When Memory Hurts: MemoryForge for Diagnosing Memory Utility in AI Agents

Anonymous authors
Paper under double-blind review

Abstract

Agent memory is usually evaluated by what it retains or retrieves, not by whether remembered experience improves the next task. We introduce MemoryForge, the first controlled evaluation framework, to our knowledge, that branches native agent memory from a shared frozen acquisition state to isolate the downstream behavioral effects of persistence, consolidation, full-history availability, and vetted content. These effects are instantiated as five task-paired interventions: Stateless, Frozen Context, Raw Memory, Dreamed Memory, and Oracle guidance. The protocol separately measures condition-independent artifact correctness and condition-aware process compliance, then records their joint attainment. On FinStateBench, three Claude tiers crossed with three native agent runtimes yield 1,350 executions over 30 held-out tasks. Oracle achieves the highest mean artifact score (0.971) but only 45.2% joint pass, compared with 72.6% for Stateless; Raw reaches 49.3%. Moreover, 58 of the 89 Stateless passes lost under Raw also fail under Oracle, exceeding the overlap expected under independence (44). Under Raw memory, runtime accounts for 78.0% of between-system variation, whereas model tier accounts for 3.1%. The five observed branches nevertheless cover 90.4% of cells, 17.8 points above the best fixed condition, with gains in all nine systems. These results expose a Memory Utility Gap: useful behaviors exist, but current agents do not reliably select, adapt, and execute the right memory mode for each task.

1 Introduction

Language-model agents increasingly preserve experience across interactions. Memory has become a core systems capability: agents maintain episodic records, summarize interactions, reflect on failures, and distill reusable workflows or skills (Park et al., 2023; Packer et al., 2023; Shinn et al., 2023; Wang et al., 2023; Zhao et al., 2024; Wang et al., 2025). These advances rest on a compelling premise: remembered experience should improve what an agent does next.

Current evaluation does not directly test that premise. A system may recall the right fact, retrieve a relevant episode, or produce a polished memory summary while its downstream execution becomes less reliable. Once invoked, memory can redirect planning, evidence selection, parameter choices, verification, and termination. It may repair a previously failed task, but it may also overturn an execution that already succeeded. Memory therefore changes behavior far beyond whether an answer can be recalled.

This gap reflects a mismatch between memory capability and memory utility. Capability asks whether experience can be stored, retrieved, updated, or summarized. Utility asks whether that experience improves the same agent's downstream behavior. Three distinctions are central: having experience available is not the same as controlling persistent memory; high-quality memory content is not the same as applying it correctly; and a correct final artifact is not evidence that the execution remained grounded, adapted, and verified. An evaluation that collapses these distinctions cannot identify where memory helps or harms.

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

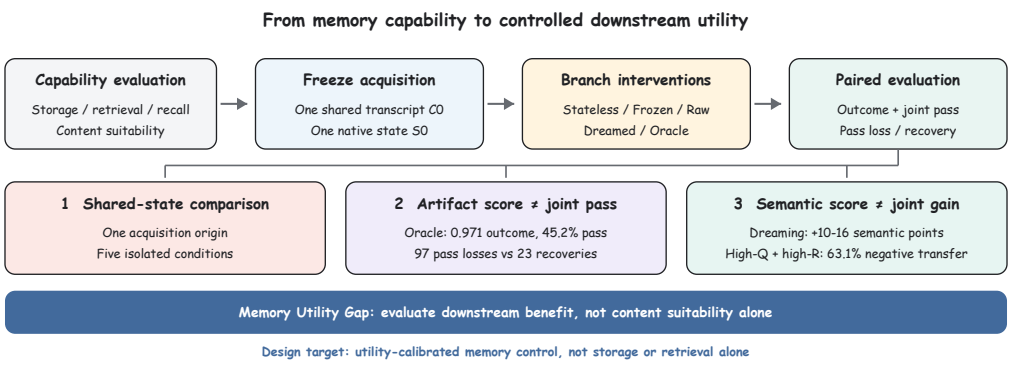

Figure 1: From memory capability to downstream utility. MemoryForge freezes a shared acquisition state, applies isolated task-paired memory interventions, and evaluates their behavioral consequences. The resulting diagnosis separates availability from utility, artifact correctness from process compliance, and semantic suitability from realized task gain.

Existing work measures complementary parts of memory capability. Long-horizon benchmarks test recall, temporal reasoning, updating, and abstention (Maharana et al., 2024; Wu et al., 2025; Tan et al., 2025). Memory-system evaluations study retrieval, reflection, consolidation, and selective forgetting (Hu et al., 2026). Agentic benchmarks move closer to behavior by testing trajectory knowledge, interdependent sessions, and the grounding of remembered constraints into actions (Zhao et al., 2026; He et al., 2026; Shen et al., 2026). Together they show whether an agent can remember and reuse experience.

However, downstream utility requires comparisons that hold both the acquisition origin and the evaluation task fixed. Existing comparisons may inherit different histories or runtime states, rarely separate raw persistence from consolidation at a shared origin, and often reduce behavior to the final artifact. They therefore cannot determine whether memory caused a repair, overturned a successful execution, or changed only the path to the same output.

We introduce MemoryForge, the first controlled evaluation framework, to our knowledge, that branches native agent memory from a shared frozen acquisition state. Its fresh, full-context, raw, consolidated, and vetted-content branches isolate the downstream behavioral effects of persistence, consolidation, full-history availability, and content sufficiency; every contrast pairs the same model–runtime system with the same task. Executable outcomes, process compliance, semantic judgments, and paired transitions reveal a Memory Utility Gap: available or semantically promising memory need not improve downstream behavior. By aligning content suitability, runtime behavior, and realized utility within the same configuration–task pair, the protocol distinguishes failures of memory content from failures of selection, adaptation, and execution.

Oracle is a vetted-content intervention, not a performance upper bound: it tests whether stronger procedural guidance alone is sufficient to improve downstream behavior. This distinction turns the comparison into a direct test of content sufficiency rather than an approximation to ideal execution.

Our contributions are threefold:

- A controlled native-state protocol. MemoryForge branches native memory from one frozen acquisition state, isolating persistence, consolidation, full-history availability, and vetted content through task-paired interventions with no cross-branch contamination.
- A benchmark of realized memory utility. FinStateBench links a ten-task acquisition curriculum to 30 held-out procedural tasks, producing a complete 1,350-execution matrix across models, runtimes, and memory interventions.
- A system-level diagnosis. Paired transitions, blinded semantic ratings, and runtime decompositions show that memory utility is governed by task-conditional selection and application, not by content quality or model scale alone.

2 Related Work

2.1 Agent Memory and Experience Reuse

Contemporary agent memory spans persistent conversation state and reusable procedural experience. Generative Agents combine a memory stream with retrieval, reflection, and planning (Park et al., 2023); MemGPT manages hierarchical memory tiers through an operating-system-inspired control loop (Packer et al., 2023); and MemoryBank maintains and updates long-term conversational memory (Zhong et al., 2024). A parallel line converts experience into future action. Reflexion retains verbal feedback in an episodic buffer (Shinn et al., 2023), Voyager accumulates executable skills (Wang et al., 2023), ExpeL extracts transferable natural-language knowledge from trajectories (Zhao et al., 2024), and Agent Workflow Memory induces reusable routines for web navigation (Wang et al., 2025). Systems such as Mem0 scale extraction, consolidation, and retrieval (Chhikara et al., 2025). Together, these systems expand what agents can preserve and reuse. MemoryForge evaluates the behavioral consequence of these capabilities inside the native systems that create, consolidate, retrieve, and apply memory.

2.2 From Memory Capability to Agentic Action

Long-horizon benchmarks first established rigorous tests of remembered information. LoCoMo evaluates question answering, event summarization, and dialogue generation over multi-session conversations (Maharana et al., 2024); LongMemEval targets extraction, temporal and multi-session reasoning, knowledge updates, and abstention (Wu et al., 2025); and MemBench distinguishes factual from reflective memory while measuring effectiveness, efficiency, and capacity (Tan et al., 2025). MemoryAgentBench evaluates incremental agents across retrieval, test-time learning, long-range understanding, and selective forgetting (Hu et al., 2026). These benchmarks characterize memory capability; MemoryForge characterizes its behavioral consequence.

Recent work closes part of the distance between recall and action. AMA-Bench constructs expert-curated and rule-based questions over real and synthetic agent trajectories (Zhao et al., 2026). Mem2ActBench tests whether agents recover latent constraints and ground them into tool parameters (Shen et al., 2026), while MemoryArena couples memory and action across interdependent sessions (He et al., 2026). AgentMemoryBench studies continual system and personal memory through online learning, transfer, replay, and repair (Ma et al., 2026). StreamMemBench separates formation, management, retrieval, and application under streaming evidence dependencies (Liu et al., 2026), and system-level analysis decomposes agent-native memory into representation, extraction, retrieval and routing, and maintenance (Zhou et al., 2026). MemoryForge contributes a shared frozen ancestor, isolated task-paired interventions, and explicit behavioral transitions. The comparison asks not simply whether memory supports action, but what behavior changes because memory is present.

2.3 Positioning MemoryForge

Mem2ActBench and MemoryArena are the closest action-oriented comparisons: the former tests whether remembered constraints ground tool calls, and the latter tests whether memory supports interdependent multi-session tasks (Shen et al., 2026; He et al., 2026). MemoryForge addresses a different identification question: what behavioral change is caused by native memory when acquisition history, model–runtime configuration, and evaluation task are held fixed? To our knowledge, it is the first controlled framework to combine native-state preservation, a shared frozen acquisition ancestor, task-paired intervention branches, and separate estimates for persistence, consolidation, full-history availability, and vetted content. Paired-transition accounting and joint outcome–process diagnosis then expose both repairs and memory-induced failures. Table 1 positions this diagnostic scope against existing memory benchmarks.

Table 1: MemoryForge evaluates the behavioral consequence of native agent memory. √ denotes explicit coverage, ○ partial or indirect coverage, and × absence from the released protocol. P/C/H/O denotes persistence, consolidation, full-history availability, and vetted content interventions.

Benchmark	Long-horizon / incremental	Downstream action / artifact	Native state	Shared frozen ancestor	P/C/H/O decomposition	Harmful transition	Outcome–process diagnosis
LoCoMo (Maharana et al., 2024)	√	×	×	×	×	×	×
LongMemEval (Wu et al., 2025)	√	×	×	×	×	×	×
MemoryAgentBench (Hu et al., 2026)	√	×	×	×	×	○	○
AMA-Bench (Zhao et al., 2026)	√	×	×	×	×	×	×
Mem2ActBench (Shen et al., 2026)	√	√	×	×	×	○	○
MemoryArena (He et al., 2026)	√	√	×	×	×	○	○
AgentMemoryBench (Ma et al., 2026)	√	√	×	×	×	○	○
StreamMemBench (Liu et al., 2026)	√	○	×	×	×	○	√
MemoryForge (ours)	√	√	√	√	√	√	√

3 MemoryForge

3.1 Evaluation target and intervention estimands

MemoryForge evaluates memory as a behavioral intervention on a complete native agent system. Let $p = (M, A)$ denote a language model M operating through a native agent runtime A, including the runtime's persistent-state representation, write policy, retrieval behavior, and native consolidation operation when available. MemoryForge preserves this system boundary and evaluates the native components end to end.

Let $\mathcal{B} = (b_1, \ldots, b_K)$ be an ordered build curriculum and $\mathcal{X} = \{x_1, \ldots, x_N\}$ a disjoint evaluation set. Executing $\mathcal{B}$ once produces trajectory τ_p and a native post-build state. The benchmark freezes both views of the acquired experience:

$$C_{0,p} = \mathrm{Freeze}(\tau_p), \qquad S_{0,p} = \mathrm{Freeze}(\mathrm{State}_A(\tau_p)), \tag{1}$$

where C_0 is the complete build-only transcript and S_0 is the runtime's post-build state, including its native memory artifacts. The main experiment uses $K = 10$ build tasks and $N = 30$ evaluation tasks.

For condition c, evaluation task x_i, and model–runtime pair p, let $Y_{p,c,i}$ denote the resulting trajectory and artifact from a newly initialized task-local branch. We evaluate Y along three deliberately separated axes: Final-Outcome Correctness $a_{p,c,i}$, Process Compliance $q_{p,c,i}$, and the Joint Execution Criterion $h_{p,c,i}$. For $S \in \{a, h\}$, write $s^S_{p,c,i} = S(Y_{p,c,i}, x_i)$. The task-paired effect of intervention c against reference c' is

$$\Delta^S_{p,i}(c, c') = s^S_{p,c,i} - s^S_{p,c',i}. \tag{2}$$

This defines two distinct evaluation targets. Task-outcome utility, Δ^a, compares the delivered artifacts using the same executable checks in every condition. Joint-criterion attainment, Δ^h, combines artifact quality with compliance with each condition's process requirements. The latter includes a setting-aware memory-use dimension and is not a condition-independent measure of answer correctness. A Joint-Pass Transition records whether the same configuration–task cell crosses that criterion, as Pass→Fail or Fail→Pass.

These task-paired contrasts estimate intervention utility for the frozen post-build states instantiated by the protocol. Repeated acquisition trajectories define a complementary acquisition-level estimand and are outside the main matrix.

3.2 Frozen-state interventions

Build tasks execute sequentially in one persistent native runtime and write memory through the runtime's ordinary interfaces. After the final build task, MemoryForge records C_0, exports S_0 with a content hash, and makes the snapshot read-only. Native Dreaming is invoked exactly once on a writable clone:

$$S_{1,p} = \mathrm{Freeze}\big(D_A\big(\mathrm{Clone}(S_{0,p})\big)\big), \tag{3}$$

where D_A is runtime A's native consolidation operation. Raw and Dreamed Memory therefore share the same acquisition ancestor; their difference is one native consolidation pass rather than additional task experience.

At evaluation time, every (c, x_i) receives a new workspace and task-local runtime state. Raw and Dreamed Memory receive writable clones of S_0 and S_1, respectively. Stateless, Frozen Context, and Oracle Memory start from empty native state and receive only their authorized evidence. Writes remain available for audit but never update a base snapshot or another task; a retry discards its partial state and restarts from the same initializer.

Figure 2 exposes the intervention structure: one acquisition origin, immutable transcript and native-state snapshots, five isolated interventions, and a task-paired behavioral readout.

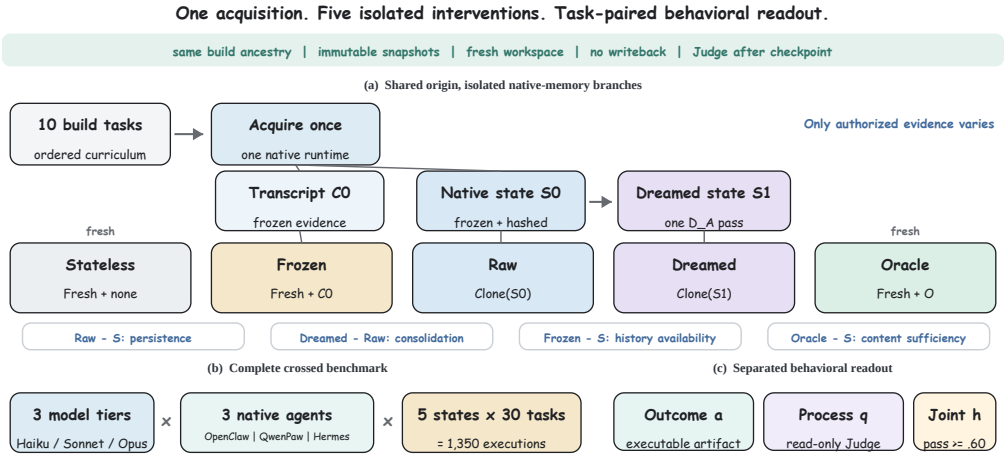

Figure 2: MemoryForge isolates the behavioral effect of native agent memory. (a) One build trajectory produces frozen transcript C_0 and native state S_0; one native consolidation pass creates S_1, after which every condition--task pair runs independently with no writeback. (b) Three model tiers, three native agents, five interventions, and 30 held-out tasks yield 1,350 executions. (c) Outcome correctness and process compliance remain separate until the joint pass criterion; task-paired transitions expose directional behavioral change.

The primary intervention contrasts follow directly. Raw versus Stateless measures the behavioral utility of native post-build persistence; Dreamed versus Raw isolates consolidation; Frozen Context versus Stateless tests whether complete experience availability substitutes for memory control; and Oracle versus Stateless tests whether stronger procedural content is sufficient. Estimating Raw and Dreamed contrasts across all nine model–runtime pairings then identifies whether memory utility belongs to a memory representation alone or emerges from the complete model–runtime system.

Stateless begins fresh; Frozen Context receives immutable C_0; Raw and Dreamed receive task-local clones of S_0 and S_1; Oracle receives the same read-only procedural memory O with no task answers. Thus each branch changes one authorized source of prior experience without changing the evaluation task or contaminating a sibling state. Oracle is an authored-content intervention: it tests the realized value of vetted guidance, rather than defining a performance upper bound. Raw evaluates the exported native post-build state, including runtime context preserved in that state, rather than a memory-text-only substitution.

3.3 Protocol invariants and executable audits

Each task independently branches from an immutable post-build snapshot. Executable audits verify shared acquisition ancestry, one-pass consolidation provenance, content hashes, clone pre-hashes, unique state paths, fresh controls, native-state fidelity, no writeback, and Judge separation. Any violation invalidates the affected execution. Appendix H specifies the full manifest and audit contract.

4 Experimental Setup

FinStateBench is a stress test for procedural memory. Its ten ordered build tasks and 30 held-out evaluation tasks require evidence retrieval, temporal alignment, parameter binding,

artifact construction, and verification while recombining financial-analysis procedures under exact, near, compositional, far-transfer, retention, and distractor conditions. All tasks have executable artifact checks; 28 additionally have process rubrics. We cross Claude Haiku 4.5, Sonnet 4.6, and Opus 4.8 with OpenClaw, QwenPaw, and Hermes, yielding nine native model–runtime systems and $3 \times 3 \times 5 \times 30 = 1{,}350$ evaluation executions. Appendices A and H give the task taxonomy and native backend implementations.

We report three complementary measures: executable Final-Outcome Correctness a, read-only judged Process Compliance q, and their rubric-weighted Joint Execution score

$$h_{p,c,i} = \frac{w_a a_{p,c,i} + w_q q_{p,c,i}}{w_a + w_q}, \tag{4}$$

with $w_a \in [0.35, 0.50]$, $w_q = 1 - w_a$, and pass threshold $h \geq 0.60$. Outcome is a mean executable-check score, not a fraction of fully correct tasks. Joint pass measures compliance with the combined artifact and process criterion; its memory-specific component assesses each condition's authorized use of prior experience. We therefore report a, q, and h separately, rather than treating joint pass as a substitute for artifact correctness. Appendix B decomposes the process dimensions that drive joint-score changes. A blinded post-execution judge separately measures intrinsic memory quality and task-conditioned relevance. All intervention effects are paired by model–runtime and task, and behavioral transitions record Pass→Fail versus Fail→Pass. Full rubrics, Judge protocols, resampling procedures, missing-data rules, and statistical tests appear in Appendices B, F, and D.

5 Main Results

5.1 Stronger Memory Does Not Guarantee Reliable Execution

A strong artifact score does not determine joint attainment. Table 2 separates these targets. Stateless, Raw, Dreamed, and Oracle have mean artifact-check scores of 0.952, 0.941, 0.952, and 0.971, respectively; their joint-pass rates are 72.6%, 49.3%, 46.3%, and 45.2%. Oracle has the highest observed artifact score but not the highest joint-pass rate. Because Oracle is a vetted-content intervention rather than an ideal-execution bound, this contrast shows that stronger procedural content alone does not guarantee reliable downstream behavior.

Paired transitions expose directional harm. Under the joint criterion, Raw converts 89 Stateless passes into failures and recovers 26; Oracle yields 97 and 23, respectively. Frozen Context reaches an artifact score of 0.753 and joint pass of 25.9%. Figure 3 displays the score profiles and transition counts together. These are complementary readouts: the artifact comparison measures task-outcome utility, while the joint comparison measures attainment under the stated process rubric.

5.2 Utility Depends on Runtime Control and Task-Conditional Selection

Expert-authored memory does not repair most Raw-induced losses. Raw converts 89 Stateless passes into failures. Fifty-eight of these same cells also fail under Oracle, so 65.2% of Raw-induced losses survive a change from native learned memory to vetted procedural guidance; 43 fail under Raw, Dreamed, and Oracle alike. Conditional on the 196 Stateless passes, independent Raw and Oracle failure sets would overlap on 44.0 cells; the observed overlap is 58 (1.32× expectation; exact $p < .001$). This concentrated shared failure set identifies a control bottleneck that content replacement alone does not remove.

Runtime choice separates memory-enabled systems more than model tier. Stateless joint-pass rates are nearly identical across Haiku, Sonnet, and Opus (72.2%, 73.3%, and 72.2%). Under Raw memory, the model-tier range is only 45.6–51.1%, whereas the runtime range expands from 37.8% for Hermes and 42.2% for OpenClaw to 67.8% for QwenPaw. The resulting 30.0-point runtime spread is 5.4 times the 5.6-point model-tier spread. Raw also has a negative mean joint-score change in every one of the nine model–runtime cells. A balanced two-way decomposition of the nine Raw pass-rate cell means assigns 78.0% of their variation to runtime, 3.1% to model tier, and 19.0% to their interaction; runtime

Table 2: Main experiment results. Panel A reports aggregate outcomes over the same 270 paired model--agent--task cells per condition; Δ and transitions use Stateless (S) or Raw (R) as indicated. Panel B reports the complete 3×3 model--agent matrix of paired joint-score effects; every cell averages the same 30 tasks.

Panel A. Aggregate intervention effects

	Scored outputs		Joint criterion		Paired behavior	
Condition	Outcome	Process	Joint	Pass (%)	Δ Pass (pp)	P→F / F→P
Stateless (baseline)	0.952	0.444	0.670	72.6	—	—
FrozenContext	0.753	0.230	0.463	25.9	−46.7 (S)	141 / 15
Raw	0.941	0.340	0.608	49.3	−23.3 (S)	89 / 26
Dreamed	0.952	0.353	0.621	46.3	— (R)	50 / 42
Oracle	0.971	0.347	0.627	45.2	−27.4 (S)	97 / 23

Panel B. Complete model–agent paired joint-score effects

Model tier	OpenClaw		QwenPaw		Hermes	
	Raw–S	Dream–R	Raw–S	Dream–R	Raw–S	Dream–R
Haiku	−0.094	+0.022	−0.025	+0.095	−0.014	+0.063
Sonnet	−0.106	−0.014	−0.025	−0.057	−0.137	+0.010
Opus	−0.112	+0.045	−0.036	−0.016	−0.006	−0.038

Outcome = condition-independent artifact correctness; Process averages the 28 rubric-bearing tasks; Pass = joint-criterion pass at $h \geq 0.60$. Colored deltas identify observed decreases, not significance. Model–agent means show variation across systems. Sensitivity analyses appear in Appendix D.

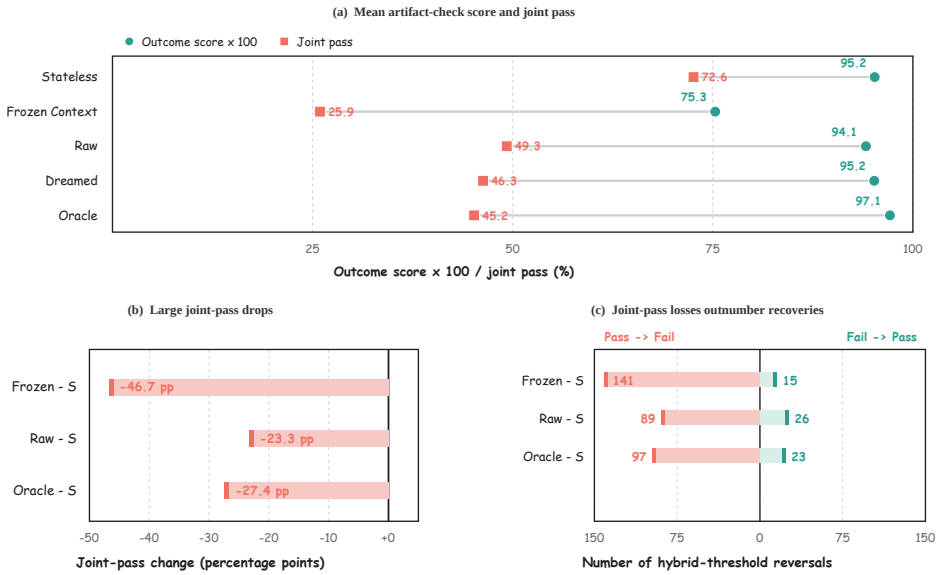

Figure 3: Memory availability does not guarantee downstream utility. (a) Mean artifact-check scores (scaled by 100) and joint-pass rates show that output correctness and joint execution reliability separate sharply. (b) Frozen Context, Raw, and Oracle reduce joint-pass rate by 46.7, 23.3, and 27.4 percentage points. (c) The same interventions show pass-to-fail transitions outnumbering recoveries for Frozen Context, Raw, and Oracle.

exceeds model tier in 99.8% of 10,000 task bootstrap samples. Memory utility therefore emerges from the complete model–runtime system rather than model scale alone.

The branches contain complementary utility that fixed policies leave unused. The best fixed condition, Stateless, passes 196 of 270 cells (72.6%). Taking the best observed union among two, three, four, and five branches covers 222, 235, 240, and 244 cells, respectively. The five-branch coverage reaches 90.4%, a 17.8-point increase, and every model–runtime system gains 1–11 additional passes beyond Stateless. The headroom remains positive in all 10,000 crossed model–runtime/task bootstrap samples. This is an observed selection upper bound

rather than a learned gate. It demonstrates that the evaluated systems already contain complementary successful behaviors and establishes prospective task-conditional routing as a measurable systems target.

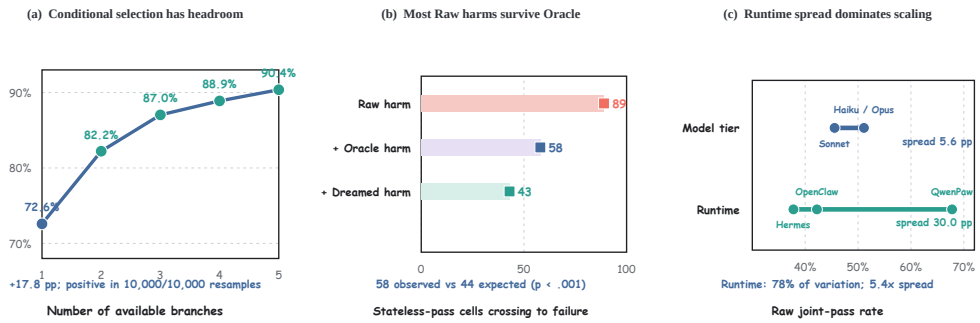

Figure 4: The Memory Utility Gap is a selection and runtime problem. (a) Best observed hindsight coverage rises from 72.6% for one fixed branch to 90.4% across all five branches and remains positive in every crossed bootstrap sample. (b) Raw--Oracle failure overlap is 58 cells versus 44 expected under independence; Oracle tests content sufficiency, not ideal execution. (c) Under Raw memory, runtime accounts for 78.0% of between-system variation and produces 5.4× the model-tier pass-rate spread.

Consolidation repairs and introduces losses. Averages combine different task-level changes. Of 89 cells that pass under Stateless but fail under Raw, Dreaming restores 33; 56 remain below threshold. At the same time, 39 cells that pass under both Stateless and Raw fall below threshold after Dreaming. Figure 5 reports all eight patterns, so repairs are evaluated alongside newly introduced losses. These transitions show that consolidation redistributes which tasks benefit, rather than uniformly repairing Raw memory.

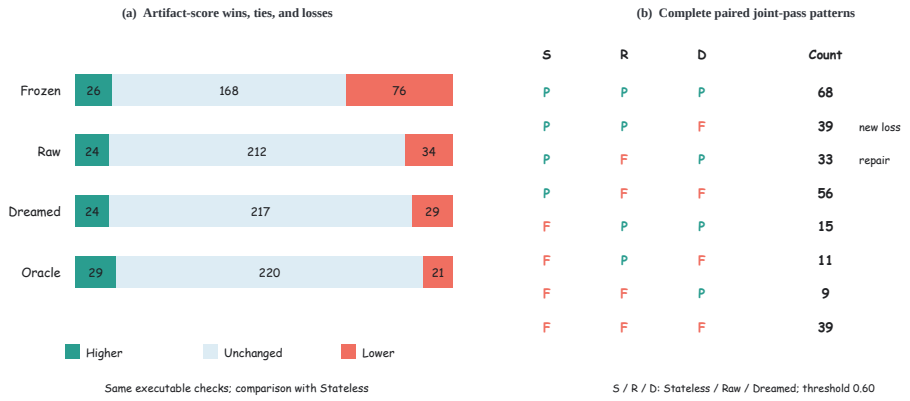

Figure 5: Separate unchanged artifacts from redistributed joint passes. (a) Artifact-check wins, ties, and losses against Stateless cover all 270 paired cells. (b) The complete Stateless--Raw--Dreamed transition table distinguishes 33 repairs from 39 new losses among previously passing cells. P/F denote joint scores above/below 0.60.

6 Diagnosing the Memory Utility Gap

Semantic suitability does not determine realized utility. We align task-level intervention effects with an independent assessment of memory content to test whether judged suitability predicts positive joint-score gain.

High semantic ratings coexist with negative joint-score gain. The offline analysis covers 295 judged memory entries (155 Raw and 140 Dreamed), 3,180 task–memory relevance judgments, and 540 task-aligned state rows under hidden labels and randomized order. Dreaming raises selected relevance, applicability, necessity, and should use by 10.2, 10.9,

16.3, and 14.1 points; all four changes are statistically resolved. Dreamed memory therefore appears more useful to an offline evaluator. This semantic advantage does not imply that the native agent selects or invokes it correctly.

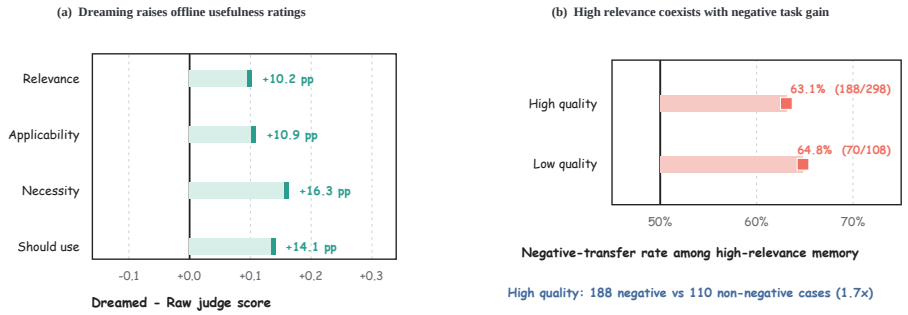

Figure 6: Semantic suitability and realized utility are distinct. (a) Dreaming raises four offline semantic ratings by 10--16 points on a 0--100 scale. (b) Negative joint-score gains occur in a majority of high-relevance task--state observations in both quality strata. Counts describe aligned observations, not independent memory artifacts.

The strongest semantic subset still transfers negatively in most cases. Among 298 high-quality, high-relevance task–state observations, 188 have negative joint-score gain and 110 have non-negative gain: a 63.1% negative-transfer rate and 1.7× count ratio. Low-quality, high-relevance observations show 64.8% negative transfer (70 of 108). High semantic ratings therefore do not certify positive realized utility. Memory selection must be evaluated against downstream compatibility as well as content suitability. Appendix D reports the full exploratory correlations, all quadrants, and artifact-type breakdowns.

7 Discussion

Memory is a control policy, not a binary capability. For the same system and task, the relevant decision is whether to remain fresh, use persisted experience, consolidate it, or invoke vetted guidance. The 30.0-point Raw spread across runtimes—versus 5.6 points across model tiers—and the corresponding 78.0% versus 3.1% variation shares show that the runtime's ability to scope, adapt, and verify recalled procedures is part of the memory mechanism itself.

Task-conditional routing converts complementarity into headroom. Stateless is the strongest fixed policy, yet the five observed branches jointly cover 90.4% of cells. The 17.8-point selection headroom defines a concrete systems target: a prospective gate should choose a memory mode before execution rather than activate memory universally. Appendix D reports the full coverage sequence and threshold sensitivity.

Semantic suitability is an input to selection, not its objective. Dreaming's higher semantic ratings capture content suitability; paired task gains capture deployed value. A utility-calibrated memory system must connect the two through three explicit control stages: gate memory before retrieval, adapt recalled procedures to current evidence, and verify the resulting action independently. MemoryForge supplies the frozen, task-paired comparisons needed to train and evaluate this Gate–Adapt–Verify policy.

8 Conclusion

MemoryForge shifts agent-memory evaluation from retention to causal behavioral utility. Shared frozen branches isolate persistence, consolidation, history, and vetted content. Across 1,350 executions, runtime dominates model tier while complementary branches raise coverage from 72.6% to 90.4%. Memory systems should optimize realized utility by selecting, adapting, and verifying experience.

AI Use Statement

Generative AI tools were used to aid and polish the writing of this manuscript and to support retrieval and discovery, including finding potentially relevant related work. The authors independently verified the cited sources, reviewed all AI-assisted text, and take responsibility for the final manuscript and its claims.

References

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. arXiv preprint arXiv:2504.19413, 2025. URL https://arxiv.org/abs/2504.19413.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland. MemoryArena: Benchmarking agent memory in interdependent multi-session agentic tasks. arXiv preprint arXiv:2602.16313, 2026. URL https://arxiv.org/abs/2602.16313.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=DT7JyQC3MR.

Guanming Liu, Haoran Yin, Tianchen Li, Sikuan Yan, Hongru Wang, Baian Chen, and Xiaoteng Ma. Streaming memory benchmark: Stage-level diagnosis with evidence dependency control. In Lifelong Agent Workshop at ICLR, 2026. URL https://openreview.net/forum?id=i1gkKNMX0K.

Zihang Ma, Jinyi Liu, Hongyao Tang, Yi Ma, Ruitao Wang, Yifu Yuan, Yan Zheng, and Jianye Hao. Benchmarking continual agent memory for online learning, transfer, and forgetting. In Lifelong Agent Workshop at ICLR, 2026. URL https://openreview.net/forum?id=MSXbrNExax.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. arXiv preprint arXiv:2310.08560, 2023. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Yiting Shen, Kun Li, Wei Zhou, and Songlin Hu. Mem2ActBench: A benchmark for evaluating long-term memory utilization in task-oriented autonomous agents. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 8173–8190. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.370. URL https://aclanthology.org/2026.acl-long.370/.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In Advances in Neural Information Processing Systems, volume 36, 2023. URL https://arxiv.org/abs/2303.11366.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents. In Findings of the Association for Computational Linguistics: ACL 2025, pp. 19336–19352. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.989. URL https://aclanthology.org/2025.findings-acl.989/.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. arXiv preprint arXiv:2305.16291, 2023. URL https://arxiv.org/abs/2305.16291.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Proceedings of the 42nd International Conference on Machine Learning, volume 267 of Proceedings of Machine Learning Research, pp. 63897–63911, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankaramapeta, Zimeng Huang, Wentao Ni, Yuandong Tian, and Jishen Zhao. AMA-Bench: Evaluating long-horizon memory for agentic applications. In International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=GoSVL7mLcM.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pp. 19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946. URL https://arxiv.org/abs/2305.10250.

Wei Zhou, Xuanhe Zhou, Shaokun Han, Hongming Xu, Guoliang Li, Zhiyu Li, Feiyu Xiong, and Fan Wu. Are we ready for an agent-native memory system? arXiv preprint arXiv:2606.24775, 2026. URL https://arxiv.org/abs/2606.24775.

A Task Inventory and Benchmark Construction

FinStateBench contains 40 fixed tasks: ten ordered build tasks and 30 held-out evaluation tasks. Every task specifies an instruction, required artifact, hidden reference answer, executable checks, and memory annotation. The build curriculum covers aligned indicator computation, common-date return correlation, volatility ranking, consecutive-window state machines, temporal coincidence, volatility compression, prior-window breakout confirmation, duration-based indicators, price–volume alignment, and MACD-transition analysis. Evaluation tasks recombine these procedures across security universes, date windows, technical indicators, portfolio statistics, sector relations, and cross-instrument verification.

Twenty-six evaluation tasks require at least one build procedure. Four selective-use controls expose plausible but inappropriate memory, and nine tasks carry explicit negative-transfer risk. Required-memory annotations are compositional: nine tasks require one procedure, 14 require two, two require three, and one requires four.

Table 3: Composition of the FinStateBench evaluation split.

Transfer type	Tasks	Transfer type	Tasks
Exact	1	Near	5
Compositional	10	Far	5
Long-horizon retention	5	Distractor retrieval	2
Negative transfer	2	Total	30

B Evaluation Dimensions and Rubrics

Final outcomes. Executable graders inspect required files and intermediate artifacts through named checks for entities, dates, formulas, numerical tolerances, universe coverage, and output completeness. The mean named-check score defines $a_{p,c,i} \in [0, 1]$; grader exceptions receive zero and remain in the record.

Process compliance. For the 28 rubric-bearing tasks, a read-only process judge receives the task, hidden reference, candidate output, saved trajectory, tool evidence, and workspace artifacts. For five-dimension records, the functional is

$$q_{p,c,i} = 0.20q_{\text{plan}} + 0.25q_{\text{data}} + 0.25q_{\text{verify}} + 0.20q_{\text{memory}} + 0.10q_{\text{closure}}, \tag{5}$$

covering planning, data/tool evidence, reasoning and calculation verification, memory transfer and boundary control, and execution closure. The two tasks without process rubrics use the executable artifact score in the joint criterion.

The memory criterion is condition-specific: Stateless rewards independent solution and respect for the absence of prior memory; Raw and Dreamed require visible evidence of retrieval and appropriate application. Frozen Context and Oracle are assessed on their authorized evidence. Its between-condition differences measure rubric compliance, not a shared retrieval test.

Semantic diagnosis. A separate blinded judge scores Raw and Dreamed memories with identical prompts. Intrinsic dimensions are correctness, completeness, abstraction, scope accuracy, actionability, redundancy, and transferability; task-conditioned dimensions are relevance, applicability, necessity, conflict, transferability, and should-use. Model, harness, state label, intervention, and downstream score are hidden.

Observed process-dimension decomposition. The largest process-dimension gap concerns the use of prior experience. On dimension-complete paired traces, the memory-use and boundary score changes from 75.4 to 7.4 points under Frozen Context, from 75.8 to 22.5 under Raw, from 76.8 to 22.5 under Dreamed, and from 76.7 to 21.0 under Oracle. Under Raw, every other process dimension changes by at most 5.0 points. This decomposition identifies which part of the joint criterion contributes most strongly to the observed gap.

The memory dimension is setting-aware: Stateless is assessed on independent solution and boundary compliance, whereas memory conditions require evidence of appropriate use of the supplied experience. These scores characterize condition-specific process compliance, rather than a common retrieval-accuracy test. Keeping this diagnostic separate from executable artifact checks makes the evaluation target explicit.

The joint-pass gap also appears across transfer groups. Raw joint-pass losses range from 13.3 points on long-horizon retention to 33.3 points on far transfer, while Frozen Context loses 37.8–60.0 points across the same groups. Appendix D reports the complete task-family breakdown.

C Full Model–Harness Results

The complete 3×3 model–harness matrix appears in Panel B of Table 2; Figure 4 summarizes its system-level spread and selection headroom. A balanced two-way decomposition of the

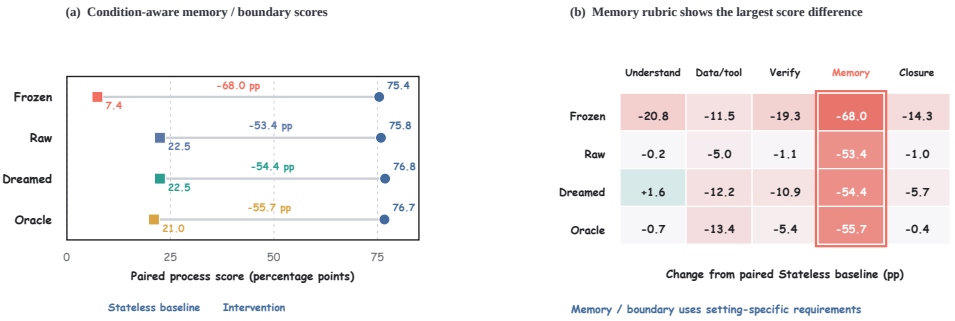

Figure 7: Decomposing the condition-aware process criterion. (a) Memory-use and boundary scores under the setting-specific rubric; each segment compares only executions with dimension-complete traces in both the intervention and its Stateless control (Frozen $n = 171$, Raw $n = 144$, Dreamed $n = 128$, Oracle $n = 137$). (b) The complete paired delta matrix identifies the largest process-dimension gaps. The memory dimension uses different authorized-experience requirements across conditions.

nine Raw pass-rate cell means attributes 78.0% of between-cell sum of squares to runtime, 3.1% to model tier, and 19.0% to their interaction. In 10,000 task-bootstrap samples, the runtime share exceeds the model-tier share in 99.8% of samples.

D Workflow- and Task-Level Analysis

Paired summaries. Every contrast contains 270 differences paired by model–harness and task. Crossed-bootstrap replicates independently resample the nine systems and 30 tasks, then average the resulting 9×30 crossed submatrix. Continuous effects use percentile intervals; binary outcomes use discordant Pass→Fail and Fail→Pass counts with supplementary McNemar summaries. Primary contrasts are Raw–Stateless, Dreamed–Raw, Frozen Context–Stateless, and Oracle–Stateless.

Table 4: Joint-pass sensitivity by transfer structure. Values are descriptive pass rates aggregated over nine model--agent systems per task; parenthesized deltas use Stateless as the reference.

Transfer structure	Tasks	Stateless	Raw	Frozen Context
Compositional	10	67.8	46.7 (−21.1)	25.6 (−42.2)
Near	5	77.8	55.6 (−22.2)	17.8 (−60.0)
Far	5	71.1	37.8 (−33.3)	22.2 (−48.9)
Long-horizon retention	5	68.9	55.6 (−13.3)	31.1 (−37.8)
Distractor retrieval	2	77.8	50.0 (−27.8)	44.4 (−33.3)
Negative transfer	2	94.4	72.2 (−22.2)	33.3 (−61.1)
Exact	1	66.7	22.2 (−44.4)	11.1 (−55.6)

For semantic analysis, task gain is $h_{p,c,i} - h_{p,\text{Stateless},i}$ and Dreaming gain is $h_{p,\text{Dreamed},i} - h_{p,\text{Raw},i}$. Median-split quality–relevance quadrants report mean gain, negative-transfer rate, and Fresh→Memory reversal rate. Pearson correlations and ordinary least squares quantify association with gain. Missing artifacts, judge calls, or score alignments are excluded rather than imputed; because semantic judgments follow execution, these regressions are associative rather than interventional.

Exploratory Pearson correlations with task gain are 0.032 for quality, 0.028 for relevance, and 0.016 for their product; correlations with Dreaming gain are −0.058, −0.038, and −0.101, respectively. Negative-transfer rates are 65.6% for procedural skills, 63.3% for individual entries, 62.2% for session entries, and 60.0% for multi-session artifacts. The high-quality/high-relevance quadrant contains 298 rows, with 63.1% negative transfer (188 negative versus 110 non-negative observations) and 35.2% Fresh→Memory reversals. Low-quality, high-relevance observations have 64.8% negative transfer. These are descriptive rates over task–state rows that share tasks, systems, and memory artifacts; the rows are not

independent Bernoulli trials. Negative transfer means any joint-score decrease, whereas a reversal requires crossing the joint-pass threshold.

The designated transfer-risk tasks expose selective invocation directly. Across the nine model–harness systems, selective-use accuracy is 40.7% and harmful-transfer rate is 59.3%, ranging from 33.3% to 88.9%. Exact and compositional reuse align most strongly with positive Dreaming effects, whereas distractor rejection aligns with negative effects.

Threshold sensitivity and hindsight coverage. Varying the threshold without rescoring preserves the largest pass count for Stateless throughout Table 5.

Table 5: Joint-pass counts out of 270.

Threshold	Stateless	Frozen	Raw	Dreamed	Oracle
.50	234	137	198	201	199
.55	216	106	165	160	160
.60	196	70	133	125	122
.65	153	52	95	93	94
.70	113	47	70	69	76

At .60, the best observed union among one, two, three, four, and five branches covers 196, 222, 235, 240, and 244 cells, respectively. The best subsets are S; S+R; S+R+O; S+F+R+O; and all five branches. Every cell is counted once if any included branch passes. Across individual model–runtime systems, the five-branch union adds 1–11 passes beyond Stateless, with a mean gain of 5.3 out of 30 tasks. In 10,000 crossed bootstrap samples that independently resample the nine systems and 30 tasks, five-branch union coverage exceeds the best fixed branch in every sample. These quantities define stable selection headroom for a prospective task-conditional gate.

Shared Raw–Oracle failure set. Among the 196 cells that pass under Stateless, Raw and Oracle produce 89 and 97 pass losses. Their observed intersection contains 58 cells, compared with 44.0 expected if the two failure sets were independent within this fixed Stateless-pass universe. An exact hypergeometric upper-tail test gives $p = 5.13 \times 10^{-5}$; 43 cells additionally fail under Dreamed memory. The concentration confirms a recurring task-paired failure set across memory content interventions.

Common-process dimensions. For dimension-complete pairs, removing the memory dimension and renormalizing the remaining weights by .8 gives mean Stateless/condition scores: Frozen .363/.196 ($n = 171$), Raw .377/.356 ($n = 144$), Dreamed .392/.317 ($n = 128$), and Oracle .387/.326 ($n = 137$). These descriptive subsets hold included dimensions fixed but differ in membership; they do not replace full-matrix outcomes.

E Action-Level Failure Analysis

The automated taxonomy provides an audit index into saved runs. Stateless has 194 records labeled `none` and 9 labeled `answer_format_failure`; corresponding counts are 76/103 for Raw, 58/176 for Dreamed, and 35/210 for Oracle. The latter label is assigned when the saved Judge object's `actual` field is null. It flags a missing extracted-output field, not a separately verified execution-closure failure. Likewise, `none` means no primary label was assigned, not that the entire trajectory was independently certified correct. These counts are retained for record-level auditing, rather than used as evidence of a behavioral failure mechanism.

The framework organizes further trajectory inspection around the control loop

$$\text{Acquire} \rightarrow \text{Consolidate} \rightarrow \text{Retrieve} \rightarrow \text{Gate} \rightarrow \text{Adapt} \rightarrow \text{Execute} \rightarrow \text{Verify}.$$

Lifecycle fields are reported where the corresponding harness exposes them; end-to-end intervention effects remain available for every condition.

F Evaluator Protocol and Auditability

MemoryForge separates the executable outcome grader, the process judge, and the semantic memory judge so that no evaluator can write into agent state or affect a saved execution. Table 6 summarizes the operational safeguards for each evaluator. These safeguards establish isolation and repeatable scoring; agreement with independent human judgments is a separate evaluator-validity question.

Table 6: Evaluator safeguards and failure handling.

Evaluator	Operational guard	Failure handling
Outcome grader	Deterministic named checks, fixed schemas, and numerical tolerances	Log and score zero
Process judge	Read-only trajectory bundle, fixed rubrics, temperature zero, serialized calls, and execution-first checkpoints	Retry cached call
Semantic judge	Identical Raw/Dreamed prompts, hidden system and condition labels, randomized order, and resumable cache	Exclude malformed row

The semantic analysis is intentionally post-execution: it tests whether quality and relevance explain observed gains without exposing downstream scores to the judge. Process judgments likewise reuse immutable saved trajectories. This separation permits rejudging and evaluator audits without changing any model–harness behavior.

G Trajectory Cases

We inspect three QwenPaw task-paired cases with five-dimension process records in all three branches. They illustrate a repaired joint pass, a newly lost joint pass, and visible method reuse; they are not estimates of mechanism frequency. Each table entry is artifact score / joint score.

Table 7: Verified cases: S, R, D denote Stateless, Raw, Dreamed.

Model	Task	S	R	D
Opus	36: intraday anomaly	1/.750	.6/.559	1/.719
Opus	33: sector leadership	1/.639	1/.630	.5/.460
Sonnet	29: momentum reversal	.5/.451	1/.765	1/.658

Case 1: score recovery versus procedural fidelity. For task 36, Raw's output explicitly substitutes daily OHLC data for intraday anomalies. Dreamed's `detect_intraday_anomaly.py` calls `stock_zh_a_hist_min_em` with `period='30'`. Dreamed recovers a joint pass, but its version-2 detector uses `volume_ratio > 1.8`, rather than the five-times volume requirement. The final artifact reports no qualifying signals. An attempted granularity correction and a changed screening threshold coexist: the recovered score does not by itself certify faithful execution.

Case 2: a newly lost pass exposes a grounding distinction. For task 33, Stateless outputs placeholder identifiers `300BCD` and `300NOP`; saved code generates values with `np.random.uniform`. Raw reports no qualifying stocks. Both receive artifact score 1. Dreamed reports data-retrieval failure and receives .5. This is a newly lost joint pass, but the visible evidence does not establish the earlier branches as verified market-analysis successes. Schema/content checks and validated data provenance are separate readouts.

Case 3: procedural reuse without verified market data. For task 29, Raw's transcript explicitly introduces procedures based on memory and refers to a multi-indicator stock-screening memory file. Its solution implements recursive Wilder smoothing, `(avg_gain *`

`(period - 1) + gains[i]) / period`. The same solution seeds a random generator and synthesizes price changes. Artifact and joint scores reach 1 and .765. This supports an observation of procedural reuse, not successful screening of real market data. Memory application and source validation must be audited separately.

Evidence access. The accompanying case-evidence manifest records exact source paths, file hashes, scores, and excerpts from the original runs. All cases use QwenPaw; tasks 36 and 33 use `claude-opus-4-8`, and task 29 uses `claude-sonnet-4-6`. Code present in a workspace is distinguished from execution proven by a trace. None of these examples attributes a failure causally to a particular memory entry.

H Reproducibility Details

Evidence map. The analysis joins task records by model, agent, setting, and task identifier. The task-level summary retains executable scores, process dimensions, joint scores, and the corresponding run path. Semantic alignment uses the separate quality–relevance–gain summary. The figure generator in `iclr2027/scripts/make_result_figures.py` reads these summaries; reported semantic row counts refer to aligned task–state observations. The runtime decomposition, exact failure-overlap test, and crossed-bootstrap selection analysis are reproduced by `iclr2027/scripts/analyze_selection_system.py`, which reads the same immutable task-level summary and writes a machine-readable statistics record. The executable process prompt is defined by `_process_judge_prompt` in `memoryforge/judge.py`; its criterion scores are normalized in `memoryforge/process_judge.py`. Saved grading records retain the Judge protocol and task-specific weights. These records connect aggregate figures to their scored executions without requiring a new Agent run.

OpenClaw, QwenPaw, and Hermes expose distinct persistence and consolidation paths. Adapters invoke these native paths without imposing a shared memory schema, retriever, or summarizer.

Table 8: Native persistence and one-pass consolidation paths.

Runtime	Persistent native artifact	One-pass Dreaming operation
OpenClaw	Workspace Markdown memory cards and index	Grounded REM backfill followed by native promotion
QwenPaw	ReMe daily memory cards	Native digest construction and indexing via the Dreaming command
Hermes	Agent-authored reusable Markdown skills	One synchronous native Curator consolidation pass

A preflight canary verifies native write, fresh-session recall from a cloned snapshot, one-pass Dreaming, and consolidated-state recall. Each task runs in a backend-specific Docker image with four CPUs, 8 GB memory, task-local workspace and state, read-only benchmark and source mounts, and credentials excluded from saved artifacts.

Executable invariants. MemoryForge groups its audits by four isolation guarantees. Acquisition: one build produces shared C_0 and S_0, and the one-pass S_1 manifest links to its immutable source. Snapshots: content hashes and clone pre-hashes must match before and after evaluation. Isolation: every condition–task pair has a unique workspace, receives only authorized evidence, and cannot write into a base or sibling. Native/scoring fidelity: adapters export and validate each native state tree, while agent execution is checkpointed before judging so that rejudging reuses the saved run.

Per-run manifests record build and Dreaming counts, snapshot hashes, clone pre-hashes, unique state paths, and uncontrollable remote state. A stale cache, hash mismatch, malformed native artifact, or shared state path invalidates the affected execution.

FinStateBench targets procedural transfer in auditable, stateful financial-analysis workflows. The primary estimand is task-paired utility over frozen post-build states across three Claude

capability tiers and three native runtimes. Repeated acquisition trajectories, online updating, learned gating, broader model families, and additional agentic domains define direct extensions. Historical analysis tasks serve as a controlled scientific testbed rather than a financial deployment protocol.