 Matched readers and representation sensitivity

The fresh panel contains seeds 970600–970611 and six probes per seed. All three readers share compiler episode fingerprints, state, inferred keys, and targets. Exact retrieval uses the inferred entity–relation key, not a gold slot. Its learned-decoder variant retains the value/answer decoder; its deterministic variant uses normalized value matching.

Table 7: Learned bridge minus matched comparator, in percentage points. Intervals resample seeds; tests are exact two-sided seed sign tests.

| Representation | Comparator | Difference | 95% interval | Holm $p$ |
|---|---|---|---|---|
| Qwen3 | Exact + learned | $-4.17$ | $[-11.11, 0]$ | .50 |
| Qwen3 | Exact + deterministic | $-13.89$ | $[-20.83, -6.94]$ | .0156 |
| Qwen2.5 | Exact + learned | $-31.94$ | $[-41.67, -20.83]$ | .00586 |
| Qwen2.5 | Exact + deterministic | $-56.94$ | $[-66.67, -44.44]$ | .00391 |

The bridge difference between Qwen2.5 and Qwen3 is $-43.06$ points (95% interval $[-56.94, -27.78]$), with one winning and eleven losing seeds and sign $p = .00635$. Qwen3 evaluation output became visible before Qwen2.5 finished. Qwen2.5 resumed using the already fixed checkpoint, seeds, compiler episodes, and evaluator, with no tuning or seed replacement. This is a reader-representation comparison on one lexical-shift condition, not a second task-family evaluation.

The Qwen3 bridge protected tensor state occupies 49,164 bytes. The exact key/value map averages 229.1 serialized bytes plus 79 parser-metadata bytes; peak CUDA allocation is 1,280,488,960 bytes in that report. Tensor-state, serialized-map, and peak-device measurements have different boundaries.

### P.2 Correction during physical execution

The fixed sample contains 120 eligible pairs balanced over three arrival windows and two initial sides. Corrections switch to the opposite side. Checkpoint incorporation updates actionable state during rollout; deferred incorporation flushes the retained correction after the horizon. Physical success is 6/120 versus 8/120, with 112 joint failures, paired gain $-1.67$ points (95% interval $[-4.17, 0]$), and McNemar $p = .50$. Checkpoint state incorporates the correction in all 120 pairs and matches the corrected preference at the horizon. Logged delay is zero control steps. The deferred arm contains 43,090 post-arrival action records carrying the obsolete instruction; action records are not independent trials. CPU/accelerator timing and a separate old-target-completion outcome were not logged.

### P.3 Fixed-sample embodied confirmation

The independent sample completes all 200 planned groups, with zero scene mismatches and zero common-action-noise-prefix mismatches. Persistence and replay each succeed in 186 groups; reset in 99 and window in 95. The paired persistence gains are $43.5$ points against reset (97.5% interval $[35.5, 51.5]$; exact and adjusted $p = 3.38 \times 10^{-24}$) and $45.5$ against window ($[37.5, 53.5]$; exact $p = 8.08 \times 10^{-28}$, adjusted $p = 1.62 \times 10^{-27}$). Two primary tests use Holm adjustment. Against replay, the paired difference is zero with a 95% interval $[-3, 3]$ points, exceeding the prespecified $-10$-point non-inferiority margin.

### P.4 Four-key composite state

The fixed 128-group sample completed after interruption and resumption. Paired scene and common-random-number-prefix checks passed. The persistence-minus-reset effect is $45.31$ points (95% interval $[36.72, 53.91]$; McNemar $p = 6.94 \times 10^{-18}$), and the window contrast is $42.19$ ($[33.59, 50.78]$; $p = 1.58 \times 10^{-15}$). Against replay it is $-0.78$ ($[-3.91, 1.56]$; $p = 1$). Persistent and replay routing are correct in 128/128 cases, versus 0/128 for reset/window under the reported routing diagnostic. All arms avoid invalidated-value reuse in 128/128 cases; this measure alone therefore does not distinguish successful current-state use.

Table 8: Four-key OpenDrawer evaluation. Bytes measure mean visible memory.

| Condition | Successes | Bytes |
|---|---|---|
| Persistent | 122/128 | 149 |
| Reset | 64/128 | 0 |
| Window | 68/128 | 2,048 |
| Replay | 123/128 | 6,856 |

### P.5 Capacity and temporal-load grid

For the grid in Section 5.3, a separate false-unknown rate of .2413 has no recorded denominator and cannot be compared directly with overall accuracy. The deterministic run took 41.69 seconds on eight CPU workers; this does not measure robot or language-model throughput.