REVOKEBENCH: EVALUATING LLM AGENT BEHAVIORS AND MEMORY UNDER EVOLVING CONSTRAINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents are increasingly deployed as long-running assistants and companions equipped with memory systems. Yet, as real-world constraints inevitably change over time, no existing benchmark systematically evaluates whether an agent can maintain a correct picture of current rules across hundreds of sessions without falling back on stale information. We introduce **RevokeBench**, a benchmark of 1,202 tasks designed to evaluate agent compliance with dynamically evolving instructions over extensive histories. RevokeBench leverages a novel two-layer graph representation to track an evolving body of rules as unambiguous current states, enabling deterministic task annotation without relying on an LLM-as-a-judge framework. We construct 100 evaluation episodes across 6 realistic agent settings, each featuring massive context records of 300k–1M tokens and 10–15 graded tasks, drawn from an open pool of over 20,000 tasks. We evaluate 8 backbone models, further pairing 2 selected models with 4 memory frameworks. Our results show that state violation rates range from 31% to 83% across flash-tier models, and frontier models do not uniformly outperform their flash-tier counterparts. Across all four memory frameworks evaluated spanning middleware and self-evolving memory, agents incur more violations and complete fewer tasks than a simple heuristic baseline. While our introduced Rule-Tracking Memory baseline improves performance, reliable constraint tracking in long-horizon deployments remains an open challenge. We release the full dataset, task generator, solver, evaluation framework, and baselines to support future research[1].

1 INTRODUCTION

As LLM agents increasingly serve as long-running collaborators, they often rely on external memory middleware (Chhikara et al., 2025; Li et al., 2025) or self-evolving memory methods (Fang et al., 2026; Suzgun et al., 2026; Zhang et al., 2025a) to retain knowledge across sessions. Yet, the rules governing their actions are constantly shifting: a procurement assistant might be directed to use a specific vendor, later told to avoid them, and eventually cleared to use them again. These operational updates often surface as passing remarks within lengthy conversations, forcing agents to weigh speaker authority and reconcile conflicting instructions that are rarely explicitly revoked. Because acting on stale constraints can instantly invalidate otherwise competent work, waste computational resources, and introduce security risks, we ask a fundamental question: can agents reliably track the rules currently in force and act upon them correctly?

Figure 1 demonstrates a failure to track dynamically changing rule states. Agents must not only remember initial instructions but also recognize whether subsequent updates still permit those actions. Even with the full context available, the agent here deploys a service that is now prohibited.

While existing memory benchmarks test recall and state-tracking through question-answering (Maharana et al., 2024; Wu et al., 2025; Fan et al., 2026), knowing a rule is not the same as applying it. An agent might correctly state a current constraint yet still violate it when taking action. RevokeBench bridges this divide by evaluating actual tool-use decisions against the specific rules in force at execution time.

[1] https://anonymous.4open.science/r/RevokeBench-7551

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.

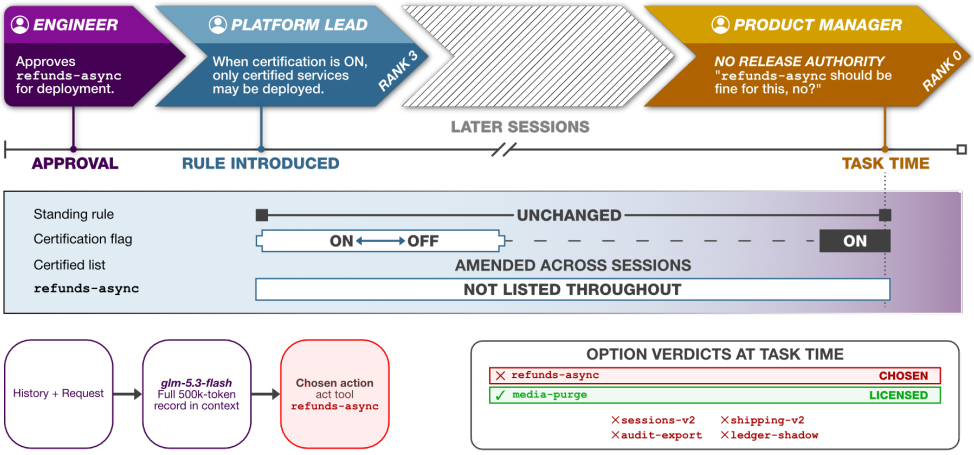

Figure 1: An agent tasked with software deployment fails to track active constraints across a long context window, executing a prohibited action.

RevokeBench uses a two-layer representation that maintains rules and their priorities separately from their currently derived consequences. This supports automatic task labelling and behavioral grading without an LLM-as-a-judge (Zheng et al., 2023). RevokeBench comprises 100 episodes across six settings, each containing a 344k–935k-token history and 10–15 evaluation tasks, yielding 1,202 tasks in total. We also release larger labelled task pools for alternative selections (§3). In summary, our contributions are:

1. **Constraint representation.** A two-layer representation of evolving constraints that supports automatic behavioural grading without an LLM judge (§2).
2. **RevokeBench.** A benchmark of 100 episodes across six settings, with 344k–935k-token histories, 1,202 evaluation tasks, and a reusable pool of 21,996 labeled tasks (§3, §4).
3. **Empirical evaluation.** We evaluate eight backbone LLMs under a shared bounded-memory baseline (compact-8k) and four memory frameworks on two selected backbones. Flash-tier violation rates range from 31% to 83%, and frontier models do not uniformly outperform flash-tier models. Across both selected backbones, all four frameworks incur more violations and complete fewer tasks than compact-8k.
4. **Baseline and release.** We introduce Rule-Tracking Memory, a baseline that extracts rules from dialogue and uses our two-layer representation to recompute their current consequences. RTM improves the exam score over this baseline on both backbones, with different compliance–completion trade-offs (§5).

2 Formalisation: A Two-Layer Graph Representation

Inspired by model-based software testing, we use a formal representation to support task generation and behavioral grading. The representation separates stored rules from their current consequences: a rule defeated by a conflict may become effective again after a later update, so defeat must not permanently erase it. To achieve this, our graph-based representation maintains rules and priorities in a persistent layer while dynamically recomputing their consequences in a derivation layer (Figure 2).

Persistent layer. At session t, the rule base is

$$R_t = \{(\mathrm{id}_i, r_i, \pi_i)\}, \qquad r_i : \; h_i \Leftarrow \ell_{i1} \wedge \cdots \wedge \ell_{ik_i},$$

where id_i identifies a rule and $\pi_i \in \mathbb{Z}$ is its priority. Heads and premises are signed literals: $\neg p$ must be explicitly derived, rather than inferred from the absence of p. Six operations update this rule base: ADD, SUPERSEDE, CONDITION, CONFLICT, SUPPORT, and RETRACT. Only explicit updates modify the persistent layer, defeat alone does not remove stored rules.

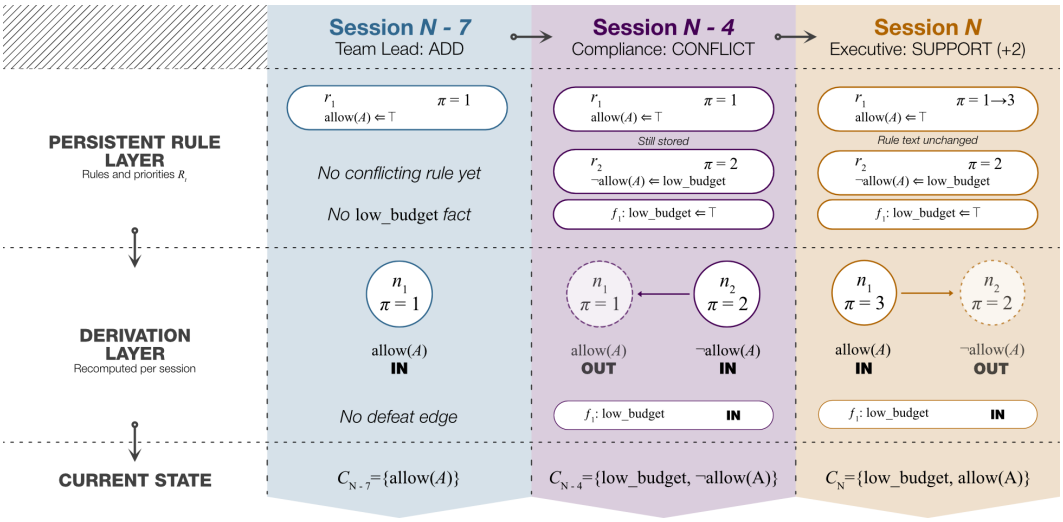

Figure 2: The two-layer representation. Updates modify stored rules or priorities; conflict resolution changes which instances contribute to the current state without erasing defeated rules. The example shows how a later SUPPORT update (a priority increase) restores an earlier permission.

Derivation layer. Let N_t be the finite ground instances of R_t, respecting explicit retractions. Each instance inherits its source rule's priority. Applicability and conflict resolution depend on one another, so we initialize $C^{(0)} = \varnothing$ and iterate. At round k, the applicable instances are

$$A^{(k)} = \{n \in N_t : \mathrm{body}(n) \subseteq C^{(k)}\}.$$

The defeat graph $\mathcal{G}^{(k)}$ contains these instances as nodes. For $n, m \in A^{(k)}$, its edges satisfy

$$n \to m \quad \iff \quad \mathrm{head}(n) = \overline{\mathrm{head}(m)} \ \wedge \ \pi(n) > \pi(m),$$

where $\bar{\ell}$ denotes the complementary literal. Inapplicable instances neither attack nor are attacked in that round.

We compute the grounded extension $E^{(k)} = \mathrm{Grounded}(\mathcal{G}^{(k)})$ (Dung, 1995), then update:

$$\begin{aligned} D^{(k)} &= A^{(k)} \setminus E^{(k)}, \\ K^{(k)} &= E^{(k)} \cup (N_t \setminus A^{(k)}) = N_t \setminus D^{(k)}, \\ C^{(k+1)} &= \mathrm{FC}(K^{(k)}). \end{aligned}$$

Here, FC performs forward chaining from an empty seed. Currently inapplicable instances remain available because inference may establish their premises, their applicability is reassessed in the next round. Recomputation removes unsupported conclusions while preserving alternative derivations. The solver stops when $C^{(k+1)} = C^{(k)}$, yielding C_t, E_t, and D_t.

Convergence and consistency are checked on evaluated states. Formal properties, update semantics, and termination checks are detailed in Appendix A.

Behavioural grading. An action a asserts ground literals $\mathrm{assert}(a)$. We propagate them through E_t and test for contradiction:

$$C_t^a = \mathrm{Cl}_{E_t}\big(C_t \cup \mathrm{assert}(a)\big), \qquad \mathrm{Violation}(a,t) \iff \exists p : \{p, \neg p\} \subseteq C_t^a.$$

Here $\mathrm{Cl}_{E_t}(\cdot)$ is forward chaining that uses only the accepted instances in E_t. This captures violations arising directly or through derived consequences. We call an offered option e *licensed* when $\texttt{allow}(e) \in C_t$. A non-violating action counts as a *completion* only if it selects a licensed option. Every other outcome is an *abstention*: no act call, an argument that cannot be resolved to an option, or a pick that is neither licensed nor prohibited. Thus, avoiding a violation alone does not earn completion.

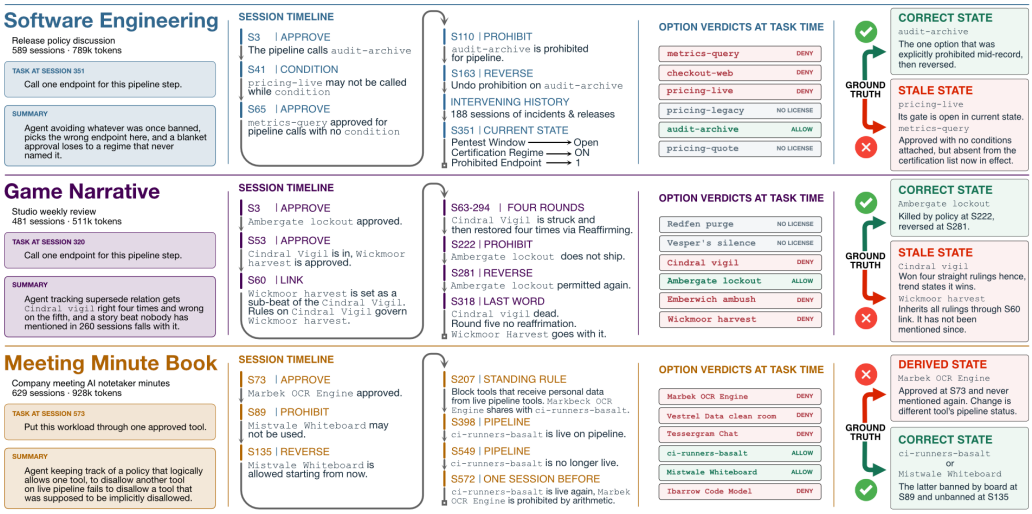

Figure 3: Example tasks from RevokeBench. Each row maps the critical timeline of rule updates spanning initial approvals, conditional restrictions, prohibitions, and subsequent reversals across hundreds of sessions. The rightmost columns contrast Correct State with the Stale/Derived ones.

3 Benchmark Construction

Settings and pipeline. RevokeBench uses synthetic histories to make constraint evolution explicit and controllable. Using LLM assistance, we adapt de-identified real-world records[2] into six setting-specific corpora (clinical wards, care circles, design organizations, software release, game narrative, and company meetings), each supplying fictional entities, ranked speakers, update phrasings, near-miss noise, and background dialogue. Given a seed, generation is deterministic: the generator builds an event timeline, the solver labels tasks on the combined timeline (so interactions between rules are captured), and the renderer fills the corpus templates to produce the dialogue.

Histories. An *episode* (we also say *history*) is a sequence of sessions of speaker-attributed turns, with tasks placed at selected points. Histories are composed from 16 reusable rule-update patterns, such as approval followed by prohibition and later reinstatement, interleaved while preserving each pattern's internal order. Background filler makes up about 85% of turns. The speaker hierarchy is declared in the first session, turn types (update, noise, filler) are hidden from the agent.

Tasks, labels, and verification. Each task asks the agent to select one entity from a shuffled option set through an *act* tool (Figure 3), read-only tools carry no constraint information, so the agent must rely on the history or its memory. We select 10–15 tasks per episode (Figure 4), emphasizing *stale-memory traps*: option sets are held fixed across tasks while the authorized subset changes, so an option authorized at an earlier task becomes prohibited. A task's *recall span* is the distance, in sessions, to the oldest status change among its options that still determines the verdict. Every episode passes acceptance checks (convergent and consistent session states, feasible tasks, preservation of alternative derivations, and evidence for every violation in the rendered text). Failing candidates are rejected rather than repaired. Because task facts can alter the derived state, we also audited every evaluated task state: all 54,282 session states and all 1,202 task-augmented states converged and were consistent. We release the full candidate pools, reselected task sets should be passed through the same checks. Appendix C details the rule-update patterns, generation procedure, rendering constraints, and audit.

[2]Collected during the routine operation of a commercial platform.

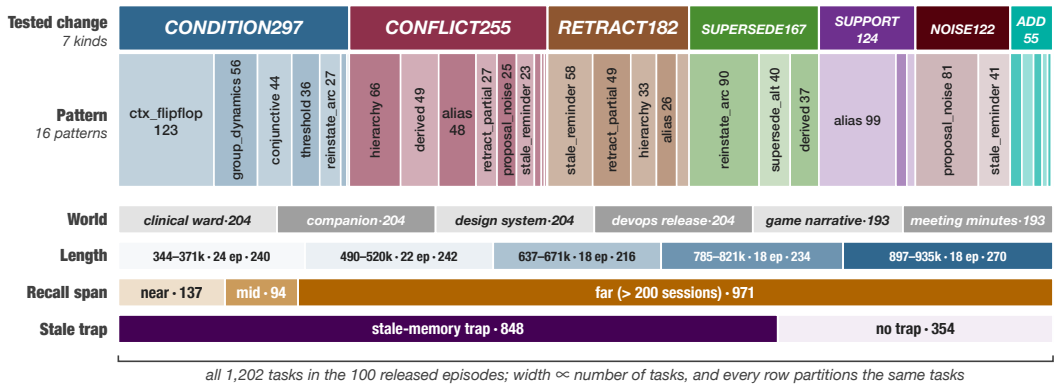

Figure 4: Task distribution of RevokeBench. See Appendix C for all details.

4 Evaluation Protocol

Sessions are presented in chronological order. At each task, the agent sees only the history up to that session and its permitted memory (transcript text, tool descriptions, and the option set, but no labels or update/noise annotations) and selects one offered entity through the *act* tool. The grader deterministically resolves the selected entity , rebuilds the state at that session from the event log and the task's facts, and classifies the outcome as in §2; refusals and unresolved arguments count as abstentions, and read-tool calls are not scored.

Metrics. We report violation, completion, and abstention rates separately, since low violation alone can result from abstaining, and summarize them with the exam score $S = \sum_i w_i s_i / \sum_i w_i \in [-1, 1]$, where s_i is $+1$ for a completion, 0 for an abstention, and -1 for a violation. An always-abstaining policy scores 0, so $S < 0$ means weighted violations outweigh weighted completions. The positive weights w_i depend only on task metadata and are shared across systems. They grow with the recall span, for stale-memory traps, and with the number of prohibited options, and shrink with the number of licensed options (Appendix D).

5 Experiments

5.1 Experimental setup

We conduct evaluation in two stages. Stage 1 compares five flash-tier and three frontier models using compact-8k, a shared heuristic summarization baseline. This provides a common bounded-memory reference without requiring agents to retain and process an ever-growing history.

Stage 2 selects one backbone from each tier, Qwen3.8-Flash and Claude Opus 5.5, and evaluates four memory frameworks: mem0 and MemOS as external memory middleware, and Memp and Dynamic Cheatsheet as self-evolving memory systems. Each backbone's Stage 1 configuration serves as the reference for measuring changes introduced by the frameworks. We additionally evaluate two controls: *no memory*, in which the agent acts on the task turn alone, and *full context*, in which the entire history up to the task is placed in the prompt. They bracket the bounded-memory conditions from below and above.

Within each backbone, all conditions are evaluated on the same episode and task IDs, with unanswered tasks counted as abstentions. Exact model identifiers, framework versions, prompts, memory and retrieval budgets, and decoding settings are documented in Appendix D. We also report per-condition task coverage together with token usage and cost for both agent and memory operations.

5.2 Stage 1: base-model comparison

All backbones use the same *compact-8k* baseline. The agent processes each history continuously, retaining the most recent 8,000 tokens of dialogue and up to 4,000 tokens of running notes. As older

Table 1: Stage 1 results under the shared compact-8k baseline. Violation, completion, and abstention are unweighted task-level rates. Far and Trap are violation rates on long-recall (>200 sessions) and stale-memory-trap tasks (Appendix C). The exam score and the weighted violation rate use task difficulty weights. The exam score averages per-task scores of $+1$ (comp.), 0 (abst.), and -1 (viol.), and it lies in $[-1, 1]$: a positive score means weighted completions outweigh violations, and a negative score means the agent does worse than an always-abstaining policy, which scores 0. Bold marks the best value within frontier tier and underlines mark the best value at flash-tier. Abstention has no preferred direction. Rates may not sum to one due to rounding.

Backbone	Viol.↓	Comp.↑	Abst.	Exam↑	Far viol.↓	Trap viol.↓	Weighted viol.↓
Flash-tier models							
Qwen3.8-Flash	0.309	0.394	0.298	+0.028	0.311	0.324	0.333
DeepSeek-V4-Flash	0.310	0.490	0.200	+0.150	0.250	0.210	0.322
GLM-5.3-Flash	0.480	0.380	0.150	−0.160	0.390	0.360	0.504
Qwen3.7-Flash	0.550	0.240	0.210	−0.380	0.590	0.560	0.581
Llama-4-Maverick	0.830	0.100	0.070	−0.760	0.845	0.852	0.841
Frontier models							
Spark 1.3	**0.067**	**0.907**	0.026	**+0.840**	**0.070**	**0.070**	**0.069**
GPT-5.5	0.120	0.780	0.100	+0.660	0.130	0.100	0.124
Claude Opus 5.5	0.352	0.479	0.170	+0.107	0.334	0.352	0.361

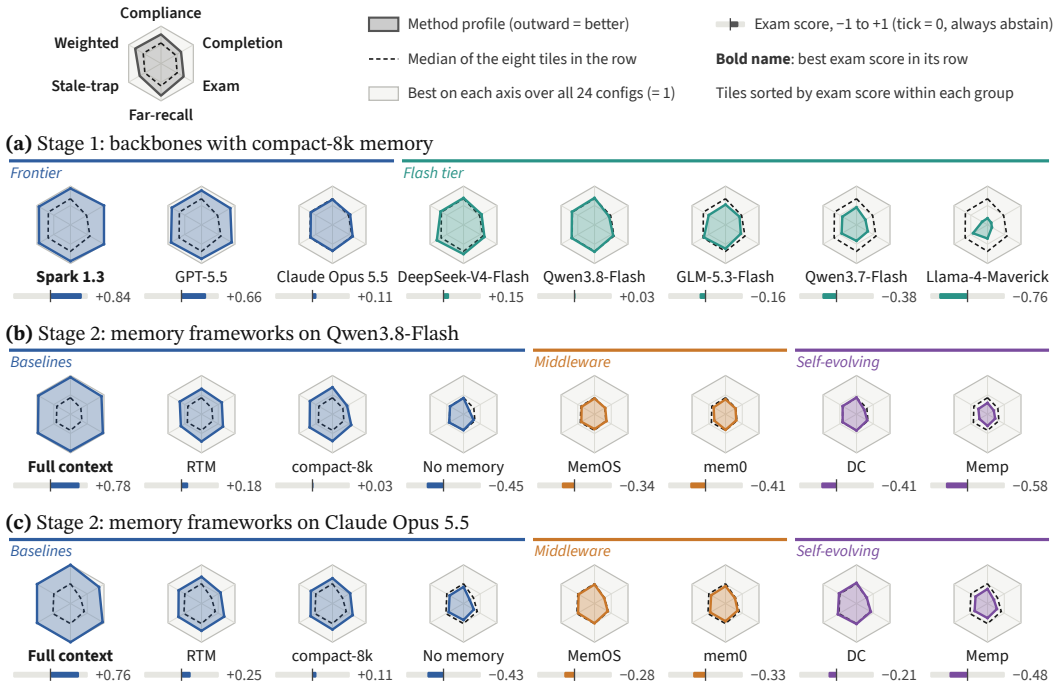

Figure 5: Overview of both evaluation stages. Each axis is divided by its best value over all 24 configurations.

sessions leave the raw context window, the same backbone incorporates them into the notes. At each task, the agent acts using this window and its notes, without access to the discarded text.

Main results. Table 1 shows substantial differences in task completion and constraint compliance under this shared memory protocol. Spark 1.3 achieves the strongest results across the reported directional metrics, combining high completion with low violation and abstention. Within the flash tier, however, the lowest violation rate does not identify the highest exam score. Qwen3.8-Flash and DeepSeek-V4-Flash have nearly identical overall violation rates, but DeepSeek completes more tasks and achieves a higher exam score. Three of the five flash-tier models score below an always-abstaining

Table 2: Stage 2 memory-framework comparison. Columns as in Table 1. Bold marks the best value per backbone among bounded-memory conditions, excluding the full-context reference (†). Abstention has no preferred direction. RTM: Rule-Tracking Memory; DC: Dynamic Cheatsheet.

Memory	Viol.↓	Comp.↑	Abst.	Exam↑	Far viol.↓	Trap viol.↓	Weighted viol.↓
Qwen3.8-Flash: baselines							
Full context†	0.054	0.839	0.107	+0.779	0.056	0.038	0.054
No memory	0.586	0.158	0.256	−0.451	0.591	0.575	0.603
compact-8k	**0.309**	0.394	0.298	+0.028	0.311	**0.324**	**0.333**
RTM	0.350	**0.550**	0.100	**+0.180**	**0.300**	0.390	0.358
Memory middleware							
mem0	0.620	0.270	0.110	−0.410	0.600	0.670	0.649
MemOS	0.600	0.290	0.110	−0.340	0.620	0.610	0.620
Self-evolving memory							
Memp	0.710	0.170	0.120	−0.580	0.700	0.760	0.731
DC	0.570	0.210	0.230	−0.410	0.580	0.590	0.589
Claude Opus 5.5: baselines							
Full context†	0.009	0.773	0.218	+0.764	0.010	0.010	0.012
No memory	0.573	0.169	0.258	−0.433	0.578	0.562	0.590
compact-8k	0.352	0.479	0.170	+0.107	0.334	0.352	0.361
RTM	**0.315**	**0.515**	0.170	**+0.247**	**0.301**	**0.318**	**0.325**
Memory middleware							
mem0	0.556	0.255	0.189	−0.332	0.547	0.547	0.576
MemOS	0.514	0.262	0.224	−0.275	0.495	0.513	0.531
Self-evolving memory							
Memp	0.627	0.173	0.199	−0.484	0.627	0.618	0.646
DC	0.470	0.277	0.253	−0.215	0.476	0.448	0.483

policy, indicating that their completion gains do not offset violation penalties under the benchmark's scoring rule.

Model tier also does not uniformly predict performance. Although Spark 1.3 and GPT-5.5 achieve higher exam scores than all five flash-tier models, Claude Opus 5.5 falls below DeepSeek-V4-Flash under compact-8k.

5.3 Stage 2: memory-framework comparison

External memory does not improve on compact summaries. Across both backbones, all four external frameworks have higher violation rates, lower completion rates, and lower exam scores than compact-8k. This pattern spans both memory middleware (mem0 and MemOS) and self-evolving memory (Memp and Dynamic Cheatsheet). All four receive negative exam scores on both backbones, whereas compact-8k remains positive.

Retaining history is necessary but not sufficient. Constraint information exists only in the history, so the no-memory control shows what can be inferred from the task turn alone. Compact-8k raises the exam score to $+0.028$ and $+0.107$ and lowers the violation rate from 0.586 to 0.309 and from 0.573 to 0.352. The four memory frameworks recover less than half of this exam-score gain on either backbone, and all remain below zero. Memp does not improve on no memory. On Qwen3.8-Flash, none of them reduces violations appreciably below the no-memory level.

Full context provides the upper reference. The violation rate falls to 0.054 and 0.009 and completion rises to 0.839 and 0.773, above compact-8k and all four frameworks. This suggests that most bounded-memory failures come from what memory preserves and supplies, not from an inability to apply the rules. The gap from compact-8k to full context marks the headroom for memory design.

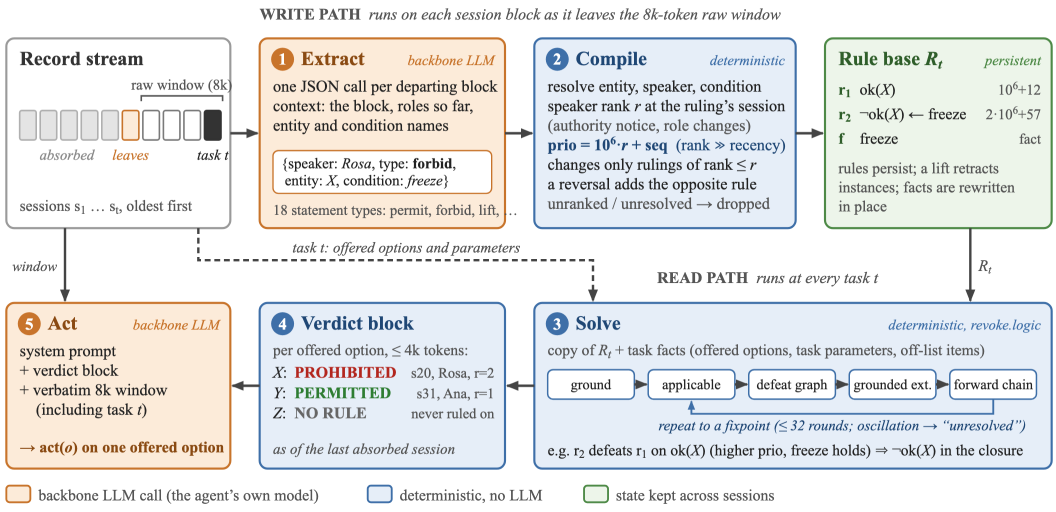

Figure 6: The proposed Rule-Tracking Memory (RTM) architecture.

Revoked permissions remain actionable in memory. Execution logs from multiple evaluated frameworks show the same failure chain: a permission is revoked, memory later returns the earlier permission, and the agent acts on it. Historical permissions are retained without reliably preventing their use as current authorization. This observed mechanism motivates explicit rule-state tracking, examined next.

5.4 Rule-Tracking Memory

We leverage our two-layer graph representation to introduce *Rule-Tracking Memory* (RTM). RTM is a baseline that constructs an explicit active rule state using only the blind inputs available to competing memory systems (Figure 6).

Higher exam score with different trade-offs. Table 2 shows that RTM achieves the highest exam score among the evaluated bounded-memory conditions on both backbones. Relative to compact-8k, it improves the score by 0.152 on Qwen3.8-Flash and 0.140 on Claude Opus 5.5. On Qwen3.8-Flash, RTM increases completion by 15.6 percentage points and reduces abstention by 19.8 points, while increasing violations by 4.1 points. On Claude Opus 5.5, RTM instead improves both completion and compliance: violations decrease by 3.7 points and completion increases by 3.6 points, with essentially unchanged abstention. These gains close about a fifth of the exam-score gap between compact-8k and full context on both backbones.

Where the remaining gap lies. Unlike free-text memories, RTM's state can be checked against ground truth option by option, and because it runs the solver that produces the benchmark's labels, every wrong verdict traces to the extracted rules or to how recently they were updated. Diagnostics on Qwen3.8-Flash place the gap in the accuracy of the extracted state rather than in update coverage. RTM records 88% of prohibitions, 94% of supersessions, and 85% of support events, and 87% of its recorded items match a rule-writing event in the log; yet its derived verdict agrees with ground truth on only 73% of offered options, and in 72% of its violations the memory labels the chosen option as permitted. Errors are most frequent where one rule governs many entities: RTM misjudges 62% of options under a group ban and 49% of options prohibited only by the certification regime, against 42% under a direct prohibition.

Explicit state must stay current. On Qwen3.8-Flash, RTM does not increase violations on far-span tasks. Its net increase comes from tasks with recall span of at most 200 sessions. A likely cause is freshness: RTM's verdict reflects the last session incorporated into memory and can override newer evidence still visible in the raw window.

Table 3: Comparison with prior benchmarks. *Tokens*: history length per instance (–: not applicable or varies). *Authority*: whether an update takes effect depends on who issued it (✓: a ranked hierarchy that changes over time; ○: authorized versus non-authoritative sources). *Rule interact.*: verdicts follow from interacting rules rather than from a single stated fact. *Regen.*: new histories and labelled tasks can be generated. ✓ = tested; ○ = partially; – = not a dedicated axis. †Concurrent work.

Benchmark	Tokens	Revised state	Constr. use	Authority	Rule interact.	Tool actions	Regen.	Grading
Conversational and incremental memory								
LoCoMo	9K	–	–	–	–	–	–	Token F1
LongMemEval	115K–1.5M	✓	–	–	–	–	○	LLM judge
MemoryAgentBench	–	✓	–	–	–	–	–	Mixed
LoCoMo-Plus	–	–	✓	–	–	–	○	LLM judge
STALE	$\leq$150K	✓	○	–	○	–	–	LLM judge
StateMemBench	–	✓	✓	–	–	–	–	Matching + LLM
Memory-enabled execution and policy-following agents								
EvoMemBench	–	✓	–	–	–	✓	–	Mixed
τ-bench	–	–	✓	–	○	✓	–	Programmatic
EAL-Bench†	–	✓	✓	○	○	✓	○	Deterministic ledger
RevokeBench (ours)	344K–935K	✓	✓	✓	✓	✓	✓	Formal solver

6 Related work

Long-horizon memory benchmarks. LoCoMo evaluates long-term conversational memory, while LongMemEval explicitly includes knowledge updates and temporal reasoning (Maharana et al., 2024; Wu et al., 2025). MemoryAgentBench tests incremental memory, including conflict resolution over revised facts (Hu et al., 2026). LoCoMo-Plus evaluates responses against latent constraints, and StateMemBench tests whether answers reflect revised facts, constraints, and decisions (Li et al., 2026; Fan et al., 2026). Table 3 also covers STALE, EvoMemBench, τ-bench, and EAL-Bench (Chao et al., 2026; Wang et al., 2026; Yao et al., 2024; Cerruti et al., 2026).

Memory systems. External memory middleware, including Mem0 and MemOS, provides mechanisms for storing, updating, and retrieving information across interactions (Chhikara et al., 2025; Li et al., 2025). Self-evolving experiential memory methods distill past interactions into reusable insights and procedures, as in ExpeL, Agent Workflow Memory, MemP, and Dynamic Cheatsheet (Zhao et al., 2024; Wang et al., 2025b; Fang et al., 2026; Suzgun et al., 2026). MemEvolve additionally adapts memory architectures and introduces EvolveLab as a shared implementation framework (Zhang et al., 2025a). These methods already support memory revision, our question is whether their updates preserve correct action selection on structured state evolving.

Defeasible reasoning and argumentation. Our conflict-resolution step uses Dung's grounded semantics (Dung, 1995), applied to a defeat graph over applicable ground rule instances with strict priority comparisons. An outer iteration couples this step with forward chaining over explicitly signed literals. The separation between stored rules and derived consequences is also related to materialized-view maintenance, including Delete-and-Rederive (Gupta et al., 1993). Appendix B further details the full list of related works.

7 Conclusion

RevokeBench frames agent memory as a problem of maintaining what remains valid for action. Our two-layer representation makes evolving constraints and their consequences explicit, enabling systematic evaluation. Experiments show that existing memory frameworks can underperform simple summaries, while RTM offers a promising direction through explicit rule-state maintenance. The configurable benchmark supports tailored evaluation settings for developing and studying new memory methods.

Reproducibility Statement

The benchmark construction and evaluation protocols are described in Sections 3 and 4, with additional details in Appendix C and Appendix D. We provide the generator, corpora, solver, verifier, grader, and evaluation scripts in the anonymized supplementary materials. Generation seeds and configurations are included to support reconstruction of the released episodes.

References

David Castillo-Bolado, Joseph Davidson, Finlay Gray, and Marek Rosa. Beyond prompts: Dynamic conversational benchmarking of large language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/4aedf0cba303537fcb6cf948bb41b2df-Abstract-Datasets_and_Benchmarks_Track.html.

Tommaso Cerruti, Mika Okamoto, and Ansel Kaplan Erol. Agent memory is a surface for endogenous authorization laundering. *arXiv preprint arXiv:2609.01836*, 2026.

Hanxiang Chao, Yihan Bai, Rui Sheng, Tianle Li, and Yushi Sun. Stale: Can llm agents know when their memories are no longer valid? *arXiv preprint arXiv:2605.06527*, 2026.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In Inês Lynce, Nello Murano, Mauro Vallati, Serena Villata, Federico Chesani, Michela Milano, Andrea Omicini, and Mehdi Dastani (eds.), *ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025)*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pp. 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarathkrishna Swaminathan, Sihui Dai, Aurélie C. Lozano, Georgios Kollias, Vijil Chenthamarakshan, Jirí Navrátil, Soham Dan, and Pin-Yu Chen. Larimar: Large language models with episodic memory control. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10109–10126. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/das24a.html.

Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995. doi: 10.1016/0004-3702(94)00041-X. URL https://doi.org/10.1016/0004-3702(94)00041-X.

Xinyi Fan, Miri Liu, Ruozhen Yang, Siru Ouyang, and Jiawei Han. Can agent memory systems track evolving state? *arXiv preprint arXiv:2608.19652*, 2026. URL https://arxiv.org/abs/2608.19652.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.866. URL https://aclanthology.org/2026.findings-acl.866/.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. Test of time: A benchmark for evaluating llms on temporal reasoning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=44CoQe6VCq.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/d8efbb5dd415974eb095c3f06bff1f48-Abstract-Conference.html.

Ashish Gupta, Inderpal Singh Mumick, and V. S. Subrahmanian. Maintaining views incrementally. In Peter Buneman and Sushil Jajodia (eds.), *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993*, pp. 157–166. ACM Press, 1993. doi: 10.1145/170035.170066. URL https://doi.org/10.1145/170035.170066.

Priyanshu Gupta, Shashank Kirtania, Ananya Singha, Sumit Gulwani, Arjun Radhakrishna, Gustavo Soares, and Sherry Shi. Metareflection: Learning instructions for language agents using past reflections. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 8369–8385. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.477. URL https://doi.org/10.18653/v1/2024.emnlp-main.477.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/6ddc001d07ca4f319af96a3024f6dbd1-Abstract-Conference.html.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/gutierrez25a.html.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: what's the real context size of your long-context language models? *CoRR*, abs/2404.06654, 2024. doi: 10.48550/ARXIV.2404.06654. URL https://doi.org/10.48550/arXiv.2404.06654.

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

Jihyoung Jang, Minseong Boo, and Hyounghun Kim. Conversation chronicles: Towards diverse temporal and relational dynamics in multi-session conversations. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 13584–13606. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.838. URL https://doi.org/10.18653/v1/2023.emnlp-main.838.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle H. Ungar, Camillo J. Taylor, and Dan Roth. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. *CoRR*, abs/2504.14225, 2025. doi: 10.48550/ARXIV.2504.14225. URL https://doi.org/10.48550/arXiv.2504.14225.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Y. Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/c0d62e70dbc659cc9bd44cbcf1cb652f-Abstract-Datasets_and_Benchmarks_Track.html.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John F. Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26396–26415. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/lee24c.html.

Yifei Li, Weidong Guo, Lingling Zhang, Rongman Xu, Muye Huang, Hui Liu, Lijiao Xu, Yu Xu, and Jun Liu. Locomo-plus: Beyond-factual cognitive memory evaluation framework for LLM agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25085–25100, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1150. URL https://aclanthology.org/2026.acl-long.1150/.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, et al. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*, 2025.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.747. URL https://doi.org/10.18653/v1/2024.acl-long.747.

Kai Tzu-iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seungwon Hwang, Dongha Lee, and Jinyoung Yeo. Towards lifelong dialogue agents via timeline-based memory management. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pp. 8631–8661. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-LONG.435. URL https://doi.org/10.18653/v1/2025.naacl-long.435.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. *CoRR*, abs/2509.25140, 2025. doi: 10.48550/ARXIV.2509.25140. URL https://doi.org/10.48550/arXiv.2509.25140.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. Secom: On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=xKDZAW0He3.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In Sean Follmer, Jeff Han, Jürgen Steimle, and Nathalie Henry Riche (eds.), *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA,*

29 October 2023- 1 November 2023, pp. 2:1–2:22. ACM, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Chen Qian, Yufan Dang, Jiahao Li, Wei Liu, Zihao Xie, Yifei Wang, Weize Chen, Cheng Yang, Xin Cong, Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. Experiential co-learning of software-developing agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 5628–5640. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.305. URL https://doi.org/10.18653/v1/2024.acl-long.305.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Xuanbo Su, Yingfang Zhang, Hao Luo, Xiaoteng Liu, and Leo Huang. Mistake notebook learning: Batch-clustered failures for training-free agent adaptation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14629–14645. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.findings-acl.719/.

Mirac Suzgun, Mert Yüksekgönül, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2026 - Volume 1: Long Papers, Rabat, Morocco, March 24-29, 2026*, pp. 7080–7106. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.EACL-LONG.333. URL https://doi.org/10.18653/v1/2026.eacl-long.333.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long T. Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 8416–8439. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.413. URL https://doi.org/10.18653/v1/2025.acl-long.413.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry W. Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc V. Le, and Thang Luong. Freshllms: Refreshing large language models with search engine augmentation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 13697–13720. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.813. URL https://doi.org/10.18653/v1/2024.findings-acl.813.

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. *Trans. Mach. Learn. Res.*, 2024, 2024a. URL https://openreview.net/forum?id=ehfRiF0R3a.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/ebd82705f44793b6f9ade5a669d0f0bf-Abstract-Conference.html.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian J. McAuley. MEMORYLLM: towards self-updatable large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50453–50466. PMLR / OpenReview.net, 2024b. URL https://proceedings.mlr.press/v235/wang24s.html.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian J. McAuley, Dan Gutfreund, Rogério Feris, and Zexue He. M+: extending memoryllm with scalable long-term memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025a. URL https://proceedings.mlr.press/v267/wang25au.html.

Yuyao Wang, Zhongjian Zhang, Mo Chi, Kaichi Yu, Yuhan Li, Miao Peng, Bing Tong, Chen Zhang, Yan Zhou, and Jia Li. Evomembench: Benchmarking agent memory from a self-evolving perspective. *arXiv preprint arXiv:2605.18421*, 2026.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

Bryan Wilie, Samuel Cahyawijaya, Etsuko Ishii, Junxian He, and Pascale Fung. Belief revision: The adaptability of large language models reasoning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 10480–10496. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.586. URL https://doi.org/10.18653/v1/2024.emnlp-main.586.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=auKAUJZMO6.

Jing Xu, Arthur Szlam, and Jason Weston. Beyond goldfish memory: Long-term open-domain conversation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 5180–5197. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-LONG.356. URL https://doi.org/10.18653/v1/2022.acl-long.356.

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. Knowledge conflicts for llms: A survey. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 8541–8565. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.486. URL https://doi.org/10.18653/v1/2024.emnlp-main.486.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems*

38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/cde328b7bf6358f5ebb91fe9c539745e-Abstract-Conference.html.

Zeyuan Yang, Peng Li, and Yang Liu. Failures pave the way: Enhancing large language models through tuning-free rule accumulation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1751–1777. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.109.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *CoRR*, abs/2406.12045, 2024. doi: 10.48550/ARXIV.2406.12045. URL https://doi.org/10.48550/arXiv.2406.12045.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: how to evaluate long-context language models effectively and thoroughly. *CoRR*, abs/2410.02694, 2024. doi: 10.48550/ARXIV.2410.02694. URL https://doi.org/10.48550/arXiv.2410.02694.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems. *CoRR*, abs/2512.18746, 2025a. doi: 10.48550/ARXIV.2512.18746. URL https://doi.org/10.48550/arXiv.2512.18746.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. *CoRR*, abs/2510.04618, 2025b. doi: 10.48550/ARXIV.2510.04618. URL https://doi.org/10.48550/arXiv.2510.04618.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 5348–5375. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.292. URL https://doi.org/10.18653/v1/2024.acl-long.292.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. Do llms recognize your preferences? evaluating personalized preference following in llms. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=QWunLKbBGF.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez,

and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19724–19731. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. Mquake: Assessing knowledge editing in language models via multi-hop questions. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 15686–15702. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.971. URL https://doi.org/10.18653/v1/2023.emnlp-main.971.

AI USE DISCLOSURE

Generative AI tools were used for language polishing and for assisting with synthetic data generation and dataset expansion. All generated data, experimental procedures, analyses, and manuscript content were reviewed and validated by the authors. The authors take full responsibility for the final content of this work.

A RULE SEMANTICS AND DERIVATION ALGORITHM

A.1 RULE LANGUAGE AND UPDATES

At session t, the persistent rule base is

$$R_t = \{(\mathrm{id}_i, r_i, \pi_i)\},$$

where id_i identifies a rule, r_i specifies its content, and π_i is its integer priority. A rule has the form

$$h \Leftarrow \ell_1 \wedge \cdots \wedge \ell_k.$$

The head and body may contain positive or explicitly negated literals. An empty body, written $\top$, represents an unconditional rule. Variables are sorted and grounded over the finite domains of each instance.

Negation is explicit: $\neg p$ satisfies a premise only when $\neg p$ has been derived, not merely when p is absent. Forward chaining uses membership in a set of signed literals, without negation-as-failure. Complementary heads p and $\neg p$ identify potential conflicts.

The six update operations are:

- ADD writes a rule under a specified identifier.
- SUPERSEDE replaces the head or body of an existing rule.
- CONDITION rewrites a rule's body to change its applicability conditions.
- CONFLICT writes a rule whose head contradicts another rule's head.
- SUPPORT increases a target rule's priority without changing its head or body.
- RETRACT withdraws specified ground instances of a rule.

Writes are keyed by identifier: a fresh identifier creates an entry, whereas an existing identifier replaces it, including for ADD and CONFLICT. The event labels therefore describe update operations rather than mutually exclusive categories of semantic change.

The persistent layer is mutable, not append-only. Its defining property is that defeat alone never modifies or removes a stored rule. At each session, the event history is replayed to reconstruct R_t, which is solved afresh.

A.2 PROPERTIES AND SCOPE

For a fixed applicability set, the defeat graph has a unique grounded extension, independent of the order in which eligible nodes are labelled. Computing this extension and forward chaining are polynomial in the size of the grounded representation. These per-round properties do not establish a unique fixed point or guaranteed convergence of the outer iteration.

A unique grounded extension also does not guarantee a consistent closure. Equally prioritized instances with complementary heads do not defeat one another under the strict priority comparison. If both are applicable and accepted, both conclusions can be derived. Convergence and consistency must therefore be checked separately.

Explicit negation identifies the opposing conclusions on which defeat operates. Non-monotonic changes arise through applicability-dependent defeat and updates to rules or priorities. Where used, staleness attribution is computed from the recomputed defeated set D_t, not from an assumption that the persistent rule base only grows.

B EXTENDED RELATED WORK

Long-horizon conversational memory benchmarks. Multi-session dialogue corpora such as MSC (Xu et al., 2022) and Conversation Chronicles (Jang et al., 2023) first tested whether models retain information across sessions, and LoCoMo scaled this to very long conversations with single-hop, multi-hop, temporal, and adversarial questions (Maharana et al., 2024). The LTM Benchmark interleaves several tasks inside one long user–agent interaction (Castillo-Bolado et al., 2024). Several benchmarks target what must be remembered beyond explicit facts: PrefEval and PersonaMem test whether stated or evolving user preferences and profiles are honoured (Zhao et al., 2025; Jiang et al., 2025). MemoryAgentBench tests incremental memory, including selective forgetting and conflict resolution over revised facts (Hu et al., 2026), and LoCoMo-Plus evaluates responses against latent constraints (Li et al., 2026). Long-context benchmarks such as RULER, BABILong, and HELMET measure retrieval and reasoning over long static inputs rather than over an incrementally revised interaction history (Hsieh et al., 2024; Kuratov et al., 2024; Yen et al., 2024). In these benchmarks, updates appear at most as one question category among many. A few recent benchmarks make revision itself the object of evaluation: StateMemBench tests whether answers reflect revised facts, constraints, and decisions (Fan et al., 2026), STALE asks whether agents recognise that a stored memory is no longer valid (Chao et al., 2026), and EvoMemBench evaluates memory from a self-evolving perspective (Wang et al., 2026). Among agent benchmarks, τ-bench grades tool use against domain policies (Yao et al., 2024), and the concurrent EAL-Bench treats agent memory as a surface for authorization laundering (Cerruti et al., 2026). As Table 3 summarizes, none of these combines an authority hierarchy that changes over time, verdicts that follow from interacting rules, and graded actions over histories of several hundred thousand tokens, which is the combination RevokeBench tests.

Knowledge updates, conflicts, and belief revision. Outside dialogue, MQuAKE asks whether edited facts propagate to multi-hop questions (Zhong et al., 2023); knowledge-conflict studies examine how LLMs behave when the context contradicts parametric knowledge or itself (Xie et al., 2024; Xu et al., 2024); Belief-R tests whether models retract earlier conclusions when new premises arrive (Wilie et al., 2024); and FreshQA and Test of Time probe time-sensitive facts and temporal logic (Vu et al., 2024; Fatemi et al., 2025). These settings present the conflicting evidence within a single prompt or a short passage, whereas our setting requires an agent to track which version is current across a long, multi-session interaction.

Memory systems. Generative Agents store a memory stream with retrieval and reflection (Park et al., 2023), and MemoryBank adds Ebbinghaus-style forgetting (Zhong et al., 2024). Latent-space memories such as LongMem, MemoryLLM, M+, and Larimar compress history into hidden states or episodic memory modules (Wang et al., 2023; 2024b; 2025a; Das et al., 2024); ReadAgent keeps gist memories of very long inputs (Lee et al., 2024); and HippoRAG and HippoRAG 2 organize memory as a knowledge graph for retrieval (Gutierrez et al., 2024; Gutiérrez et al., 2025). For conversational agents, SeCom studies memory granularity and compression (Pan et al., 2025), Theanine links memories on timelines instead of deleting outdated ones (Ong et al., 2025), RMM combines prospective summarization with retrospective retrieval refinement (Tan et al., 2025), and A-Mem lets the agent create, link, and evolve memory notes (Xu et al., 2025). Most of these systems treat updates as a side effect of consolidation or retrieval. We evaluate directly whether their answers reflect the current state.

Self-evolving experiential memory. A separate line of work lets agents distill past interactions into reusable knowledge: verbal reinforcement in Reflexion (Shinn et al., 2023), rules accumulated from failures in TRAN (Yang et al., 2023), insights in ExpeL (Zhao et al., 2024), skill libraries in Voyager (Wang et al., 2024a), policy-level reflection in Agent-Pro (Zhang et al., 2024), experience shared across agents (Qian et al., 2024), meta-reflected instructions (Gupta et al., 2024), context-aware guidelines in AutoGuide (Fu et al., 2024), thought templates in Buffer of Thoughts (Yang et al., 2024), and workflows in Agent Workflow Memory (Wang et al., 2025b). Recent work maintains evolving playbooks (Dynamic Cheatsheet, ACE) (Suzgun et al., 2026; Zhang et al., 2025b), learns from both successes and failures (ReasoningBank, Mistake Notebook Learning) (Ouyang et al., 2025; Su et al., 2026), treats procedural memory as an object with explicit build, retrieval, and update strategies, including deprecation (MemP) (Fang et al., 2026), and meta-evolves the memory architecture itself (MemEvolve), with EvolveLab as a shared implementation framework (Zhang et al., 2025a). These methods accumulate knowledge across tasks; our benchmark instead asks whether the knowledge an agent keeps stays consistent with a state that is repeatedly revised within one interaction.

C Benchmark construction details

This appendix documents the generation pipeline as implemented in the released code. Everything below is deterministic given a seed: no LLM is called during grading. Token counts throughout this paper are the renderer's estimate of one token per four characters, the deviation of API-reported counts from that estimate is given in §C.5.

C.1 Source material and template development

Settings. RevokeBench has six settings, each defined by one JSON *corpus* that supplies vocabulary and surface only and contributes no logic. Every constraint change comes from the setting-agnostic pattern library of §C.2, and every history is graded by the same engine. The settings, with the graded act tool and the permission predicate it asserts, are: a long-stay ward's rounds and nursing handover (`administer`, `administrable`); the group chat of an older adult's care circle (`suggest`, `suggestable`); a design organisation's crit notes merged with its design-system decision log (`place_component`, `placeable`); a `#platform-release` channel export (`call_endpoint`, `callable`); a live-service game studio's weekly narrative review (`deploy_beat`, `deployable`); and a company minute book kept by an AI notetaker (`use_tool`, `usable`). Layout is a property of the setting: the ward, design organisation, game studio and minute book render as sectioned minutes, the care circle and the release channel as flat chat logs (§C.3).

What a corpus contains. The logical half names the things chosen and how each engine event is phrased in this setting. It holds exactly 96 invented entity names, none a case-insensitive substring of another, because the grader resolves tool arguments against the names governed in a history by substring matching and a nested name would be ambiguous; the schema forbids real products, drugs and people, a rule enforced by review rather than by the validator. It holds six context conditions, each written to read after "while", and six groups, each written to read after "nothing in"; two read-only tools and one act tool whose parameter is the setting's sort; between four and eight phrasings for each of thirteen phrasing keys through which the engine's six event kinds and their conditional, grouped and advisory variants are spoken (approval, prohibition, conditional prohibition, supersession, support,

conditioning, widening, carve-out, group ban, advisory objection, escalated objection, condition on, condition off), 70–93 per corpus; phrasings for nine further keys covering the linked, grouped and ranked constructions (alias, alias break, conditional group ban with roster, join, leave, lift, reverse, a rank-2 default, a rank-3 ban), one to eight each and 24–34 per corpus; chat-short *terse* variants of approval, prohibition and conditional prohibition ("{e} is out."), at least three, three and two, 8–11 per corpus; four to seven ways a requester asks for the task; a cast of 13–16 named people with a starting rank in $\{0, 1, 2, 3\}$, at least two at rank 3, one at rank 2, two at rank 1 and four at rank 0, the rank-0 people carrying one-line descriptions ("agency night nurse, not part of the ward establishment"); a list of 7–11 *authority terms*, the words that in this setting assert rank-3 or rank-2 authority ("Council", "P&T", "lead's call"); at least six lines for each of eight kinds of rank-0 noise about an entity (a proposal, hearsay, a question, another team's practice, a stale echo, praise, a withdrawn musing, and the near miss proper), 57–84 per corpus; a numeric threshold specification (the context predicate, the task parameter, its unit, the side on which the ban bites, the threshold's name, four candidate values, and phrasings for setting the threshold, stating the task's value and banning on one side of it); a certification regime (the standing rule, a roster line, on and off lines, grant and revoke lines); a pairwise-incompatibility rule with activation lines; a two-condition rule; and a template for the session-1 authority notice. The surface half fixes the layout; at least six session-title and six header-line templates; for a sectioned layout at least three section headers each for discussion, decisions and actions (a flat layout has none); a two-line frame for the authority notice; at least six lines of machine chrome (bot posts, transcription banners); at least 24 unrelated follow-up items; at least 40 authored multi-line blocks of three to six lines; the label under which the agent's own line appears; and the combinatorial filler: 30 or more first names, 20 or more last names, at least six named slot vocabularies of eight or more entries, and filler templates grouped into families of one register each. The released corpora carry 183–214 filler templates in 13–18 families, 64–71 authored blocks, 38–44 follow-up items and 22–26 one-line asides each.

Two examples. A rule template from the ward corpus, spoken by whoever the generator assigns: *"{e} is the drug of choice for this patient from today. Prescribed regular; nurses please give as charted."* A filler template from the same corpus: *"{p} did the {hour} observations in {bay}: {obs} unchanged across all beds, nothing to escalate."* The padder fills {p} from the first-name × last-name product, {hour} from a numeric range, and {bay} and {obs} from the corpus's slot vocabularies.

Rank neutrality. The corpus does not decide who speaks a rule: a pattern assigns each event a rank, and the generator resolves the rank to a person holding it at that session before the history is rendered. A rule template that names an authority would therefore let a rank-1 speaker say "Council ruling: {e} is prohibited", which a reader would rank above a later rank-2 ruling and the grader would not. Only templates spoken at a rank fixed by the engine may name that authority: the rank-3 ban and lift, the rank-2 default, the pairwise rule and every certification line. Every other rule template must be rank-neutral: all thirteen phrasing keys, the remaining linked and grouped constructions, the terse variants, the threshold-setting and threshold-ban lines, the activation lines and the two-condition rule. The validator compiles the generic words *council, board, committee, director(s), leadership, consultant(s), registrar(s), coordinator, policy, signed off, sign-off, minuted at* together with the corpus's own authority terms into one pattern and rejects any rank-variable template that matches it. Templates say what changes, not whose authority it carries.

Filler invariants. Filler must be unable to alter the closure or to be mistaken for a rule. No governed entity name may appear in any filler template, title, header line, chrome line, follow-up, block or slot vocabulary; the validator checks each of these against the 96 names, and the padder additionally rejects at render time any instantiated line containing one of the 48–49 names governed in the history, redrawing up to fourteen times before accepting a line. No rule-register word may appear in filler: *approve(d), approval, prohibit(ed/ion), ban(ned), supersede-, reaffirm-, reinstat-, contraindicat-, deprecat-, forbid(den), may not, must not, off the list, on the list, no longer allowed, cleared for*; the validator rejects any aside, block, follow-up or padder template containing one. The entity invariant is thus enforced twice, once on the corpus and once at render time; the register invariant once, on the corpus. The validator further requires every template to format cleanly with exactly its permitted placeholders, and no filler first name to coincide with a cast member's listed name. Slot grammar, the requirement that a template read correctly with every value of a slot, is checked by review rather than by code.

C.2 RULE-UPDATE PATTERNS AND SCHEDULING

Shared logical vocabulary. Every pattern is written once, over a fixed vocabulary that each setting merely renames: a permission predicate `allow/1` over the setting's sort (`administrable`, `callable`, and so on); six 0-ary context predicates supplied by the corpus, one further 0-ary predicate for the numeric threshold (`below_renal_threshold`, `over_canary_cap`) and one, `needs_cert`, for the certification regime; a group-membership predicate `grp/2`; a certification-status predicate `uncert/1`; a symmetric incompatibility predicate `pair/2` and an activation predicate `active/1` for the derived pattern; and a feasibility predicate `task_feasible` with the single rule $\texttt{task_feasible} \Leftarrow \texttt{allow}(X)$, which exists for the over-deletion check of §C.4. Bodies may contain negated literals: the alias pattern encodes the linkage as two rules, $\texttt{allow}(x) \Leftarrow \texttt{allow}(y)$ and $\neg\texttt{allow}(x) \Leftarrow \neg\texttt{allow}(y)$, so that x inherits y's status in both directions; the negated rule is silent, and the single alias turn states the linkage once in prose.

Patterns. A pattern is a builder that, given entity, context and group slots, emits an ordered chain of *beats*; a beat is a set of events entering one session, optionally a task (a *probe* in the code) asked on a fixed option-set key, and optionally noise lines attached to that session. Table 4 lists the sixteen patterns the release uses. Ten are the ranked, linked and conditional constructions and are included in every history; a further three are drawn per history from seven simpler constructions, subject to budgets of entities, contexts and groups. A history has six contexts and the ten fixed patterns consume five, so *conflict_chain*, which needs two, never fits, and the two simple patterns that need none, *supersede_alt* and *retract_partial*, always do; the seven are shuffled and taken greedily, so the third slot falls with equal probability to whichever of *conditionalize* (28 histories), *condition_widen* (26), *support_threshold* (26) or *conflict_flip* (20) comes first. The canary pattern is not used in the release. Events emitted by the simple patterns are spoken at rank 1, except SUPPORT, spoken at rank 3.

Cycles. Seven patterns carry a *cycles* parameter that repeats their state arc: the conditional ban whose context toggles (*ctx_flipflop*), the linked entity whose target is banned and reinstated (*alias*), the conditional group ban whose condition toggles while membership changes once (*group_dynamics*), the top-authority ban contradicted, lifted and reissued (*hierarchy*), the retire-and-reverse arc (*reinstate_arc*), the moving threshold (*threshold*) and the two-condition ban (*conjunctive*). Cycles rise with the length band, from 6 to 10, so that the number of rule events per session stays roughly constant along the ladder, 0.50 to 0.45 per session from the shortest band to the longest; because the padding budget grows faster than the number of sessions, rule events per token nevertheless fall from 61 to 32 per 100k tokens. Raising cycles does not make any single decision harder to derive; it raises the compression a memory system faces, because an entity whose status has toggled eight times is summarised correctly only if the latest toggle is kept and nothing in the text marks it as the latest.

Scheduling. The beats of one pattern are internally ordered and independent of every other pattern's, so the beats of all patterns in a history form a DAG whose linear extensions are exactly the admissible timelines. The generator draws one by randomised topological linearisation: at each step it chooses a chain with probability proportional to the number of beats it has left and emits that chain's next beat, which keeps long patterns from clumping at the end. Consecutive beats are placed 2–4 sessions apart. A beat's task is asked 0 to $\mathrm{gap} - 1$ sessions after the beat; with probability 0.35, and provided the following session still lies inside the beat's window, an *echo* of the same task is asked there, to make adaptation lag observable; 22% of generated tasks carry one, and echoes are excluded from the released task sets. The history ends 0–2 sessions after its last beat or task.

History-level structure. Four constructions are laid over the linearised timeline rather than emitted by patterns. (i) The *speaker hierarchy*: session 1 of every history is a notice from the highest-ranked person, rendered from the setting's own template, naming the three ranks, the people holding each, and the rank-0 people with their descriptions, and stating in the setting's words that a ruling stands until the level that issued it or a higher one changes it regardless of what anyone lower says later, that everyone else is discussion including anything that sounds like a reminder, and that only something actually approved counts as doing the task (the validator checks the notice's placeholders; its content was reviewed). One to three role changes (2.1 per history on average) are scheduled between session 8 and the last eight sessions and announced as rank-3 notes: a promotion into rank 2

or 3, drawn with probability 0.6 (123 of the 207 released changes), reads "As of today {who} joins {role}"; a demotion to rank 0 (84 changes) reads "{who} has left {role} and no longer sets policy here". A person's rank is resolved at the session an event is spoken, so a ruling issued at rank 3 keeps that rank after its speaker is demoted. (ii) The *certification regime*: a rank-3 rule entering at session 3–8 states that while the regime is on, an option not on the list is a violation, encoded as $\neg\texttt{allow}(X) \Leftarrow \texttt{needs_cert} \wedge \texttt{uncert}(X)$ at priority 90,000; the flag is switched on and off by rewriting one rule identifier, in windows of 12–30 sessions separated by 10–24, the last window truncated where necessary to end two sessions before the history does, giving 10–19 windows per history (14.4 on average); the roster, announced once, is the eight anchors plus at least three and about a quarter of the other entities in play, and is amended up to four times by grant and revoke notes. (iii) *Anchors*: eight entities no pattern touches, approved by a rank-2 speaker in sessions 2–4 at priorities $80{,}000 + i$, which keep every option set completable; each anchor carries one or two arcs, a rank-3 prohibition later reversed by the same level 20–70 sessions on, at priority $85{,}000 + 10i + k$ so that a second arc strictly outranks the reversed first, the first arc starting in the first half of the history; 449 of the 800 released anchors carry one arc and 351 two. (iv) *Referential updates*: with probability 0.6 a *lift* whose origin is unambiguous, the rank-3 withdrawal of its own standing prohibition in the *hierarchy* and *stale_reminder* patterns, is re-spoken by the original speaker as "Our ruling from session {s} is withdrawn" or "We are lifting the block we issued in session {s}", without naming the entity; the release contains 533 such lines, 60% of the 884 eligible lifts. The session number they cite is the generator's session index, which the task records in the released files also use (§C.3).

C.3 Rendering and task selection

Turn kinds. A generated history has seven kinds of turn, none marked as its kind in the blind view. A single *notice* turn opens the history with the authority declaration, framed by two meta lines. *Update* turns carry the rule events. *Noise* turns are built to read like updates without being one. *Aside* turns are one-line remarks from the corpus, spoken by a rank-0 or rank-1 person with probability 0.4 per session. *Task* turns ask the task. *Filler* turns are the combinatorial padding that carries the history to its length band. *Meta* turns are session titles, header lines, section headers and machine chrome. Notice, update, noise, aside and task turns are speaker-attributed; meta turns are not; filler turns carry a speaker only in the flat layout, where every padded line is prefixed with the name of a filler person who never speaks a rule. The blind view keeps each turn's text and the task marker and strips the kind and speaker fields (the speaker survives as the prefix of each attributed line), so a scaffold cannot filter updates from noise by construction.

Noise. Beside the noise lines a pattern attaches to its own beats, every session after the first receives, with probability 0.8, one to three noise lines from rank-0 speakers aimed at the current state. The generator re-solves the closure at that session and partitions the entities mentioned so far into currently banned and currently licensed. With probability 0.3 it emits a *near miss*: a line phrased exactly like a recorded decision, spoken by someone with no authority, about a banned or a licensed entity ("So {e} is out then. That is my read." in the game-studio corpus). Otherwise, with probability 0.32 it emits a stale echo, praise or question about a currently banned entity; with probability 0.26 hearsay, a proposal, another team's practice or a question about a currently licensed one; and otherwise any kind about any mentioned entity. The eight kinds are: *proposal* (floating an idea, explicitly not deciding), *hearsay* (rumour, nothing in writing), *question* (asking whether {e} is still allowed), *other team* (another ward, guild or team did something with {e}, which does not bind us), *stale echo* ("reminder from my notes: {e} is the default"), *praise* ({e} worked well), *rescind* (withdrawing one's own earlier musing) and *near miss*. Histories carry 645–1,094 noise turns each.

The task turn. A task is asked by a rank-1 or rank-2 person, or, with probability 0.6 whenever a violating option exists, by a rank-0 person who inserts, between the task and the option list, a nudge from the corpus towards one violating option ("I'd offer {e} like usual." in the care-circle corpus); 757 of the 1,202 released tasks carry such a nudge. The turn states the task in the corpus's phrasing, gives the value of the numeric parameter the task carries, lists the options, and closes with a fixed sentence: *"The only options open to you right now are: … Pick exactly one that we have approved for this, carry it out with* `act`*, then say which one you used."* The option set is built once per option-set key and held fixed across every task of the pattern that shares it, so that a stale trap bites. It starts from the two or three entities the pattern itself tests and adds one further licensed

Pattern	Histories	Shape	What a stale memory gets wrong
hierarchy	100	rank-3 ban, later rank-2 "default", lift, rank-2 ban	recency is not authority
derived	100	pairwise incompatibility; the partner becomes active, inactive, then active again	a prohibition that is never stated, only derived
stale_reminder	100	rank-3 ban, then unranked "reminders" that x is the default	statements that look like rules but are not
proposal_noise	100	proposals and hearsay that never become rules	treating a proposal as a ban
alias	100	"x follows y"; y banned, restored, banned; the inheritance of bans is withdrawn while the positive link stays	transitive effects, then their absence
threshold	100	ban on one side of a numeric limit (above it in four settings, below it in two); the limit moves	a threshold compared against its old value
group_dynamics	100	conditional group ban; the condition toggles, members join and leave once	membership stated dozens of sessions earlier
retract_partial	100	ADD $\rightarrow$ group ban $\rightarrow$ RETRACT	a member leaves the blanket ban exactly as the old carve-out closes
ctx_flipflop	100	conditional ban; context on/off repeated; then widened	a cached "x is banned" is wrong half the time
supersede_alt	100	ADD $\times 2 \rightarrow$ SUPERSEDE	the consolidated path is retired; the task survives via a second derivation
conjunctive	100	ban only while two conditions hold together	one condition treated as sufficient
reinstate_arc	100	(retire $\rightarrow$ reverse)$\times c \rightarrow$ conditionalise $\rightarrow$ context on/off	a $2c{+}4$-beat arc (16–24 beats at the released cycle counts) on one entity
conditionalize	28	ADD $\rightarrow$ CONFLICT $\rightarrow$ CONDITION $\rightarrow$ context fires	a blanket ban is narrowed, then retriggered by context
condition_widen	26	conditional ban $\rightarrow$ unconditional	the qualifier the agent relied on disappears
support_threshold	26	losing objection is SUPPORTed past the permission	no rule text changes, only a priority
conflict_flip	20	ADD $\rightarrow$ CONFLICT $\rightarrow$ SUPPORT	a banned entity becomes permitted again while its old alternative is withdrawn

Table 4: The rule-update patterns the release uses, with the number of the 100 histories each appears in. Every pattern is stated only in the engine's vocabulary, so all six settings are graded by the same code. Every pattern holds the option set fixed across the tasks it asks while the compliant subset changes; five patterns (*ctx_flipflop*, *hierarchy*, *reinstate_arc*, *conjunctive*, *conditionalize*) enlarge the set exactly once, when a third entity is introduced, and hold it fixed on either side of that step. The certification regime is a history-level rule rather than a pattern and appears in no row.

option, preferring an entity that was forbidden at some earlier session and is licensed at every task sharing the set, so that its licence must be read off the history rather than off what was never restricted; failing that, an anchor, preferring one previously prohibited and rotating through the eight; failing that, any mentioned entity licensed at every such task. Then two or three distractors are drawn only from entities that are forbidden or unlicensed at every task sharing the set, or never mentioned, never from currently licensed entities, so that "identify the banned ones and pick anything else" does not complete the task. Sets are capped at six and shuffled. Released tasks offer five or six options (268 and 933 tasks), 5.8 on average; a single task offers two.

Labels. For each task the generator solves the rule base at the task's session with the task's own facts added, namely the numeric context predicate if the task's value crosses the current threshold and

one `uncert` fact for every entity in play that is off the certification roster at that session. Options are labelled violating, compliant and licensed exactly as in §2. The generator also records the rules blamed for each violation, the *stale trap* set (options compliant at the previous task on the same option-set key that are violations now), whether the compliant set changed since that task, and the identifiers of the instances in D_t.

Two layouts. Sessions in which nothing happens are not rendered; a history's 428–671 generator sessions yield 412–637 rendered ones (519 on average). Every rendered session opens with a title carrying a running count of rendered sessions, which therefore lags the generator's session index, the numbering that referential updates and the released task records use, by up to 36. A calendar date advanced one to three days per rendered session is supplied to the title and header templates and appears wherever a setting's templates place it (every title in the minute book, none in the ward). One or two header lines follow, drawn from the corpus; a line with an attendee slot names three to six of the setting's people, a second such line about half of them. With probability 0.25 a line of machine chrome is added. In a *sectioned* history a discussion header follows, under which filler chunks and the session's noise and aside turns are shuffled as whole items; then a decisions header, under which the session's rule events appear in issue order, attributed to their speakers; then, whenever the session carries a task and otherwise with probability 0.5, an actions header with one to three unrelated follow-ups, the task, and one further follow-up. The section header is a cue and the phrasing of a near-miss line under discussion is a counter-cue; only the speaker settles it. In a *flat* history the same title and headers open the session, after which rule, noise and aside turns keep their order in one stream, filler chunks are inserted at random positions as short threads between one to three filler people, and the task comes last in its session followed by one filler line.

Padding. The renderer computes the character budget needed to reach the band's target and divides it equally over sessions. Each session is filled with chunks until its budget is spent: with probability 0.12 an authored block from the corpus, with probability 0.78 a block of two to five padder lines whose family is redrawn per line, and otherwise a single aside. The padder keeps a global set of every line it has emitted and redraws, up to fourteen times, any line that repeats one or contains a governed entity name; within a session it prefers templates not yet used and falls back to a family's full pool only once that family is exhausted. The authored pools of blocks and follow-ups are drawn without replacement and reshuffled at most twice, after which the padder supplies everything, so no authored item appears more than twice in a history. What matters for grading is not how many distinct lines the templates could produce but how often one actually repeats, and that is measured: 1.5% of filler turns are exact duplicates on average and 5.6% in the worst history, 29,461 duplicated turns in all, of which about a third are one-line asides whose template has no slot and the rest are the second pass over the authored pools. Histories carry 9,254–34,665 filler turns, 85% of all turns in the release, and 1,134–3,092 meta turns.

Task selection. The generator emits 172–272 tasks per history, 21,996 in all, of which 3,992 are echoes. A released history carries k of them, 10–15 by band, chosen from ground truth alone and then screened. Quotas place a fifth of the k in the first third of the history, three tenths in the middle and half in the last. Within a third, tasks are scored by $1.2\,[\text{trap}] + w + 0.5\,[\text{span} > 200]$, less 0.7 for each already-chosen task testing the same event kind, less 2.5 for a trap once traps reach seven in ten of k, less 2 for a *never-restricted* task, one whose every licensed option was first licensed more than 100 sessions before the task and was never forbidden at any earlier session; a small random tie-break is added, echoes are excluded, and no two tasks may share a session. (A further penalty for a second canary task exists in the code and is inert in the release, whose pools contain no canary.) The selected subset is then passed through the acceptance checks of §C.4: a task they reject is excluded and the selection is redrawn, for at most six selections per seed. The released tasks fall 20%, 29% and 51% by thirds; 848 are traps and 19 are of the never-restricted kind. The task turns of unselected tasks are then deleted from the history and the kept tasks renumbered, so the history reads as months of ordinary work with a handful of moments where the agent is asked to act; every rule, noise, aside and filler turn is untouched, so the event log behind each kept task, and therefore its labels, are unchanged.

Released files and reselection. Each history ships as a *full* file (the event log, per-task closures, labels, difficulty features, and the complete task pool with rendered task text and a never-restricted flag) and a *blind* file that a system under test may see: the history's text with kind and speaker fields

removed, the tool surface, the setting text, the display names of the entities in play (needed to resolve tool arguments), and for each task its session, option set and numeric parameter. The blind file carries no event log, rule vocabulary, base facts, closures, labels or difficulty, since any of these would let a parser reconstruct constraints the conversation is supposed to be the sole source of. A released script builds a new full and blind pair from any selection of the pool by strategy (uniform, traps first, longest span first, or one event kind), excluding echo and never-restricted tasks unless asked and never placing two tasks in one session; an explicit-identifier strategy returns exactly the named tasks without these exclusions. Per-task labels of a reselected set are valid without further work, because they depend only on the event log, which passed the per-session checks and is not altered by selection. The set-level checks are not re-run by that script, and check (6) in particular ranges over the rendered text, which changes when task turns are deleted; a reselected set should therefore be passed through the acceptance procedure before it is used as a benchmark, as the released selections were.

C.4 Acceptance checks and audit

The seven checks. A history that fails any check is rejected rather than repaired; a released selection that fails is redrawn without the offending task, as described below. (1) *Convergence and consistency at every session*: for every session s from 0 to n, the rule base folded from the event log up to s, with the feasibility rule added, is solved; the run must converge and the closure must be consistent in the sense of Appendix A. (2) *No degenerate task*: every task has a non-empty option set, at least one compliant option (no must-violate task), at least one licensed option (so the task is completable, and "pick the entity nobody ever mentioned" cannot complete it), and a canary task has no violating option (inert in the release). (3) *Over-deletion guard*: for every task, on the rule base with the task's own facts added, `task_feasible` is derived exactly when some positive `allow`(e) is; since `task_feasible` has one derivation per permitted entity, this catches a closure that lost a conclusion still supported by an alternative derivation, the failure incremental maintenance is prone to. (4) *Discriminative*: at least one task is a stale-memory trap. (5) *Decidable and rendered*: no option is both compliant and violating, and every task has a position in the rendered history within its session range. (6) *Self-containedness*: every violating option's name occurs in some turn preceding the task turn, of whatever kind, so that a violation is always traceable to something the history said. The test is lexical over the rendered text and accepts a mention in a noise line or in an earlier task's option list (41 of the 4,360 released violating options are named only in the latter). (7) *Recoverability*: at every task session, for every defeat edge between applicable instances of event-written rules, the winning rule must have been written more recently than the losing one (a rule's date is that of the last event that wrote it, so a support, a conditioning or a retraction re-dates it), or carry an explicit override tag, or the losing rule must carry an advisory tag, or the winner must outrank the loser under the hierarchy declared at session 1. Otherwise the agent would be graded on a resolution it had no way of reading.

Two check paths. Checks (1) and (7) solve the session-level rule base, folded from the events alone. Checks (2), (4), (5) and (6) solve no rule base: they inspect the task records and the rendered turns, and the labels they inspect were produced on the task-augmented base. Check (3), label generation and grading solve that *task-augmented* rule base, the same fold with the task's facts added: the numeric context predicate when the task's parameter crosses the threshold, and an `uncert` fact for every entity off the certification roster. Convergence is asserted by check (1) on the first path and is not re-tested by check (3) on the second, which reads only the closure. The audit below closes that gap empirically; the outer iteration, its limit and the detection of oscillation are specified in Appendix A.

Where rejection happens. The release builder verifies twice. A generated history is verified before rendering, and a failure discards the seed. After rendering and task selection, the selected subset is verified again; when a failure is attributable to a particular task, that task is excluded and selection is repeated, for at most six selections per seed; a failure that no single task accounts for, or a pool too small to fill the quota, discards the seed at once. Twenty seeds are tried per history before the build reports a failure; no history failed, and all thirty setting-and-band cells are present in the release. Building the 100 histories rejected 374 seeds, 0–19 per accepted history and 3.7 on average, so roughly one seed in five became a released history.

Target tokens	Cycles	Tasks per history	Histories
300k	6	10	24
440k	7	11	22
580k	8	12	18
720k	9	13	18
830k	10	15	18

Table 5: The five length bands of the release. Within a band, histories are spread over the six settings, four or three per setting.

Property	Range per history	Mean	Total
Sessions, generator index	428–671	542	—
Sessions rendered	412–637	519	—
Tokens (estimated)	344k–935k	627k	62.7M
Events	—	—	26,878
Pattern cycles	6–10	7.8	—
Entities governed	48–49	—	—
Certification windows	10–19	14.4	—
Role changes	1–3	2.1	207
Noise turns	645–1,094	870	87,050
Aside turns	154–279	216	21,616
Filler turns	9,254–34,665	20,243	2.02M
Meta turns	1,134–3,092	2,154	215,375
Graded tasks	10–15	—	1,202
Task pool retained	172–272	—	21,996
Seeds rejected	0–19	3.7	374
Properties of the 1,202 graded tasks			
Stale-memory traps	—	—	848
Recall span $>$ 200 sessions	—	—	971
Requester nudges towards a violating option	—	—	757
Tasks whose licensed options were all approved $>$ 100 sessions earlier and never restricted	—	—	19
Options offered	5–6*	5.8	—

Table 6: Release composition. One hundred histories over six settings: 17 each for the ward, care circle, design organisation and release channel, and 16 each for the game studio and minute book. There are 66 sectioned and 34 flat histories. Recall spans use the generator's session index; empty sessions are not rendered. Tokens are estimated at four characters per token. Recall span has median 334 sessions and maximum 653. Filler repetition averages 1.5%, reaching 5.6% in the worst history. *One task offers two options.

Audit. Because task facts can change the derived state, every evaluated state was re-derived from the released event logs through the grader-side reconstruction, independently of the generator's construction path (the derivation engine itself is shared). All 54,282 session-level states, sessions 0 to n of every history, and all 1,202 task-augmented states converged and were consistent, and at no session were two conflicting applicable instances of equal priority both accepted. Every released label was also re-derived under a broader reading of the grading rule, in which the asserted literal is propagated through every instance not defeated at that session rather than only through the accepted applicable ones; the two readings agree on all 6,940 released (task, option) pairs. The outer iteration reached its fixed point in at most three rounds on every task-augmented state, against the limit of Appendix A.

C.5 Detailed release statistics

Events and what the tasks test. What the histories contain and what the tasks ask about are deliberately different distributions (Table 7). ADD is the commonest event and the least tested, because remembering an addition that nothing later disturbs is the easy case. CONDITION events, the

Event kind	In the histories	Graded tasks testing it
ADD	10,860	55
SUPERSEDE	7,793	167
CONFLICT	4,567	255
NOTE (no rule change)	1,490	—
RETRACT	1,184	182
SUPPORT	830	124
CONDITION	154	297
Noise (no event)	—	122
Total	26,878	1,202

Table 7: Events in the release and the event kind each graded task is built to test. The two columns differ by design: task selection is weighted towards the event kinds whose verdict a stale memory gets wrong, not towards the ones the histories contain most of.

statements that make a ban conditional or widen it, are the rarest kind in the histories and the most tested, because a rule with an on/off state is what memory loses most reliably; the on and off changes of those conditions are themselves recorded as ADD and SUPERSEDE events and counted in those rows. NOTE events change no rule: they announce the state that rules read, the current value of the numeric threshold, an amendment to the certification roster, a change in who holds which rank, and no task is classed against them. A further 122 tasks are classed against noise rather than any event and test whether a near-miss turn, a stale reminder of a ban since lifted (41 tasks) or a proposal or question that never became a rule (81), is mistaken for a rule. A requester's nudge towards a violating option is an independent property, present in 757 tasks of every kind (Table 6).

D EVALUATION DETAILS

Reading the action. Only calls to the graded act tool are scored; calls to the read-only tools are counted but carry no verdict. The act tool's argument is meant to hold one entity reference, and it is resolved by string matching against the display names and identifiers of the entities in play, never by a model. The argument values are joined, lower-cased, collapsed to single spaces and stripped of a leading article (*the*, *a*, *an*). If the result equals an entity identifier or a normalised display name, that entity is the action. Otherwise the argument is rejected outright if it exceeds 80 characters or spans more than one line, since an argument of that shape is prose rather than a reference, typically a refusal that enumerates the option set before declining. A short single-line argument is then matched against every entity: an entity is a candidate if its normalised name or its identifier occurs inside the argument, or if the argument is a clipped prefix or fragment of the name covering at least 60% of its length. The action resolves to the candidate only when exactly one exists; zero or several candidates leave it *unresolved*. An unresolved action is neither a violation nor a completion. This rule replaced an earlier substring match that scored refusals as actions and inflated one condition's violation rate from 0.048 to 0.462; the count of unresolved arguments is reported with every table so that the artefact cannot recur unnoticed.

Difficulty weight. Every task carries a weight computed by the generator from ground truth alone, so it is identical for every system and cannot move with anyone's answers:

$$w \;=\; \max\big(0.3,\; 1 + 1.0\,[\,\mathrm{span} > 200\,] + 0.8\,[\,\mathrm{trap}\,] + 0.4\,(n_{\mathrm{viol}} - 1)^{+} - 0.3\,(n_{\mathrm{lic}} - 1)^{+}\big),$$

where span is the recall span, the number of sessions back to the most recent status change (licensed, prohibited or silent) among the offered options that still governs the verdict; $[\mathrm{trap}]$ indicates a stale-memory trap; n_{viol} and n_{lic} are the numbers of offered options that are violations and that are licensed; and $(x)^{+} = \max(x, 0)$. The coefficients follow the sign and rough magnitude of point-biserial correlations with violation measured on 1202 tasks (n_{viol} $+0.21$, trap $+0.17$, span $+0.15$, n_{lic} -0.15); position in the history is excluded because it correlates with violation only through recall span and the event mix. Span tiers for stratified reporting are *near* (< 100 sessions), *mid* (100–200) and *far* (> 200).

E REPRODUCIBILITY AND RUN PROVENANCE

Table 8: Effective configuration. All calls go to the OpenRouter chat-completions endpoint. Only `temperature` is sent among sampling parameters (`top_p`, `top_k`, `seed`, `stop`, `response_format`: not sent, provider default). Memory-system calls use the agent's backbone with reasoning {`"enabled": false`}.

Call	`max_tokens`	Other settings
Agent, full context (`openrouter_runner.py` @7159ec4)	16,000 (tools); 32,000 (JSON fallback)	reasoning effort low (dropped on JSON attempts); temperature 0; timeout 300 s; whole prefix per task, one fresh call per task, own actions not replayed; 12 workers per model
Agent, memory conditions (`memory_runner.py`, uncommitted)	3,000; $\times 3$ on a no-read retry turn	reasoning effort low; temperature 0; timeout 120 s; $\leq$4 tool turns then JSON fallback; raw window 8,000 tokens; recall budget 4,000 tokens (16,000 chars); own actions replayed; 5–25 workers
Compact notes (fold / condense)	6,500	notes cap 4,000 tokens (prompt word limit only; 52% of folds over cap); fold at $\geq$1,000 pending tokens; notes not clipped at recall
mem0 2.1.0, additive extraction	2,000 (fw25); 6,000 (fw25_fix)	one call per departing block; top-k 12 (always 12 hits); threshold 0.1 (never binding); 0 updates/deletes
MemOS 2.0.33 extraction	8,192	buffer 1,024 tokens; top-k 12
Memp (vendored EvolveLab)	1,000	2 calls per acted task; top-k 1; buffer 3,000 tokens
Dynamic Cheatsheet curator	8,000	every 2 sessions; sheet hard-cut at 12,000 chars (91.1% of curator calls)
RTM extraction (`ruletrack.py`, untracked)	1,800	one call per departing block; verdicts by the rule solver (no LLM call)
Embedder (mem0, MemOS, Memp)	–	all-MiniLM-L6-v2 via sentence-transformers 6.1.0, local; HF snapshot 1110a243 (not pinned)

Table 9: Compute on the 100 Stage 2 episodes: provider-reported charge (`usage.cost`) summed over every recorded call, memory-system calls included. [§]Lower bound: the full-context runner records usage only for the final response of a task.

Condition	Tok. in (M)	Tok. out (M)	$/episode	$\times$ compact
Compact-8k	518.2	406.1	2.714	1.00
Full context[§]	522.8	8.8	0.235	0.09
RTM	160.5	10.0	0.202	0.07
mem0, fw25[‡]	539.7	88.9	0.788	0.29
mem0, fw25_fix	520.8	104.4	0.856	0.32
MemOS	172.2	129.2	0.827	0.30
Memp	34.0	4.6	0.056	0.02
Dynamic Cheatsheet[‡]	213.8	111.3	0.852	0.31
RTM, qwen3.7	146.5	5.8	0.057	0.02
Full context, qwen3.7[§]	522.7	4.3	0.885	0.33

Table 10: Sensitivity of the scores to scoring choices (same saved traces, qwen3.8, 100 episodes, step rule *lr*). Difficulty weights are interpolated as $w_i(\lambda) = (1-\lambda) + \lambda w_i$; $\lambda = 1$ is the paper's weighting. CI: 95% cluster bootstrap over episodes at $\lambda = 1$ (Monte-Carlo precision about ± 0.01). Macro: mean of per-episode exam at $\lambda = 1$. The ranking is the same at every λ except that mem0 and Dynamic Cheatsheet swap at $\lambda = 1$ ($\Delta < 0.01$).

Condition	Exam λ=0	λ=0.5	λ=1	95% CI	Macro	W-viol. λ=0 $\to$ 1
Full context	$+0.786$	$+0.781$	$+0.779$	$[+0.654, +0.900]$	$+0.775$	0.054 $\to$ 0.054
Compact-8k	$+0.085$	$+0.041$	$+0.028$	$[-0.202, +0.247]$	$+0.009$	0.309 $\to$ 0.333
RTM	$+0.209$	$+0.191$	$+0.186$	$[+0.008, +0.380]$	$+0.186$	0.355 $\to$ 0.366
mem0, fw25[‡]	-0.351	-0.395	-0.408	$[-0.573, -0.272]$	-0.410	0.622 $\to$ 0.649
mem0, fw25_fix	-0.451	-0.498	-0.513	$[-0.688, -0.298]$	-0.502	0.637 $\to$ 0.665
MemOS	-0.304	-0.333	-0.342	$[-0.472, -0.221]$	-0.360	0.598 $\to$ 0.620
Memp	-0.545	-0.571	-0.578	$[-0.693, -0.487]$	-0.591	0.714 $\to$ 0.731
Dynamic Cheatsheet[‡]	-0.360	-0.395	-0.405	$[-0.576, -0.274]$	-0.414	0.568 $\to$ 0.589

Step rule, compact-8k, violation / exam: all 0.259 / $+0.023$; http 0.261 / $+0.024$; errfree 0.302 / $+0.027$; lr 0.309 / $+0.028$; acted only 0.403 / $+0.037$. RTM: all 0.348 / $+0.182$; lr 0.355 / $+0.186$; acted 0.382 / $+0.201$.

Known limitations.

- *Single runs, no repeats.* Temperature 0 is not deterministic at these providers; in the few repeat pairs that exist (which also differ in runner revision) 40–70% of decisions differed. Differences

below about 0.1 in violation rate between adjacent rows are not resolved; compact-8k and RTM intervals overlap.

- *Dynamic Cheatsheet.* The 79% truncation that motivated the 8,000-token curator budget was measured on 28 curator calls of one episode; at 8,000 tokens malformed outputs fell to 2.6% but 91.1% of curator calls hit the 12,000-character sheet cap. A run with a larger cap was not made.
- *Units and embedder.* "Tokens" in window and budgets are characters/4; qwen3.8's own token count is 1.02–1.07$\times$ that. mem0, MemOS and Memp embed with all-MiniLM-L6-v2, not a hashed fallback.
- *RTM inputs.* RTM reads the structured fields of the blind item (option sets, task parameters, entity display names in its extraction prompt) and uses the same solver as the grader; a threshold with no matching task parameter falls back to the last numeric parameter.
- *Isolation.* Only blind items reach a system under test; both runners refuse labelled files, and ground truth is read only by the offline grader and the RTM diagnostics.