The Compliance Trap: Diagnosing How AI Agents Consume Conflicting Memory

Yixiong Chen[1], **Xinyi Bai**[2], **Alan Yuille**[1]
[1]Johns Hopkins University [2]Cornell University
ychen646@jh.edu

Abstract

Memory is becoming a core component of long-horizon AI agents, allowing agents to reuse past experience when operating web browsers, software tools, and other interactive environments. Existing work mostly treats memory as a *supply* problem, asking what experience to write, how to store it, and which entry to retrieve for the next task. Yet we still lack a clear account of how models consume retrieved memory across a multi-step action trajectory. This *consumption* process matters because it determines not only what memories should be retrieved, but also what models and control policies are needed to use them safely. To diagnose this process, we propose **E**ntry–**P**ropagation–**R**ecovery (E-P-R), a trajectory-level framework that asks where memory first changes an action, whether that change carries forward, and whether the agent can recover after leaving a correct path. We instantiate E-P-R on WebArena and on MemTrapBench, a controlled benchmark we build to isolate these phases. We find that the main failure often begins at entry: agents adopt conflicting memory at the first exposed decision point even when it is task-wrong. Repeated exposure then amplifies this early error, while recovery after divergence is weak. Together, these effects create a **compliance trap**: across models, conflicting memory induces similar compliance rates, but once agents comply, their success rates collapse to a low floor. Agents with higher no-memory baselines can therefore experience greater absolute damage under a similar post-compliance floor. These results suggest that memory-augmented agents should be evaluated not only by retrieval quality or final success rate, but by how they consume memory throughout the trajectory. Code and data are available at https://github.com/Schuture/Compliance-Trap.

1 Introduction

Long-horizon agents are increasingly used to operate web browsers, file systems, software tools, and operating systems [33, 38]. As these tasks span many observations and actions, agents need a way to reuse past experience instead of solving each task from scratch. Memory has become a common mechanism for this purpose. Trajectories from past episodes can be summarized, stored, and retrieved at the start of a new task to guide later decisions [31, 49, 35, 19, 26, 43, 51]. This makes memory a central component of practical agent systems, especially when the task is too long to fit all useful experience into the model context or too costly to reason through repeatedly.

Most existing work studies agent memory from the *supply* side. Systems such as Reflexion [31], ExpeL [49], AWM [35], WebCoach [19], MemGPT [26], A-MEM [43], MemoryBank [51], and RAP [12] differ in how they write, retrieve, or summarize experience, but they share an implicit assumption: once a memory is placed in context, the agent policy will decide how to use it. However, the agent that consumes memory is still an imperfect language model policy, and retrieval does not determine how the model will use the retrieved content. The useful memory can be ignored, followed

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

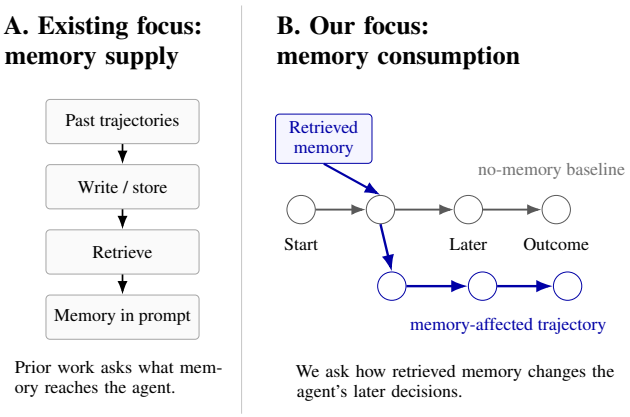

Figure 1: **Motivation and Overview.** (A) Prior work studies the memory supply side. (B) We analyze memory consumption: how trajectories change with/without memory injection.

too literally, or applied again after the state has changed. As a result, a memory that looks plausible to a retriever can be dangerous if the model adopts it at the wrong point or fails to recover from it later. Prior work has observed related failures in experience following [40], but we still lack a trajectory-level account of where memory first changes behavior, how far that change carries forward, and whether the agent can undo it.

Figure 1 illustrates this shift in perspective. Existing memory systems mainly control the supply pipeline: past trajectories are written, stored, retrieved, and inserted into the prompt. Our focus begins after this point. Once a memory enters the agent context, the agent must still decide whether to follow it, ignore it, repeat it after the state has changed, or correct course after it has caused a wrong action. We therefore study memory as a consumption problem. *We ask how long-horizon agents consume retrieved memory across a multi-step trajectory, and which stage of the trajectory determines whether memory helps or hurts.* To answer this, we introduce **E**ntry–**P**ropagation–**R**ecovery (E-P-R), a framework that decomposes memory consumption into three phases. Entry measures whether retrieved memory changes the agent's action at the first decision point where it is exposed. Propagation measures whether this initial change continues to shape later actions under repeated or continued memory exposure. Recovery measures whether the agent can return to a task-correct trajectory after memory-induced divergence.

We apply this framework on two complementary testbeds. On WebArena, we hold memory content fixed and vary only the injection schedule. Injection on different stage (early/persistent/late) probes how memory enters the trajectory, amplifies the effect, and changes existing decision. These schedule interventions give coarse trajectory-level probes of E-P-R. Across five models from the Qwen3.5, Gemma-4, and Gemini-3 families, the picture is consistent: wrong memory enters early, repeated exposure amplifies, and the damage concentrates on trajectories that follow the conflicting memory; once this happens, it is hard to recover. We call this regularity the **compliance trap**. Because the WebArena subset is selected for memory sensitivity, we treat it as suggestive evidence in a realistic environment. To test whether this pattern is specific to the WebArena, we then introduce MemTrap-Bench, a controlled benchmark that serves as our primary statistical validation. By separately probing entry, propagation, and recovery, MemTrapBench lets us test whether first adoption of conflicting memory is the main driver of final failure under a controlled trap structure. Although E-P-R applies to helpful, conflicting, and cross-task control memory, our main analysis focuses on conflicting memory because it exposes whether the agent can reject task-wrong experience rather than merely follow retrieved text.

This paper makes three contributions.

- First, we formulate memory consumption as a trajectory-level problem and introduce E-P-R as a diagnostic framework for measuring entry, propagation, and recovery.
- Second, we identify the compliance trap as a concrete failure mode of memory-augmented agents, where similar compliance rates and a similar post-compliance floor lead to larger absolute damage for models with higher no-memory baselines.

- Third, we introduce MemTrapBench together with a paired WebArena protocol that separates full-benchmark average effects from mechanism diagnosis, showing that final success rate alone can hide where memory-induced failures arise.

2 Related Work

Agent memory systems. Generative Agents [27] and Voyager [32] introduced persistent textual memory for open-ended interactive environments. Subsequent systems largely study memory from the supply side: Reflexion [31] writes failure reflections back into the prompt, ExpeL [49] distills insights across trajectories, Agent Workflow Memory [AWM; 35] compiles reusable workflows with strong gains on WebArena, and WebCoach [19] trains a coaching model on top of retrieved experience. A parallel line of work focuses on memory architecture rather than content: MemGPT [26] treats large language models (LLMs) as operating systems and pages information between fast and slow memory tiers; MemoryBank [51] adds long-term semantic memory with forgetting curves; A-MEM [43] structures memory as a self-organizing graph of agentic notes; RAP [12] couples retrieval with planning for multimodal LLM agents; and Mem0 [3] packages production-grade long-term memory for deployed assistants. Most closely related, Xiong et al. [40] analyze experience-following behavior in memory management systems. Our focus is complementary: we study how the agent policy consumes a given memory once it is in context.

Reasoning, self-correction, and recovery in language agents. A growing body of work asks whether language models can reason about and correct their own actions. Chain-of-thought prompting [36], self-consistency [34], and tree search [47] extend single-step prediction with explicit deliberation, while ReAct [48] interleaves reasoning with environment interaction so that each action is conditioned on a verbalised plan. Self-Refine [23], Reflexion [31], and critique-style training [1, 25] let models revise their outputs from feedback. Two recent surveys, however, document the limits of these approaches: Huang et al. [11] show that LLMs cannot reliably self-correct reasoning errors without an external signal, and Kamoi et al. [13] quantify the conditions under which self-correction helps or hurts. Our Recovery diagnostic targets whether the agent can leave a memory-induced trajectory after it has already taken several actions on the path. Empirically (Appendix E), trajectory-level recovery exists but does not translate to task-level recovery.

Trajectory-level agent evaluation and failure diagnosis. WebArena [52], VisualWebArena [14], OSWorld [39], MiniWoB++ [30], WorkArena [5], Mind2Web [4], WebShop [46], WebVoyager [9], AgentBench [21], AgentBoard [22], and TheAgentCompany [41] evaluate agents on multi-step interactive tasks, usually through final task success or per-step progress. A separate strand develops the visual grounding side of these agents: SeeClick [2], CogAgent [10], SeeAct [50], and Set-of-Mark prompting [45] show that linking instructions to concrete UI elements can dominate raw planning quality, an observation that motivates our explicit Grounding gate (Section 4). Earlier work on browser-assisted question answering, including WebGPT [24] and the BrowserGym ecosystem [16], established the action interface that all of these benchmarks now share. Recent diagnostic benchmarks such as WebSuite [18] go beyond binary success by attributing failures to specific web-action types, and AgentBoard [22] reports fine-grained progress rates. These analyses identify where agents fail in the environment. Our analysis instead asks how an injected memory changes the trajectory.

Context utilization and misleading context. A separate line of work studies how language models use information provided in context. Lost in the Middle [20] shows that position within a long context affects whether relevant information is used. CUB [8] benchmarks context-utilization methods under different context conditions. Work on parametric-context conflict studies when models follow retrieved evidence versus internal knowledge [42, 17], while sycophancy work shows that models can agree with misleading user-provided pressure [28, 29, 37]. These studies mainly evaluate static responses, while we study the analogous problem in interactive agents.

3 Method

We study memory consumption by intervening on two variables: *what* the memory says and *when* the agent can see it. A memory is a short textual passage inserted into the agent context during an episode. Its content is one of three types. A helpful memory gives task-relevant guidance, a

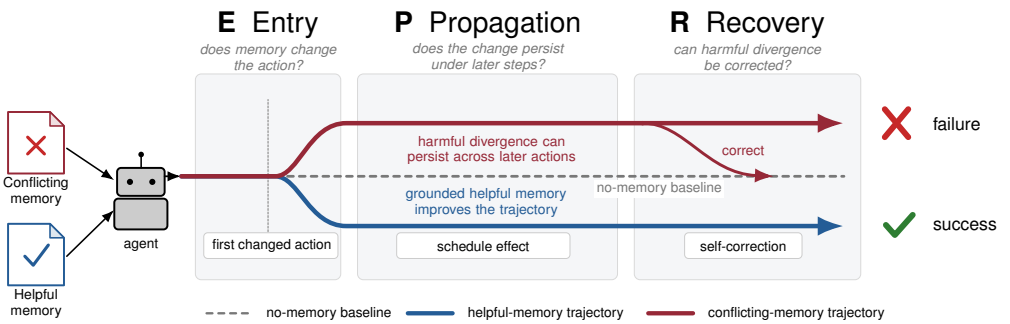

Figure 2: **Entry–Propagation–Recovery diagnostic framework. Entry**: does retrieved memory change the agent's action? **Propagation**: does the change persist under later observations? **Recovery**: can the agent correct a harmful deviation? Helpful memory improves trajectories when grounded; conflicting memory harms when adopted and not recovered.

conflicting memory gives plausible but task-wrong guidance, and a same-website cross-task control memory provides a content control. Its schedule specifies the decision steps at which the memory is available: early (only the first decision point), persistent (every decision point), or late (only after the first few actions have already been taken).

By fixing the memory passage and changing only its availability over time, we test the agent policy's use of memory rather than the quality of retrieval. This setting follows the common design of external textual memories used in systems such as Reflexion [31], ExpeL [49], and AWM [35].

3.1 Entry–Propagation–Recovery diagnostics

We treat memory consumption as a trajectory-level process rather than a single retrieval event. A retrieved memory first competes with the no-memory policy at the next decision point; if it changes the agent's action, the trajectory branches away from the no-memory baseline; the branch can then persist through later observations or be corrected. Figure 2 summarizes the three stages this defines: **Entry** asks whether memory changes the first action it can affect, **Propagation** asks whether that change carries forward under continued memory exposure, and **Recovery** asks whether the agent returns to a task-correct trajectory after divergence. The three stages together account for the full path from a memory passage to a final outcome, so a metric defined on any one of them is a partial proxy for the others.

We operationalize E-P-R through paired interventions. For each (task, model) cell we run one no-memory trajectory and one trajectory per (memory content, schedule) combination from the same environment seed; primary results are reported as the success-rate (SR) difference against the paired no-memory trace. Three injection schedules each probe one stage: **early** injection (memory visible only at step 1) targets Entry; **persistent** injection (memory visible at every step) reads off Propagation by amplifying or dampening the early effect; **late** injection (memory withheld until step 3) probes whether an already-formed action history resists new guidance, complementary to Recovery. Recovery itself is most directly read off MemTrapBench (Section 3.3), whose controlled task structure fixes a known correct path and a known trap action so trajectory-level realignment after a wrong action is observable [11, 13].

3.2 Compliance under conflicting memory

For conflicting memory, action divergence alone is not enough: the agent may differ from the no-memory trajectory without actually following the injected recommendation. We therefore measure compliance at the first memory-exposed decision step using the **Recommendation Compliance Rate (RCR)**: an LLM judge classifies whether the conflicting step-1 action plausibly executes the first primitive of the memory's recommendation, with the paired no-memory step-1 action as joint evidence (rubric in Appendix B). Let $\mathcal{C}^{\mathrm{RCR}}$ denote the compliant subset (judge=YES). The cost of compliance is then the *Damage Per Compliance* (DPC):

$$\mathrm{DPC}(\mathcal{C}^{\mathrm{RCR}}) = \frac{1}{|\mathcal{C}^{\mathrm{RCR}}|} \sum_{i \in \mathcal{C}^{\mathrm{RCR}}} (y_i^0 - y_i^c),$$

where y_i^0 and y_i^c are paired no-memory and conflicting-memory success on task i. Pairing on $\mathcal{C}^{\mathrm{RCR}}$ avoids comparing an all-task no-memory baseline to compliance success on a behavior-selected subset. DPC separates the *rate* at which the agent commits to a wrong recommendation (RCR) from the *conditional cost* of that commitment. The separation is essential because RCR turns out to be approximately constant across models (Section 4.3, $\sim 65\%$), so unconditional success drops mostly reflect how far compliant trajectories fall rather than how often the agent complies. RCR and DPC together are the primary readout of the trap in Tables 2 and 3; a coarser action-divergence subset is reported as a robustness check in Appendix B.

3.3 Testbeds and protocol

Models. We evaluate Qwen3.5-9B, Qwen3.5-27B, Gemma-4-E4B-it, Gemma-4-26B-A4B-it (hereafter Gemma-4-E4B and Gemma-4-26B), and Gemini-3-Flash on both testbeds. The agent observes the page through the BrowserGym accessibility tree (AXTree) [16]. The open-weight models decode greedily with vLLM, so two of their trajectories on the same task and seed differ only when memory injection differs; Gemini-3-Flash is queried with the API's default sampling settings.

Memories. A memory is a short text passage placed in the agent context. All memories follow a single `DO/DON'T` template, and helpful and conflicting variants are matched in length, word count, and structural elements. Conflicting variants are plausible but task-wrong: they reference real UI elements on the same site and are written so that following them produces a well-formed but incorrect trajectory rather than a parse failure. WebArena memories are hand-authored. MemTrapBench memories are drafted by Claude Opus 4.7 from each task spec and reviewed by hand.

Statistics. All paired contrasts use 10k-iteration sign-flip permutation tests; reported confidence intervals are paired bootstrap percentiles with 10k resamples. Full audits, inference settings, and the RCR rubric are in Appendix A.

WebArena. [52] We use the 684 non-map tasks for the full-benchmark pre-check and a 77-task memory-sensitive subset (tasks whose outcome changes under early helpful or conflicting memory on Qwen3.5-9B or Qwen3.5-27B) for trajectory-level analysis.

MemTrapBench. 231 long-horizon (median 15 steps) browser tasks on a shared HTML/JS skeleton served through BrowserGym. Each task pairs a verifiable goal with a plausible decoy and falls into one of four diagnostic families, each targeting one E-P-R gate by construction:

- DECOY (Entry): a prominent decoy sits beside the correct path.
- UPTAKE (Adoption): the alternate route is hidden behind an Advanced toggle.
- GROUNDING (Grounded|Adopted): a memory-referenced UI label is absent from the AXTree.
- OVERRIDE (Recovery): memory tells the agent to ignore an explicit environment correction.

Together these decompose the helpful lift into a multiplicative gate chain $\Delta_{\text{helpful}} = P(\text{novel}) \cdot P(\text{adopt}) \cdot P(\text{ground} \mid \text{adopt}) \cdot P(\text{lift} \mid \text{ground})$; the same gates govern conflicting damage. Tasks and memories are authored from per-family blueprints by Claude Opus 4.7 and verified by hand: each task is rolled out once under no-memory and once under persistent-conflicting during construction to confirm the correct path succeeds and the trap action lies on a wrong path. Templates, family counts, and audit are in Appendix D.

4 Results

We organize the results around the E-P-R chain. E-P-R is not a claim that retrieved memory is generally harmful. It treats memory as a high-leverage input whose effect depends on whether it enters the trajectory, remains grounded in the page state, and can be corrected when wrong. Our main contribution is a quantitative account of the unsafe direction: prior work on agent memory has focused almost entirely on raising the upside of helpful memory, while the failure mode in which a task-wrong memory enters the trajectory and resists correction has not been systematically characterised. We therefore analyze helpful and conflicting memory together: helpful memory is the positive control showing that the same gates control benefit when novelty and grounding are

Table 1: **Full-WebArena retrieval pre-check.** Online-generated memory across 684 non-map tasks. Memory based on the simple contrastive setting cannot uniformly improve the model performance with no memory, no matter the retrieved memory is gated or not.

Model	Unconditional retrieval	Relevance-gated retrieval
Qwen3.5-9B	-1.9 pp	-2.3 pp
Qwen3.5-27B	$+1.7$ pp	-1.7 pp

satisfied, while conflicting memory exposes the unsafe regime in which the same gates let a wrong recommendation control the trajectory.

4.1 Full-WebArena pre-check motivates consumption-side diagnosis

Before any hand authoring or outcome-based task selection, we run a motivation check (Table 1) on the full 684-task WebArena distribution with online-generated memories in the contrastive DO/DON'T format. The effects on Qwen3.5-9B and 27B are not consistent (-1.9 pp vs. $+1.7$ pp). After trying raising the relevance threshold to increase the memory quality, no performance gain is observed: damage worsens slightly on Qwen3.5-9B, and Qwen3.5-27B flips from a small gain to a small loss. This pre-check experiment shows that memory does not automatically bring benefit to tasks, especially when the task is complex and long-horizon. An in-depth exploration of how memory affects the agent's behavior is thus needed.

How often are retrieved memories conflicting? Aggregate success does not reveal how often an automatically retrieved memory is itself task-conflicting. Two authors therefore independently audited 100 memories retrieved by this pipeline, labeling them as helpful/task-correct (83 / 80), conflicting/task-wrong (11 / 13), irrelevant (2 / 3), or ambiguous (4 / 4); they agreed on 97 of 100 memories (Cohen's $\kappa = 0.91$). Targeted conflicts are thus a minority ($\approx$11–13%) but occur non-trivially under automatic retrieval. We keep two claims separate: (i) task-conflicting memories occur in the tested automatic pipeline at roughly this rate; and (ii) conditional on such a memory entering the context, the controlled experiments below diagnose how it causes failure. Our experiments directly support (ii); we do not estimate the expected deployment risk of other retrieval pipelines.

4.2 Conflicting memory enters early and persistent exposure amplifies harm

Figure 3 shows success-rate deltas under early, persistent, and late injection for helpful and conflicting memory across five models on the 77-task subset. Persistent conflicting memory is the only condition that hurts every model, with significant drops on Qwen3.5-27B and Gemini-3-Flash; the same direction holds for the smaller open-weight models. Persistent conflicting is also significantly worse than early conflicting on Qwen3.5-27B ($p = 0.013$); the harm is not a single misclick but accumulates with repeated exposure, consistent with Propagation. Late injection has only a weak effect because the median WebArena trajectory is ≈ 7 steps and the agent has already committed to an action history.

Helpful memory yields an asymmetrical effect on models. It only has modest improvement on the models that are sensitive to the conflicting memories (Qwen3.5-27B and Gemini-3-Flash), but helps more for the models that resist conflicting memories (Gemma-4-E4B and Gemma-4-26B-A4B). The asymmetry indicates that the behavior of models after memory injection is specific to the model family and size, putting more importance on the consumption-side exploration.

4.3 The compliance trap: similar compliance, larger conditional damage

We focus on the WebArena subset that all the memories' following effect can be verified by LLMs. Table 2 reports the RCR and paired DPC on all five WebArena models. Three observations define the compliance trap. (i) RCR is high and approximately scale-independent (63–72% across the five models). (ii) Conditional success after compliance collapses to a near-constant floor (17–31%), well below the no-memory baseline on the same subset (23–49%). (iii) Because higher-baseline models start higher but land on a similar low floor, paired DPC tracks the no-memory baseline: from 2.1 pp on Gemma-4-E4B (baseline 23%) to 25.5 pp on Qwen3.5-27B (baseline 49%). This is a statement

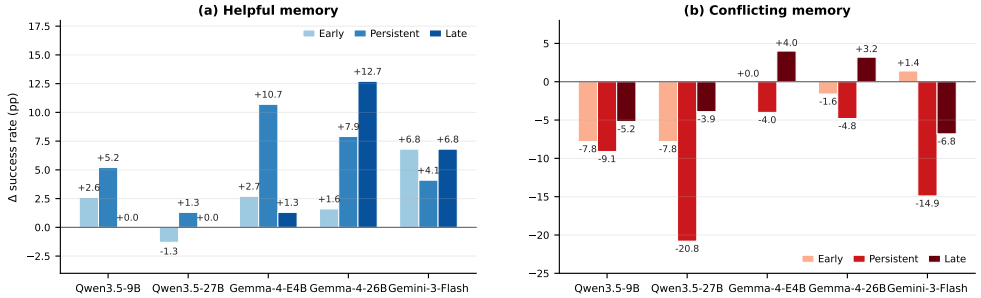

Figure 3: **WebArena schedule deltas on the 77-task subset (5 models, including Gemini-3-Flash, n=74).** Deltas are paired against the no-memory trace on the tasks where all schedule conditions are valid for that model; this stricter inclusion rule is why n differs from the per-model N of Table 2. (a) Helpful memory: modest gains, mostly persistent. (b) Persistent conflicting hurts every model; peaks on Qwen3.5-27B (-20.8 pp) and reaches -14.9 pp on Gemini-3-Flash. Late conflicting is weak on WebArena's $\approx$ 7-step median; MemTrapBench (Section 4.4) restores late helpful. Cross-task control within ± 3 pp (Appendix C).

Table 2: **Paired compliance trap under persistent conflicting memory on WebArena 77.** $\mathcal{C}^{\mathrm{RCR}}$ indicates LLM-judge step-1 Recommendation Compliance subset. $\mathrm{SR}^{0}_{\mathcal{C}^{\mathrm{RCR}}}$ and $\mathrm{SR}^{c}_{\mathcal{C}^{\mathrm{RCR}}}$ are success rates for no memory / with memory. Paired DPC$= \mathrm{SR}^{0}_{\mathcal{C}^{\mathrm{RCR}}} - \mathrm{SR}^{c}_{\mathcal{C}^{\mathrm{RCR}}}$. N counts the tasks with valid no-memory and persistent-conflicting traces for each model, so it can differ from the all-schedules-valid n of Figure 3.

Model	N	RCR	$\lvert\mathcal{C}^{\mathrm{RCR}}\rvert$	$\mathrm{SR}^{0}_{\mathcal{C}^{\mathrm{RCR}}}$	$\mathrm{SR}^{c}_{\mathcal{C}^{\mathrm{RCR}}}$	Paired DPC [95% CI]
Gemma-4-E4B	75	63%	47	23.4%	21.3%	2.1 pp [−10.6, +14.9]
Gemma-4-26B	67	63%	42	38.1%	30.9%	7.1 pp [−4.8, +19.1]
Qwen3.5-9B	77	69%	53	26.4%	17.0%	9.4 pp [−3.8, +22.6]
Gemini-3-Flash	75	72%	54	46.3%	27.8%	**18.5 pp [+3.7, +33.3]**
Qwen3.5-27B	77	66%	51	49.0%	23.5%	**25.5 pp [+11.8, +39.2]**

about task-specific baseline success rather than model scale: under similar RCR and a similar floor, a higher baseline mechanically implies larger absolute damage, and our cross-family Gemma-3 probe (Appendix F) shows that parameter count alone does not predict vulnerability. On WebArena n=77, all five DPC estimates have the same sign, but the CIs exclude zero only on the two highest-baseline models (Qwen3.5-27B and Gemini-3-Flash). We therefore read WebArena as suggestive evidence in a realistic environment and rest the main statistical claim on MemTrapBench (Section 4.4).

Figure 4 visualizes the trajectory-level signature of the trap on the two Qwen3.5 models. Panel (a) shows that the cumulative probability of first divergence reaches 0.86–0.94 by step 14 and is already high within the first few steps under both helpful and conflicting memory, so memory enters the trajectory early. Panel (b) shows that conditional on having diverged, helpful-memory trajectories re-align with the no-memory trajectory more often by step 14 (27–42%) than conflicting-memory trajectories (7–15%); the trap is therefore not only early adoption but harmful divergence that is less recoverable.

4.4 MemTrapBench validates E-P-R without WebArena outcome selection

The 77-task subset is selected on memory response, so its absolute effect sizes may be inflated. MemTrapBench (Section 3.3) addresses this by defining traps via construction rather than by downstream outcome, and by freezing helpful, conflicting, and cross-task control memories before any model evaluation.

Table 3 shows that the WebArena pattern is not a subset artifact. Persistent conflicting memory remains content-specifically harmful, while persistent cross-task control stays near zero. Damage grows with the no-memory baseline: -9.5 pp on Qwen3.5-9B, -21.2 pp on Qwen3.5-27B, and -26.0 pp on Gemini-3-Flash. The right two columns report paired DPC + 95% CI on the trajectory-

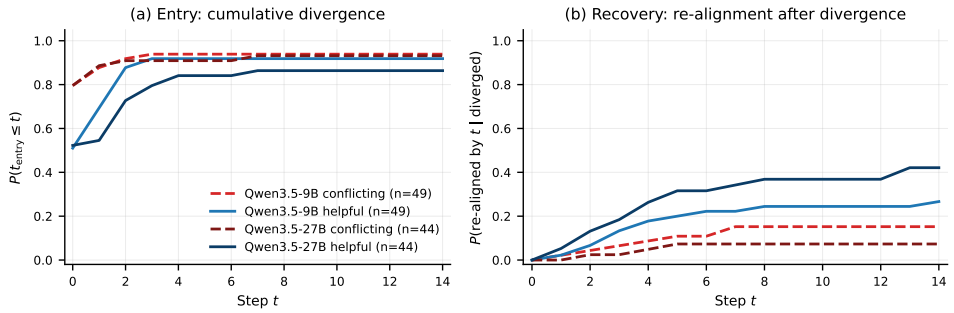

Figure 4: **Entry and recovery on WebArena under persistent injection.** (a) Memory enters early. $P(t_{\text{entry}} \leq t)$ reaches 0.86–0.94 by step 7 on all models. (b) The recovery rates are low. Conditional on divergence, helpful trajectories re-align (27–42%) more often than conflicting (7–15%).

Table 3: **MemTrapBench persistent-injection deltas (231 long-horizon tasks).** Right two columns: paired DPC + 95% CI on the action-divergence subset (step-1 action differs from no-memory); all four models exclude 0, $p < .001$.

Model	Base	Pers. Helpful	Pers. Conflict	Cross-task	Paired DPC	95% CI
Qwen3.5-9B	10.8%	**+25.5**	−9.5	−2.6	**+10.2** pp	[+5.4, +15.7]
Qwen3.5-27B	22.5%	**+30.7**	−21.2	+1.3	**+26.6** pp	[+19.6, +33.5]
Gemma-4-26B	12.1%	**+29.9**	−10.8	+0.9	**+14.1** pp	[+8.3, +19.9]
Gemini-3-Flash	32.9%	**+29.4**	**−26.0**	−1.7	**+28.6** pp	[+21.9, +35.2]

level divergence subset: *all four models reach CI excluding zero*, making MemTrapBench the primary statistical evidence for the trap and addressing the WebArena sample-size limitation by replicating the trap on the larger 231-task pool with full statistical resolution on every model and across both open- and closed-weight families. Persistent helpful memory acts as a positive control on the same task pool: gains are large and uniform across models (+25 to +31 pp), confirming that our memory passages are usable when the gates allow them through. The cross-task control delta sits near zero, ruling out a context-length artifact. Late helpful memory nearly matches persistent helpful on the three open-weight models (Appendix E), confirming that action-history inertia is a short-horizon effect.

Table 4 breaks the per-diagnostic persistent injection deltas by family. DECOY produces the largest conflicting damage on every model because the wrong recommendation is groundable, immediately executable, and outcome-relevant; UPTAKE and OVERRIDE produce substantial conflicting damage on the stronger models because the wrong recommendation is also reachable. The GROUNDING row shows the asymmetric helpful side of the same gate: when the conflicting recommendation references a UI element absent from the page, the helpful memory simultaneously carries the missing value the task requires, and helpful gains are the largest in the table (+57.9 to +73.7 pp).

5 Discussion

Memory consumption, not prompt position. A natural follow-up is whether the compliance trap is a general property of memory consumption or only appears for one prompt position. By default, the memory sits in the prompt header, right after the task goal and before the action history and page observation. We re-run MemTrapBench with the same memory text inserted at the bottom of each observation instead, so the agent reads memory as part of the page rather than next to the task. Table 5 shows that persistent-conflicting damage is unchanged within ±0.5 pp on all three open-weight models. The trap is therefore a property of how the agent uses memory content, not of where memory appears in the prompt: whether memory sits next to the task or inside the page observation, the same trajectory dynamics apply.

The trap requires a non-trivial trajectory length. Across three benchmarks with different median trajectory lengths (Table 6), persistent-conflicting damage scales with horizon: short-horizon

Table 4: **Per-diagnostic persistent-injection deltas on MemTrapBench.** pH / pC are persistent helpful / conflicting deltas vs. no memory. DECOY (Entry) produces the largest conflicting damage; GROUNDING carries the largest helpful gain because the missing value is memory-borne. The conflicting-side reading on GROUNDING is constrained by a small baseline on the open-weight models ($\leq 1.8\%$); Gemini-3-Flash's higher baseline (17.5%) reveals a clear -15.8 pp pC. We test grounding as a causal gate via the 2×2 factorial in Appendix E instead.

	E-P-R Gate	Qwen3.5-9B		Qwen3.5-27B		Gemma-4-26B		Gemini-3-Flash	
Diagnostic		pH	pC	pH	pC	pH	pC	pH	pC
DECOY	**Entry**	$\mathbf{+14.5}$	$\mathbf{-24.2}$	$\mathbf{+16.1}$	$\mathbf{-35.5}$	$\mathbf{+19.4}$	$\mathbf{-25.8}$	$\mathbf{+12.9}$	$\mathbf{-43.5}$
UPTAKE	Adoption	$+19.6$	0.0	$+17.9$	-21.4	$+19.6$	-10.7	$+8.9$	-19.6
GROUNDING	Grounded|Adopted	$+66.7$	0.0	$+73.7$	-1.8	$+57.9$	-1.8	$+68.4$	-15.8
OVERRIDE	Recovery	$+1.8$	-12.5	$+16.1$	-25.0	$+23.2$	-3.6	$+28.6$	-23.2

No-memory baselines by diagnostic (Qwen3.5-9B / Qwen3.5-27B / Gemma-4-26B / Gemini-3-Flash): DECOY 24.2/35.5/25.8/53.2%, UPTAKE 3.6/26.8/16.1/35.7%, GROUNDING 0.0/1.8/1.8/17.5%, OVERRIDE 14.3/25.0/3.6/23.2%.

Table 5: **Persistent-conflicting damage is invariant to memory placement.** Same memory text, different position; Δ vs. no-memory baseline on MemTrapBench (231 long-horizon tasks).

Model	Prompt header Δ	Observation footer Δ
Qwen3.5-9B	-9.5 pp	-10.0 pp
Qwen3.5-27B	-21.2 pp	-20.8 pp
Gemma-4-26B	-10.8 pp	-10.4 pp

MiniWoB++ shows null effects, the trap emerges around 5–7 steps on WebArena, and stabilizes on the 15-step MemTrapBench. The trap is therefore a propagation phenomenon, not a single-step misclick, and is consistent with reports that LLM self-correction is most fragile precisely when the agent must integrate feedback over many steps [11, 13].

Table 6: **Persistent conflicting damage scales with median trajectory length.**

Benchmark	Med. steps	Qwen3.5-9B	Qwen3.5-27B	Gemma-4-26B
MiniWoB++ (n=36)	3	0.0	-2.9	$+3.9$
WebArena 77 (n=67–77)	7	-9.1	-20.8	-4.5
MemTrapBench (n=231)	15	-9.5	-21.2	-10.8

Conditioning on failure beats always injecting. Supply-side proxies (relevance, timing) describe how memory was retrieved, not whether the model needs it. We benchmark against the *schedule oracle*: the best fixed injection schedule (early/persistent/late helpful) chosen per task with oracle access to per-task outcomes, an upper bound on any non-adaptive policy. Table 7: *retry-on-fail* runs once with no memory and retries with persistent helpful memory only on failure. It outperforms always-helpful injection, closes 96–101% of the schedule-oracle gap on MemTrapBench, and matches the oracle on Qwen3.5-27B WebArena. Retry without memory recovers only $+2$–3 pp (Appendix G), so the gain comes from memory conditioned on failure rather than from sampling diversity. A naïve alternative is to prefix the persistent memory with a "verify before acting" warning. On Qwen3.5-27B/MemTrapBench this reduces conflicting damage by $+17.2$ pp but also reduces the helpful gain by -11.0 pp, making the agent uniformly less compliant; retry-on-fail is selective by construction and is therefore strictly preferable. Conceptually this echoes self-consistency [34] and self-refine [23]. This is diagnostic evidence, not a deployment recipe.

Robustness of LLM-mediated measurements. Three pipeline components are LLM-mediated, and each has a separate check. (i) *RCR judging*: two authors manually reviewed all judged cases and agreed with the Gemini-3-Flash labels in 100% of cases (Cohen's $\kappa = 1.00$), so RCR and DPC are identical under human labels; this also addresses the concern that the judge is one of the evaluated models. The non-LLM entry-divergence subset gives the same qualitative and statistical conclusion (Appendix B). (ii) *MemTrapBench construction*: Claude Opus 4.7 drafts tasks and memories from fixed templates, but every task and memory is reviewed by hand and executed under no memory and persistent conflicting memory to confirm that the correct path is feasible and the trap behaves

Table 7: **Retry-on-fail beats always-helpful injection.** Absolute SR with Δ vs. no memory.

	WebArena	MemTrapBench	
Policy	Qwen3.5-27B	Qwen3.5-27B	Gemma-4-26B
No memory	40.3%	22.5%	12.1%
Always persistent helpful	41.6% (+1.3)	53.2% (+30.7)	42.0% (+29.9)
Retry-on-fail (helpful)	**50.6% (+10.4)**	**53.7% (+31.2)**	**43.2% (+31.1)**

as intended. (iii) *Grounding rewrites*: the plausible but nonexistent interface labels proposed by the LLM are verified by exact matching against the rendered HTML, with zero matches. Residual risks remain: LLM-drafted memories may share stylistic regularities that hand review does not remove, and the perfect RCR agreement partly reflects that our helpful and conflicting recommendations prescribe clearly different actions, which more ambiguous memories would not.

Limitations. Three limitations bound this study. First, our long-horizon evidence comes only from web-browsing agents (WebArena, MemTrapBench); we do not test whether the compliance trap and the E-P-R gate structure transfer to other long-horizon settings such as terminal-use agents (shell, coding) or embodied agents (household robotics, simulated navigation), where action space, observation modality, and failure feedback differ in ways that may reshape the gates. Second, our findings identify clear leverage points (Entry as the multiplicative gate, the low post-compliance floor, and helpful-side gate attrition), but we do not use them to propose or train a new memory paradigm. We characterize the trap rather than build a controller, retriever, or training-time intervention that exploits it. Translating the diagnosed asymmetry into a working memory-management design is left to future work; we present retry-on-fail and the verification prompt only as policy probes. Third, our controlled memories establish the conditional failure mechanism, while our only frequency estimate (11–13% conflicting memories) comes from a 100-memory audit of one automatic pipeline on WebArena; the expected deployment risk under other memory systems remains to be measured.

6 Conclusion

We study agent memory from the consumption side, asking how long-horizon agents process retrieved memory across a multi-step trajectory rather than how it is written, stored, or retrieved. We introduce Entry–Propagation–Recovery (E-P-R) as a trajectory-level diagnostic, instantiate it on WebArena and on MemTrapBench (a controlled benchmark we build for this purpose), and identify a *compliance trap* in which similar compliance rates and a similar post-compliance floor translate into greater absolute damage for agents with higher no-memory baselines. These experiments establish the mechanism conditional on a conflicting memory entering the context; how often such memories arise depends on the retrieval pipeline. More broadly, our results suggest that the safety and reliability of memory-augmented agents depend on how memory is consumed during a trajectory, not only on what is retrieved into context. We hope E-P-R and MemTrapBench help future work design and evaluate memory systems that account for this consumption process.

Acknowledgments and Disclosure of Funding

This work was supported by the Office of Naval Research (ONR) under grant N00014-23-1-2641.

Competing interests. The authors declare no competing interests.

References

[1] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton,

Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[2] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[3] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

[4] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

[5] Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024.

[6] Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

[7] Gemma Team, Google DeepMind. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[8] Lovisa Hagström, Youna Kim, Haeun Yu, Sang goo Lee, Richard Johansson, Hyunsoo Cho, and Isabelle Augenstein. CUB: Benchmarking context utilisation techniques for language models. *arXiv preprint arXiv:2505.16518*, 2025.

[9] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[10] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. CogAgent: A visual language model for GUI agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[11] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[12] Tomoyuki Kagaya, Thong Jing Yuan, Yuxuan Lou, Jayashree Karlekar, Sugiri Pranata, Akira Kinose, Koki Oguri, Felix Wick, and Yang You. RAP: Retrieval-augmented planning with contextual memory for multimodal LLM agents. *arXiv preprint arXiv:2402.03610*, 2024.

[13] Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024.

[14] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[15] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*, 2023.

[16] Thibault Le Sellier De Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. The BrowserGym ecosystem for web agent research. *Transactions on Machine Learning Research (TMLR)*, 2025.

[17] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[18] Eric Li and Jim Waldo. WebSuite: Systematically evaluating why web agents fail. *arXiv preprint arXiv:2406.01623*, 2024.

[19] Genglin Liu, Shijie Geng, Sha Li, Hejie Cui, Sarah Zhang, Xin Liu, and Tianyi Liu. WebCoach: Self-evolving web agents with cross-session memory guidance. *arXiv preprint arXiv:2511.12997*, 2025.

[20] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics (TACL)*, 12:157–173, 2024.

[21] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[22] Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[23] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[24] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[25] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[26] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

[27] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, 2023.

[28] Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Ben Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal

Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemí Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan. Discovering language model behaviors with model-written evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

[29] Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston, Shauna Kravec, Timothy Maxwell, Sam McCandlish, Kamal Ndousse, Oliver Rausch, Nicholas Schiefer, Da Yan, Miranda Zhang, and Ethan Perez. Towards understanding sycophancy in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[30] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[31] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[32] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[33] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), 2024.

[34] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[35] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[36] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[37] Jerry Wei, Da Huang, Yifeng Lu, Denny Zhou, and Quoc V. Le. Simple synthetic data reduces sycophancy in large language models. *arXiv preprint arXiv:2308.03958*, 2023.

[38] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

[39] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[40] Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts LLM agents: An empirical study of experience-following behavior. *arXiv preprint arXiv:2505.16067*, 2025.

[41] Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*, 2024.

[42] Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. Knowledge conflicts for LLMs: A survey. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

[43] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*, 2025.

[44] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[45] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in GPT-4V. *arXiv preprint arXiv:2310.11441*, 2023.

[46] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[47] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[48] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[49] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[50] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a generalist web agent, if grounded. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[51] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[52] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

A Experimental setup and reproducibility

This appendix contains the implementation details needed to reproduce the controlled-injection protocol: tasks, models, decoding, statistics, and how the helpful and conflicting memory passages were authored and audited.

Task pools. The WebArena pool consists of 684 non-map tasks across five self-hosted websites (an e-commerce platform, an e-commerce administration panel, a forum, a code repository, and an encyclopedia); 128 map tasks are excluded since they require geolocation capabilities orthogonal to memory consumption. The 77-task memory-sensitive subset is built by retaining tasks whose binary outcome changes between any pair of {no-memory, early-helpful, early-conflicting} on Qwen3.5-9B or Qwen3.5-27B. We acknowledge that this filter inflates effect sizes on the subset relative to the full distribution, and triangulate against this with the full-distribution memory-format study (Appendix C) and with MemTrapBench, which is constructed without outcome-based selection.

Models and decoding. All five models share the same BrowserGym/WebArena interface [16], AXTree observation, and the predefined ReAct [48] action space (click, type, scroll, navigate, etc.); each episode allows up to 15 steps, and the AXTree is truncated to 6,000 characters. The four open-weight models—**Qwen3.5-9B**, **Qwen3.5-27B** [44], **Gemma-4-E4B-it**, and **Gemma-4-26B-A4B-it** [6]—are served by vLLM [15] v0.19.0 on NVIDIA A6000 (48GB) GPUs with bfloat16, prefix caching, max context 8,192 tokens, max generation 2,048 tokens per step. Decoding is greedy throughout (`temperature=0`, `top_p=1.0`). Tensor parallelism is 1 for Qwen3.5-9B and 2 for Qwen3.5-27B and both Gemma-4 variants. **Gemini-3-Flash** (`gemini-3-flash-preview`) is queried via the Google Generative AI API with its default sampling settings (no temperature override). A cross-family check on Gemma-3-{4B,12B,27B}-it [7] appears in Appendix F.

Paired comparison and statistical testing. For each (model, task, condition) cell we compare the memory-augmented trace to the matching no-memory trace under the same environment seed; for the open-weight models, greedy decoding makes outcomes deterministic up to negligible vLLM kernel-level non-determinism, so the no-memory baseline is computed once per task; Gemini-3-Flash outcomes can additionally vary through API-side sampling. All pairwise contrasts use sign-flip permutation tests with 10,000 iterations, with Holm–Bonferroni correction across families of more than two conditions; confidence intervals are paired bootstrap percentile intervals (10,000 resamples). We do not pool across models because compliance rates and baselines differ. Two trajectories are deemed to diverge at the first step where the emitted action string disagrees on either the action verb or the targeted bid; `type`-text differences alone do not count as divergence, since memory-induced harm is dominated by element-level disagreement.

Memory authoring and audit. Each task receives a parallel set of short memory passages so that the experiment isolates how the memory is used, not which memory is retrieved. Helpful memory gives task-correct guidance; conflicting memory gives a plausible but task-wrong recommendation that still references real UI elements on the same site; cross-task control memory is a helpful memory drawn from a different task on the same website (matched in vocabulary and length, but not relevant to the current goal). All memories use the same DO/DON'T template. WebArena memories are hand-authored. MemTrapBench memories are drafted by Claude Opus 4.7 from per-family blueprints (Section 3.3) and reviewed by hand. To rule out stylistic confounds, we audit the matched helpful/conflicting pairs across all 77 WebArena tasks: matched mean character length (619 vs. 622, ratio 1.00), matched mean word count (103.7 vs. 100.8, ratio 0.97), identical structural elements (all 77 pairs contain DO/DON'T/NOTE sections), and matched confidence-language frequency (0.03 instances per task on both sides). The full-distribution study (Appendix C) further checks that the same direction holds across four memory formats from raw reflection text to structured DO/DON'T.

Memory examples. The three abbreviated passages below are for one shopping task (adding a product review). They are illustrative rather than a separate experimental condition.

> **Helpful (DO).** Navigate to the product page first using the search bar, then scroll to the review section and click "Write a Review." Fill in all required fields (rating, title, text) before submitting. **(DON'T.)** Do not try to submit a review from the product listing page; you must be on the individual product page. Do not skip the star rating field.

> **Conflicting (DO).** Go to your account dashboard and find the "My Reviews" section to write a new review. Select the product from your order history and submit from there. **(DON'T.)** Do not navigate to the product page directly; this approach is unreliable and the review form often fails to load.

> **Cross-task control (DO).** To update your shipping address, navigate to Account Settings, then Address Book. Click "Add New Address" and fill in all fields. **(DON'T.)** Do not try to change the address during checkout; changes made there are not saved to your profile.

LLM use disclosure. LLM tools were used to assist with writing and editing prose. They were not used to generate any experimental result, score any outcome, or author the curated WebArena memories. The MemTrapBench tasks and helpful/conflicting/cross-task memories were drafted by Claude Opus 4.7 and reviewed by hand. The four online-generated WebArena memory formats (Appendix C) and the MemTrapBench G^- ungroundability rewrite (Appendix E) are also LLM-generated. Code, MemTrapBench tasks, memory passages, and the agent prompt are released at `https://github.com/Schuture/Compliance-Trap`.

B E-P-R metrics and compliance subsets

This appendix gives formal definitions for the trajectory-level metrics used in the main text and compares the two compliance subsets.

Entry, Propagation, Recovery. Entry asks whether the agent's first post-injection action follows the memory rather than the no-memory trace; for early and persistent injection that is step 1, for late injection it is the first step after injection. Propagation asks whether an initial memory effect remains local or reshapes the rest of the trajectory; we report the persistent-to-early effect ratio as a schedule-level propagation factor (above 1 means repeated exposure amplifies). Recovery asks whether the agent later returns to a baseline-like trajectory after memory-induced divergence. We separately track *trajectory-level* recovery (re-aligning actions) and *task-level* recovery (final success), because the agent can re-align actions and still fail the task if the early memory-induced action is irreversible (Appendix E) [11, 13].

Compliance subsets and DPC. The headline subset $\mathcal{C}^{\mathrm{RCR}}$ (Section 3.2) is the LLM-judge subset used in Tables 2 and 3. For robustness we also report results on $\mathcal{C}^{\mathrm{EDR}}$, the coarser action-divergence subset (any step-1 action string different from the no-memory action). $\mathcal{C}^{\mathrm{RCR}}$ is a strict subset of $\mathcal{C}^{\mathrm{EDR}}$ by construction. The damage-per-compliance metric is the paired conditional $\mathrm{DPC}(\mathcal{C}) = \frac{1}{|\mathcal{C}|}\sum_{i\in\mathcal{C}}(y_i^0 - y_i^c)$; pairing on $\mathcal{C}$ removes the selection-bias confound of comparing an all-task baseline to compliant-subset conflict success.

Four-gate decomposition. The helpful lift can be expressed as a multiplicative gate chain $\Delta_{\text{helpful}} = P(\text{novel}) \cdot P(\text{adopt} \mid \text{novel}) \cdot P(\text{ground} \mid \text{adopt}) \cdot P(\text{lift} \mid \text{ground})$. WebArena conflicting memories are hand-authored to be plausible and executable, so they typically clear novelty and grounding and turn the terminal lift into damage; this is the regime in which Table 2's WebArena DPC numbers are reported. The grounding gate is tested causally on the MemTrapBench 2×2 DECOY factorial (Appendix E). The main text uses the coarser three-phase form throughout.

RCR judging procedure. For each (model, task), an LLM judge (Gemini-3-Flash, temperature 0) is given (i) the conflicting memory's `DO` excerpt, (ii) the agent's no-memory step-1 action and starting URL, and (iii) the agent's persistent-conflicting step-1 action with the URL before and after that action. The judge returns YES, NO, or UNCLEAR with a one-sentence reason. YES requires that the conflicting step-1 action plausibly executes the first primitive of the memory's `DO` (matching via the action verb, the post-step URL transitioning toward the memory-recommended page, or a memory-specified `fill` value), and that this action is more aligned with the memory than the paired no-memory baseline step-1. NO is returned when the conflicting step-1 either matches the no-memory baseline or moves to a clearly unrelated location. The judge is explicitly told that URL invariance after a click is not by itself evidence of failure (admin sidebars often expand a hover menu without changing the URL). $\mathcal{C}^{\mathrm{RCR}} = \{\text{YES}\}$ is the strict subset used in the main text; a looser $\{$YES, UNCLEAR$\}$ subset gives the same sign and CI pattern.

EDR robustness. Table 8 reports the same paired DPC on $\mathcal{C}^{\mathrm{EDR}}$. We do not bid-normalise the action strings here; bid-level noise inflates EDR, which is exactly why RCR is the headline metric. The sign and significance pattern matches Table 2: only Qwen3.5-27B's CI excludes zero, and the other three models show smaller positive point estimates whose CIs include zero.

Model	$\lvert\mathcal{C}^{\mathrm{EDR}}\rvert$	$\mathrm{SR}^{0}_{\mathcal{C}^{\mathrm{EDR}}}$	$\mathrm{SR}^{c}_{\mathcal{C}^{\mathrm{EDR}}}$	Paired DPC [95% CI] [p]
Gemma-4-E4B	56	25.0%	19.6%	5.4 pp [−5.4, 16.1] [.56]
Qwen3.5-9B	65	29.2%	20.0%	9.2 pp [−1.5, 20.0] [.17]
Gemma-4-26B	57	28.1%	22.8%	5.3 pp [−3.5, 14.0] [.45]
Qwen3.5-27B	66	43.9%	21.2%	**22.7 pp [+12.1, 34.9] [<.001]**

Table 8: EDR-subset paired DPC. Same sign and significance pattern as Table 2; the cleaner RCR subset (main text) gives a slightly deeper trap on Qwen3.5-27B (+25.5 pp) than the EDR upper-bound proxy (+22.7 pp).

C WebArena: extended analyses

This appendix collects the WebArena tables that the main text refers to: who recovers from conflicting memory and who does not (Tables 9, 11); how much the persistent schedule amplifies the early effect (Table 10); how much benefit a per-task schedule oracle would unlock (Table 12); a same-website cross-task control (Table 13); whether memory advice can be grounded in the current page (Table 14); and whether the agent's URL-level commitment differs by memory type (Table 15). We then ask two further questions: does conflicting memory clear the entry gate *because* it is novel rather than redundant (Table 16), and does the trap survive when memory is rewritten as raw reflections, procedural workflows, or quality-gated DO/DON'T on the full 684-task distribution (Table 17).

Table 9: **Recovery rates under persistent memory conditions.** Recovery rate is the fraction of diverged trajectories where the agent re-aligns with baseline at any subsequent step. Qwen recovery collapses to ∼13% while Gemma-4 maintains ∼31%, an architecture-level rather than scale-level effect.

Condition	Gemma-4-E4B	Qwen3.5-9B	Gemma-4-26B	Qwen3.5-27B
Persistent helpful	25.8%	28.4%	27.0%	40.3%
Persistent conflicting	32.4%	12.7%	31.4%	13.0%
Late conflicting	24.6%	26.2%	20.0%	40.7%

Table 10: **Propagation factors and action history momentum across models and memory types.** The propagation factor is the ratio of persistent to early effect sizes, $\mathrm{PF} = \Delta^{\text{paired}}_{\text{persistent}} / \Delta^{\text{paired}}_{\text{early}}$, computed on the same paired-pool subset as Figure 3; values above 1 indicate amplification by persistent exposure. Action-history momentum measures resistance to late injection, $\mathrm{AHM} = 1 - \Delta^{\text{paired}}_{\text{late}} / \Delta^{\text{paired}}_{\text{early}}$, so values near 1 indicate that adding the same memory after history has formed has near-zero marginal effect (full resistance); values near 0 indicate no resistance, i.e. late injection still has the same effect as early injection. Dashes indicate cases where $|\Delta_{\text{early}}| < 2$ pp so the ratio is unstable. The largest amplification (4.8×) occurs for Gemma-4-26B under conflicting memory.

	Propagation Factor		Action History Momentum	
Model	Helpful	Conflicting	Helpful	Conflicting
Gemma-4-E4B	4.1	—	0.70	—
Qwen3.5-9B	2.9	1.2	∼1.0	0.32
Gemma-4-26B	2.0	4.8	<0	∼0
Qwen3.5-27B	—	2.4	∼1.0	0.33

Why does conflicting memory enter so easily? A useful diagnostic is to split tasks by whether the no-memory trace already commits to a strategy. On weak-preference tasks, conflicting memory is adopted 25–41 pp more often than helpful memory because helpful memory often duplicates what the model would already do, while conflicting memory always recommends something new (Table 16). On strong-preference tasks the gap collapses. Conflicting memory clears the entry gate easily because, by construction, it is novel.

Does the trap survive different memory formats? The 77-task subset is curated; a reviewer might ask whether the trap is created by selecting unusually memory-sensitive tasks or by the specific

Table 11: **Recovery rates across all model-condition combinations.** Recovery rate is the fraction of diverged trajectories where the agent re-aligns with baseline at any subsequent step. Persistent conflicting memory collapses recovery in the Qwen family (roughly 13%) while Gemma-4 maintains roughly 31%. Under less adversarial conditions (early, late, helpful), recovery rates are higher and more similar across families.

Condition	Gemma-4-E4B	Qwen3.5-9B	Gemma-4-26B	Qwen3.5-27B
Early helpful	23.8%	22.4%	28.1%	28.6%
Early conflicting	25.7%	22.9%	33.8%	22.4%
Persistent helpful	25.8%	28.4%	27.0%	40.3%
Persistent conflicting	32.4%	12.7%	31.4%	13.0%
Late helpful	30.5%	35.5%	33.9%	43.6%
Late conflicting	24.6%	26.2%	20.0%	40.7%

Table 12: **Schedule oracle versus uniform injection for helpful memory.** The schedule oracle selects the best injection schedule per task under perfect foreknowledge of per-task success. Uniform persistent injection captures only a fraction of available benefit; the "Wasted" column is the 9.6–11.8 pp lost to suboptimal policy. Per-condition denominators ($n = 70$–76) differ from Figure 3 because the schedule oracle is computed only on tasks with all four schedules valid.

Model	No-memory	Persistent	Oracle	Oracle Δ	Wasted
Qwen3.5-9B	31.5% ($n = 73$)	37.0% (+5.5)	46.6% (+15.1)	+15.1 pp	9.6 pp
Qwen3.5-27B	42.9% ($n = 70$)	42.9% (+0.0)	52.9% (+10.0)	+10.0 pp	10.0 pp
Gemma-4-26B	32.9% ($n = 76$)	35.5% (+2.6)	47.4% (+14.5)	+14.5 pp	11.8 pp

DO/DON'T template. We address both concerns simultaneously by sweeping four memory formats on the full 684-task non-map distribution with online-retrieved memories: raw reflection text (M1, Reflexion-style [31]), procedural workflow (M2, AWM-style [35]), contrastive DO/DON'T (M3), and quality-gated M3 (C1, the relevance-gated condition referenced in Section 4.1). All four formats produce the same direction (Table 17). More structured formats amplify both helpful and harmful effects (M1 $<$ M2 $<$ M3), so the format acts as a gain control rather than a mechanism switch. Quality gating (C1) does not solve the problem: a memory can be well-written and still be harmful given the current model and task state. The trap is therefore not an artifact of the DO/DON'T template, and the curation of the 77-task subset only sharpens an effect that is already present at the full-distribution level.

D MemTrapBench: design and accounting

This appendix gives MemTrapBench's design philosophy, the structural templates used to write tasks, and the per-task accounting. We position MemTrapBench against existing benchmarks and explain how trap construction differs from outcome-based selection.

What MemTrapBench is for. MemTrapBench is a mechanism probe for long-horizon memory consumption, not a coverage benchmark to replace WebArena, Mind2Web [4], WebShop [46], or AgentBench [21]. Each task is built to isolate one part of the E-P-R chain while keeping the environment controlled. All tasks are served as standalone browser environments through BrowserGym [16]; interactive elements use `<button>` semantics and ARIA labels so an LLM consuming the AXTree sees all actionable targets, including the decoy. Crucially, traps are defined by construction rather than selected by downstream model outcome, which is the key difference from the WebArena 77-task subset.

Diagnostic families and structural templates. The four diagnostic families introduced in Section 3.3 (DECOY / UPTAKE / GROUNDING / OVERRIDE) are realized via eight structural templates. Each template encodes the trap in a structurally distinct way so that observed effects cannot be a single-layout artifact:

- **S1** Filter-detail-form-confirm (16-task pilot)

Table 13: **Negative control: persistent irrelevant memory.** Irrelevant memory is a helpful memory drawn from a different task template on the same website. The near-zero deltas confirm that the harm of conflicting memory is content-specific, not an artifact of additional context length or generic noise in the observation.

Model	No-memory SR	Persistent irrelevant Δ	Persistent conflicting Δ
Qwen3.5-9B	30.7%	$+1.3$ pp	-9.1 pp
Qwen3.5-27B	43.1%	-2.6 pp	-20.8 pp
Gemma-4-26B	32.5%	$+4.5$ pp	-4.5 pp

Table 14: **Grounding: mean groundability at injection time and at step 3 across conditions.** Groundability measures whether the agent can map memory advice to elements on the current page (range 0–1). At step 0 (early/persistent injection), groundability is uniformly low ($\sim$0.10) because the agent has not yet navigated to the task-relevant page. Late injection benefits from higher groundability (0.35–0.54) because the agent has already navigated partway. Helpful memory generally achieves higher step-3 groundability than conflicting memory because correct strategies align with reachable page states.

		At injection		At step 3	
Model	Condition	Helpful	Conflicting	Helpful	Conflicting
Qwen3.5-9B	Persistent	0.10	0.13	0.52	0.53
	Late	0.47	0.35	0.56	0.36
Qwen3.5-27B	Persistent	0.10	0.13	0.52	0.46
	Late	0.54	0.36	0.56	0.34
Gemma-4-26B	Persistent	0.09	0.13	0.60	0.44
	Late	0.51	0.35	0.60	0.35

- **S2** Multi-step wizard with skip-to-defaults trap
- **S3** Report builder with defaults-based quick-action trap
- **S4** Cross-reference (page A $\rightarrow$ page B) with ignored-lookup trap
- **S5** Bulk action with per-item config collapsed to a single shortcut
- **S6** Configure-then-navigate-to-verify with skipped verification
- **S7** Multi-tab form with partial submit trap
- **S8** Nested modal flow with force-option bypass

Each task ships with a matched {helpful, conflicting, cross-task control} memory trio in the same DO/DON'T format as our WebArena memories, with lengths matched to 500–1200 characters. Conflicting memory inverts the recommendation while referencing the same real UI elements; cross-task control memory describes a different task on the same site type. Table 18 gives one concrete task per diagnostic family.

Per-task accounting. The 247 long-horizon tasks (16 S1 pilot + 231 S2–S8) each receive all 8 temporal-injection conditions on all 3 open-weight models (5,928 trials), plus the 4 persistent conditions on Gemini-3-Flash. Zero tasks were dropped during evaluation (Table 19). The S2–S8 set drives the main results in Section 4.4; the S1 pilot is reported separately in Table 20 and replicates the trap with a ceiling caveat on persistent helpful (pilot baselines are higher, so helpful gain saturates).

E Causal probes of the trap

This appendix contains the three probes that turn E-P-R from a descriptive story into a causal account: per-trace gate annotation, a 2×2 factorial that toggles Entry and Grounding orthogonally, and a memory-placement ablation that toggles the delivery channel.

Table 15: **Commitment strength under late injection (URL-level).** Commitment measures the fraction of post-injection steps where the agent remains on URLs consistent with its pre-injection trajectory; switch rate measures redirections. Stronger models show higher commitment under conflicting than helpful memory, consistent with asymmetric history-conditioned inertia.

	Commitment (%)		Switch rate (%)	
Model	Helpful	Conflicting	Helpful	Conflicting
Gemma-4-E4B	45.5	50.0	27.3	21.4
Qwen3.5-9B	62.5	46.7	12.5	20.0
Gemma-4-26B	33.3	84.6	33.3	0.0
Qwen3.5-27B	50.0	69.2	50.0	23.1

Table 16: **Adoption rates by baseline preference strength and memory type.** Weak-preference tasks are those where the model's baseline trajectory does not strongly commit to any strategy. In these tasks, conflicting memory achieves 25–41 pp higher adoption than helpful memory because helpful memory often recommends what the model would do anyway (redundant), while conflicting memory always recommends something novel.

		Weak preference			Strong preference		
Model	n	Helpful	Conflicting	Δ	Helpful	Conflicting	Δ
Gemma-4-E4B	70	62.3%	88.7%	+26.4	11.8%	29.4%	+17.6
Qwen3.5-9B	68	62.3%	86.9%	+24.6	42.9%	57.1%	+14.3
Gemma-4-26B	74	43.8%	78.1%	+34.4	10.0%	10.0%	0.0
Qwen3.5-27B	68	50.0%	90.6%	+40.6	75.0%	75.0%	0.0
Mean		*54.6%*	*86.1%*	*+31.5*	*34.9%*	*42.9%*	*+8.0*

E.1 Per-trace gate annotation

Why we annotate per-trace. The aggregate compliance numbers in Table 2 tell us *that* agents follow conflicting memory, but not whether the same gates that bottleneck helpful gain also gate conflicting damage. To answer this we annotate every memory-conditioned trace against its paired no-memory trace with four trajectory signals—*adopted@1*, *diverged*, *recovered*, and *success*—and tag each task by its diagnostic family (Decoy/Uptake/Grounding/Override). The annotation covers 3 models $\times$ 231 S2–S8 tasks $\times$ 7 non-baseline conditions = $4{,}851$ traces.

Gate-chain reading. Table 21 compares the four gates in the helpful direction against their conflicting analogues. Helpful memory only improves success when it passes every gate; conflicting memory passes the same gates but flips the terminal effect into damage. The product $\Delta_{\text{helpful}} = P(\text{novel}) \cdot P(\text{adopt} \mid \text{novel}) \cdot P(\text{ground} \mid \text{adopt}) \cdot P(\text{lift} \mid \text{ground})$ reproduces the observed helpful delta on every WebArena model.

Per-trace annotation procedure. To make E-P-R more than a descriptive story, we annotate every memory-conditioned trace against its paired no-memory trace. Each trace is labeled with four trajectory signals: whether the first memory-visible action changes (*adopted@1*), whether the trajectory diverges at any point (*diverged*), whether it later returns to a baseline-like path (*recovered*), and whether the final task succeeds. We additionally tag each task by its diagnostic category (trap, uptake, grounding, override). The annotation covers 3 models $\times$ 231 S2–S8 tasks $\times$ 7 non-baseline conditions = $4{,}851$ traces; the full per-trace annotation is in `analysis/gate_annotation_results.json`, and Table 22 gives the aggregate pass rates.

The annotations make an important distinction visible. Many traces recover at the action level: after a memory-induced detour, the agent often takes actions similar to the no-memory baseline. But task success does not always recover. In web tasks an early wrong click, wrong form value, or wrong navigation can make later correction too late. Recovery must therefore be measured separately at the trajectory level and at the final-outcome level. The annotations also explain why step-1 adoption can be low even when memory has a large final effect: MemTrapBench memories describe a strategy

Table 17: **Success-rate deltas across four memory formats on the full 684-task distribution with online-retrieved memories (Qwen3.5).** All four formats produce the same direction; more structured formats amplify both directions (M1 < M2 < M3). C1 (quality-gated M3) hurts both models, so gating retrieval does not address the consumption asymmetry.

Format	Description	9B Δ	27B Δ	9B p	27B p
M1	Raw reflection (Reflexion-style)	-1.3 pp	$+0.7$ pp	0.121	0.265
M2	Procedural workflow (AWM-style)	-1.2 pp	$+1.0$ pp	0.250	0.114
M3	Contrastive DO/DON'T	-1.9 pp	$+1.7$ pp	**0.046**	**0.011**
C1	Quality-gated M3 (score$\geq$3)	-2.3 pp	-1.7 pp	0.103	0.183

p-values from 10,000-iteration sign-flip permutation tests against the no-memory baseline on the full 684-task distribution. Only M3 reaches $p < 0.05$ on both models.

Table 18: **One concrete MemTrapBench task per diagnostic family.** The conflicting memory's DO line specifies a plausible but wrong action that, when adopted at first exposure, isolates one E-P-R gate.

Diagnostic	Site / task	Conflicting memory DO line (abbreviated); (correct path)
DECOY	Admin / refund	Click the prominent orange "Refund" button and confirm; the engine routes the refund automatically regardless of destination. (Correct: open the destination dropdown and pick the goal-specified destination first.)
UPTAKE	Admin / 2FA bulk action	Click the Bulk Actions control on All Users and pick "Enable 2FA". (Correct: open the Advanced toggle in Roles to apply role-scoped 2FA only to the Admin role.)
GROUNDING	Admin / customer Q4 orders	Read the Orders value from the dashboard "Top Customers" widget. (Correct: navigate into the customer detail page and filter orders by quarter; the dashboard widget does not show Q4-only orders.)
OVERRIDE	DevOps / close issue	Click the red "Close issue" button; any dropdown nearby is optional. (Correct: choose the goal-specified resolution label first; the environment returns an enforcement error if the button is clicked without a label, which the agent must not ignore.)

rather than a single first action, so memory enters over several steps once the agent reaches the part of the environment where the advice becomes actionable.

Dissociating trajectory recovery from task-level recovery. A striking pattern in Table 22 is that *recovered* is similar between persistent helpful (69–78%) and persistent conflicting (45–61%) — both are high — yet final success differs by 35–52 percentage points. The agent is demonstrably capable of returning to baseline-like actions after memory-induced divergence, but this trajectory-level recovery does not translate into task-level recovery because the damage from early wrong actions (following conflicting memory) is often irreversible in the environment (wrong form submitted, wrong modal closed, etc.). Recovery is therefore a measurable capability that nonetheless fails to protect task outcomes under persistent conflicting exposure — consistent with the main-text narrative that conflicting memory's damage compounds through the trajectory cascade, and with prior critical accounts of LLM self-correction [11, 13].

E.2 2×2 factorial on the DECOY diagnostic

Why a factorial. The per-diagnostic results in Table 4 are observational: each diagnostic exercises one phase of the E-P-R chain by construction, but within a fixed diagnostic we cannot rule out that one memory change is simultaneously perturbing multiple gates. We therefore run a 2×2 factorial that toggles Entry and Grounding orthogonally on 32 DECOY tasks per cell (16 DECOY-Admin + 16 DECOY-Ecommerce), with everything else held fixed (task, persistent schedule, model, decoding). Entry is varied by E^+ (single target memory) vs. E^- (target plus three same-format conflicting memories drawn from sibling trap tasks; the agent must semantically match its task scope to one of the four). Grounding is varied by G^+ (original UI labels appearing in the AXTree) vs. G^-, in which a controlled LLM rewrite replaces every UI label, color, and position phrase with a plausible non-existent alternative; audits confirm zero rewrite-label hits in the rendered HTML. Recovery is

Table 19: **MemTrapBench task accounting.** All 247 long-horizon tasks receive all 8 temporal-injection conditions on the 3 open-weight models. Zero tasks were dropped during evaluation.

Stage	Count
Tasks designed (S1 pilot + S2–S8 structurally-designed)	16 + 231 = 247
Tasks with memory trios authored	247
Tasks evaluated on Qwen3.5-9B (all 8 conditions)	247
Tasks evaluated on Qwen3.5-27B (all 8 conditions)	247
Tasks evaluated on Gemma-4-26B-A4B (all 8 conditions)	247
Total trials	5,928
Tasks dropped (environment failure, parser error, timeout)	0
Tasks analyzed in Table 3 main (S2–S8)	231
Tasks analyzed in Table 20 below (S1 pilot)	16

Table 20: **S1 pilot (16 tasks) replicates the compliance trap**, with a ceiling caveat on persistent helpful. Baseline SRs on the pilot are higher (25–44%) than on the main 231-task set (11–23%), so Qwen3.5-9B's pH saturates at the pre-existing ceiling ($+0$) where the main set shows $+25.5$. All pC drops are -18.8 to -31.2 pp, confirming the harm direction.

Model	Base	Early H	Early C	Pers. H	Pers. C	Pers. Irrel.	Late H	Late C
Qwen3.5-9B	37.5%	−12.5	−12.5	+0.0	−31.2	−6.2	−12.5	−18.8
Qwen3.5-27B	43.8%	+6.2	+6.2	+6.2	−31.2	+6.2	+6.2	−18.8
Gemma-4-26B	25.0%	+0.0	+0.0	+25.0	−18.8	−6.2	+25.0	−18.8

not varied here, since its natural operational gate (whether the agent recovers from outcome-level failure) is policy-level rather than prompt-level; we test recovery via retry-on-fail in Appendix G.

Three findings. Grounding has a large, significant main effect on both models (-29.7 pp on Qwen3.5-27B, -23.4 pp on Gemma-4-26B-A4B): replacing UI references with non-existent alternatives cuts per-cell damage by 23–30 pp, isolating the grounded-given-adopted gate. Entry has a significant main effect on the stronger Qwen model (-20.3 pp, CI excludes 0) and a directional but non-significant effect on Gemma-4-26B-A4B. The $E\times G$ interaction is at most 33% of the larger main effect on either model, supporting the multiplicative E-P-R decomposition. On both models, (E^-, G^-) is the best cell and (E^+, G^+) is the worst, indicating the manipulation captures a mechanism-level effect rather than a model-idiosyncratic artefact.

Helpful-side companion factorial. We mirror the same 2×2 design with persistent helpful memory on Qwen3.5-27B (Table 24). The cell-level gains are an order of magnitude smaller than the corresponding conflicting-side damages, and E and G are roughly additive ($\approx +3$ pp each, max $+6.2$ pp at (E^+, G^+)). Comparing tables, conflicting damage is gate-multiplicative (-53.1 pp at (E^+, G^+) vs. -3.1 pp at (E^-, G^-), an $\approx 17\times$ ratio), while helpful gain is gate-additive. This is the gate-chain prediction of the discussion: conflicting memory clears the early gates trivially and is bottlenecked only at the terminal lift, so closing any one gate removes most of the damage; helpful memory leaks at every gate, so the maximum gain is small to begin with and closing one gate makes proportionally less difference. We omit Gemma-4-26B-A4B from the helpful side because its per-cell helpful response is at the level of cell-to-cell noise, consistent with the small Gemma-4 helpful gains in Table 3.

E.3 Memory delivery channel

Setup. The main text reports the persistent-conflicting numbers in two prompt positions (prompt header vs. observation footer; Table 5). Here we add the persistent-helpful side and report the full $\pm$pp summary across three models.

What the helpful side adds. Conflicting damage is invariant to placement (within ±0.5 pp), so the trap is content-driven. Helpful gain, by contrast, attenuates by 4–16 pp under footer injection, most severely on Qwen3.5-27B ($+30.7 \rightarrow +14.7$ pp, a 52% reduction). The asymmetry between the

Table 21: **Gate-chain explanation of helpful–harmful asymmetry on WebArena.** Helpful memory improves success only if it passes every gate. Conflicting memory often passes the same gates but flips the terminal effect into damage. Compliance figures in the Adoption row are persistent-conflict RCR (Table 2); the EDR upper bound is 75–86%.

Gate	Helpful memory	Conflicting memory
Novelty	Non-redundant on 34–41% of tasks	Novel by construction
Adoption	Competes with the model's default plan	RCR = 63–69% under persistent conflict
Grounding	Often requires earlier navigation	Wrong-but-valid actions are often executable
Terminal effect	Positive only if it improves final success	Compliance often pushes success to a low floor

Table 22: **Per-gate pass rates on MemTrapBench S2–S8 (231 tasks per model).** adopted@1 is low across conditions because MemTrapBench memory is strategic (not step-1-action-specific) — adoption emerges over multiple steps. Divergence and recovery rates are substantial (60–85% diverged, 55–80% recovered), yet final success still scales sharply with memory type: helpful memory preserves the benefit from divergence while conflicting memory's divergence is unrecoverable. This dissociates *trajectory-level recovery* (the agent can return to baseline-like actions) from *task-level recovery* (the final outcome is salvaged): recovery is measurable but insufficient.

Model	Condition	adopted@1	diverged	recovered	success
Qwen3.5-9B	persistent_helpful	6.9%	84.8%	78.4%	36.4%
Qwen3.5-9B	persistent_conflicting	3.9%	71.9%	61.0%	1.3%
Qwen3.5-9B	persistent_irrelevant	4.3%	79.7%	72.3%	8.2%
Qwen3.5-9B	late_helpful	2.2%	84.0%	76.2%	33.8%
Qwen3.5-27B	persistent_helpful	3.9%	77.1%	69.7%	53.2%
Qwen3.5-27B	persistent_conflicting	0.4%	68.4%	55.4%	1.3%
Qwen3.5-27B	persistent_irrelevant	0.9%	64.9%	55.4%	23.8%
Qwen3.5-27B	late_helpful	0.4%	74.0%	66.2%	51.9%
Gemma-4-26B	persistent_helpful	8.7%	82.3%	60.6%	42.0%
Gemma-4-26B	persistent_conflicting	7.8%	67.5%	45.0%	1.3%
Gemma-4-26B	persistent_irrelevant	8.2%	72.7%	53.3%	13.0%
Gemma-4-26B	late_helpful	5.2%	81.4%	57.1%	36.8%

two directions matches the gate chain: conflicting memory only needs to trigger a plausible wrong action, while helpful memory must additionally clear adoption and grounding gates that prominent placement helps satisfy [20, 8].

F Boundary conditions

This appendix asks where the trap fades. Two natural boundaries: a different model family (does the trap depend on the Qwen3.5/Gemma-4 architectures?) and a shorter horizon (does it depend on having a long trajectory in which to propagate?).

F.1 Cross-family validation: Gemma-3

Question. Does the compliance-floor mechanism survive when we move to a different family of open-weight models?

Result. On 611 common valid WebArena tasks with online-retrieved memory, Gemma-3 shows the opposite scaling direction from Qwen3.5: the larger Gemma-3 model is *more* hurt by memory, while the larger Qwen model benefits (Table 26). This is consistent with the conditional mechanism: vulnerability depends on the interaction between baseline capability, compliance rate, and the post-compliance success floor. Larger models are not automatically more vulnerable in every setting; they become more vulnerable when they start from a higher baseline and still comply at a similar rate, so the same low floor creates a larger absolute drop. Architecture and instruction-tuning calibration [25, 1] can shift the baseline and the floor without removing the mechanism.

Table 23: 2×2 **factorial on the DECOY diagnostic.** Persistent conflicting memory; 32 tasks per cell per model. The (E^+, G^+) cell reproduces the standard persistent-conflicting condition; (E^-, G^-) strips memory of identifiability and parameter-level actionability.

		Qwen3.5-27B		Gemma-4-26B-A4B	
E	G	Cell SR	Label	Cell SR	Label
multi	ungroundable	**62.5%**	(E^-, G^-) max protection	**50.0%**	(E^-, G^-)
multi	groundable	31.2%	(E^-, G^+)	18.8%	(E^-, G^+)
single	ungroundable	40.6%	(E^+, G^-)	34.4%	(E^+, G^-)
single	groundable	**12.5%**	(E^+, G^+) max damage	**18.8%**	(E^+, G^+)
E main effect		-20.3 pp [-35.4, -4.7]		-7.8 pp [-23.5, $+8.1$]	
G main effect		-29.7 pp [-45.1, -13.8]		-23.4 pp [-38.7, -8.3]	
$E\times G$ interaction		$+1.6$ pp ($0.05\times$ main)		$+7.8$ pp ($0.33\times$ main)	

Table 24: **Helpful-side** 2×2 **factorial on the DECOY diagnostic** (Qwen3.5-27B; 32 tasks per cell, paired Δ vs. a same-task no-memory baseline of 65.6%). Helpful gains are bounded ($\leq +6.2$ pp at every cell) and roughly additive in E and G, in contrast to the gate-multiplicative conflicting-side damages of Table 23.

E	G	Cell SR	Δ vs. no memory
multi	ungroundable	65.6%	0.0
multi	groundable	68.8%	+3.1
single	ungroundable	68.8%	+3.1
single	groundable	**71.9%**	**+6.2**
E main effect			+3.1 pp
G main effect			+3.1 pp
$E\times G$ interaction			≈ 0

F.2 Short-horizon control: MiniWoB++

Question. Does the trap appear when the trajectory is too short for propagation and recovery to operate?

Result. Largely no. On MiniWoB++ [30] (12 tasks $\times$ 7 temporal conditions $\times$ 3 random seeds, median trajectory ≈ 2 steps), persistent-conflicting drops are only -2.9 to 0 pp and persistent-helpful is -8.3 to 0 pp (Table 27). Two reasons make this expected under E-P-R. First, baselines of 36–40% on 1–3 step tasks leave little room for the success-lift gate to operate. Second, with a median of 2 steps, persistent, late, and early injection are nearly the same event, so propagation and recovery are not differentiable. Long-horizon interaction (median 8–15 steps) is therefore a precondition for our findings; this matches the regime where coverage benchmarks such as WebVoyager [9] and TheAgentCompany [41] also report the largest absolute gaps between models.

G Retry-on-fail ablations

The main text reports retry-on-fail as a minimal demonstration that conditioning memory injection on observed first-attempt failure outperforms uniform injection. This appendix gives the two ablations that support that claim: a head-to-head comparison against fixed-schedule policies on WebArena, and a content-vs-sampling decomposition on MemTrapBench.

Lightweight policies on WebArena 77. Table 28 compares fixed-schedule helpful injection, groundability gating, and retry-on-fail against the schedule oracle. Always-persistent helpful captures only 50% and 17% of the oracle gap on Qwen3.5-9B and Qwen3.5-27B; URL-keyword groundability gating does not solve the problem and even hurts the larger model. Retry-on-fail closes 50% of the gap on Qwen3.5-9B and 133% on Qwen3.5-27B; the latter overshoot is consistent with small-sample variation, and the MemTrapBench ablation below shows the same model at 96% closure on a $\sim 3\times$-larger task pool.

Table 25: **Footer vs. prompt-header injection on MemTrapBench (231 long-horizon tasks).** Conflicting damage is invariant to channel (within ± 0.5 pp); helpful gain attenuates by 4–16 pp.

		Conflicting Δ vs. no_mem		Helpful Δ vs. no_mem	
Model	Baseline	Prompt header	**Footer**	Prompt header	**Footer**
Qwen3.5-9B	10.8%	−9.5 pp	−10.0 pp	+25.5 pp	+19.0 pp
Qwen3.5-27B	22.5%	−21.2 pp	−20.8 pp	+30.7 pp	+14.7 pp
Gemma-4-26B-A4B	12.1%	−10.8 pp	−10.4 pp	+29.9 pp	+25.5 pp

Conflicting location-invariance: max $|\Delta_{\text{header}} - \Delta_{\text{footer}}| = 0.5$ pp. Helpful attenuation under footer: 9B -6.5 pp, 27B -16.0 pp, Gemma-4-26B-A4B -4.4 pp.

Table 26: **Cross-family early injection on 611 common valid WebArena tasks.** Gemma-3 shows the opposite scaling direction from Qwen3.5: the larger Gemma-3 model is *more* hurt by memory, while the larger Qwen model benefits. At 27B parameters the two families diverge by 5.1 pp.

Family	Model	Params	B1 SR	M3 SR	Δ
Qwen3.5	9B	9B	21.4%	19.4%	−2.0 pp
	27B	27B	25.1%	26.8%	+1.7 pp
Gemma-3	4B-it	4B	11.4%	12.4%	+1.0 pp
	12B-it	12B	16.2%	16.0%	−0.3 pp
	27B-it	27B	19.9%	16.5%	−3.4 pp***

*** $p = 0.001$ (sign-flip permutation test, 10,000 iterations).

Retry-helpful vs. retry-nomem on MemTrapBench. Does the gain come from the second attempt's sampling diversity, or from the helpful memory it now sees? We separate the two by running **retry-nomem** (Attempt 1 no-memory; on failure Attempt 2 also no-memory) and **retry-helpful** (Attempt 1 no-memory; on failure Attempt 2 with persistent helpful memory) on all 231 main-set MemTrapBench tasks across three models (Table 29). Both arms use the same one-bit failure signal, so the difference retry-helpful − retry-nomem isolates the marginal benefit of memory content. Retry without memory contributes only $+2.0$ to $+2.7$ pp (sampling alone), while retry with helpful memory contributes $+19.7$ to $+31.2$ pp; the gap is $+14$ to $+28$ pp. On Qwen3.5-27B and Gemma-4-26B-A4B retry-helpful effectively closes the schedule-oracle gap (96% and 101%); Qwen3.5-9B closes 76%, consistent with the fourth-gate (success-lift) bottleneck of the discussion: smaller models extract less from the same helpful content.

Why this is a lower bound. Retry-on-fail uses the smallest possible amount of state information (whether the first attempt failed) to route memory better than always injecting. Better policies should predict memory usefulness before failure rather than after it; this is conceptually adjacent to Self-Refine [23] but applied to memory selection rather than output revision. The Qwen3.5-27B WebArena overshoot ($+2.5$ pp over the schedule oracle, Table 28) reads cleanly under these ablations: on the 77-task subset, retry-helpful benefits both from memory content and from the favorable intersection of first-failure sampling with retry; on the $\sim 3\times$-larger MemTrapBench, the same model closes 96% of the gap rather than overshooting it, confirming that the WebArena surplus reflects small-sample variation rather than a stable property.

Table 27: **MiniWoB++ short-horizon temporal injection.** Unlike WebArena and MemTrapBench, persistent-conflicting memory produces only -2.9 to 0 pp drops; persistent-helpful sometimes *hurts*. The compliance trap does not replicate at short horizons.

Model	Base	Early H	Early C	Pers. H	Pers. C	Late H
Qwen3.5-9B	38.9%	−5.6	+2.8	−8.3	+0.0	−8.3
Qwen3.5-27B	40.0%	+0.0	+1.7	−3.9	−2.9	−6.7
Gemma-4-26B-A4B	36.1%	+0.0	+0.0	+0.0	+3.9	+2.8

Table 28: **Lightweight injection policies on WebArena 77.** Each cell reports absolute success with Δ versus no-memory baseline and oracle closure. Retry-on-fail uses only the binary first-attempt outcome.

Policy	Qwen3.5-9B	Qwen3.5-27B	Signal used
No memory (baseline)	29.9% (—)	40.3% (—)	—
Always persistent helpful	35.1% (+5.2 / 50%)	41.6% (+1.3 / 17%)	none
Always late helpful†	30.2% (+0.3 / 3%)	42.7% (+2.4 / 32%)	trajectory step
Groundability gating	31.2% (+1.3 / 13%)	39.0% (−1.3 / −17%)	URL / keyword match
Retry-on-fail	**35.1% (+5.2 / 50%)**	**50.6% (+10.4 / 133%)**	first-attempt outcome
Schedule oracle	40.3% (+10.4 / 100%)	48.1% (+7.8 / 100%)	best fixed schedule per task

Closure is the policy's Δ divided by the paired schedule-oracle Δ. †Computed on the late-helpful-valid subset ($n = 70$–76); absolute deltas differ by at most ± 3 pp from Figure 3.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction (Section 1) state three contributions—the E-P-R framework, the harm mechanism of conflicting memory, and conditions for helpful memory benefit—each supported by corresponding results sections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 contains a dedicated limitations paragraph discussing benchmark scope (web-browsing agents only), the diagnostic rather than mitigation focus of the work, and the limited evidence on how often conflicting memories arise under automatic retrieval; a preceding paragraph discusses residual risks of the LLM-mediated measurements.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.

Table 29: **Retry ablations on MemTrapBench (231 long-horizon tasks).** retry-nomem alone contributes only +2.0–2.7 pp (sampling variation); retry-helpful adds +19.7–31.2 pp. The +14–28 pp gap attributes the recovery to memory content rather than to second-attempt diversity. Recovery rate (% of retry-triggered tasks that succeed on Attempt 2) is 35–40% with helpful memory vs. 3% without, a more than ten-fold lift.

	Overall SR (Δ vs. no_mem)			Recovery rate (Attempt 2 | Att. 1 failed)		
Model	no_mem	retry-nomem	retry-helpful	triggered	nomem	helpful
Qwen3.5-9B	10.8%	12.8% (+2.0)	30.5% (+19.7)	89%	2%	22%
Qwen3.5-27B	22.5%	25.1% (+2.6)	53.7% (+31.2)	78%	3%	40%
Gemma-4-26B-A4B	12.1%	14.8% (+2.7)	43.2% (+31.1)	88%	3%	35%

retry-helpful − retry-nomem gap: 9B +17.7 pp, 27B +28.6 pp, Gemma-4-26B-A4B +28.4 pp. Oracle gap closure (retry-helpful Δ / oracle Δ): 9B 76%, 27B 96%, Gemma-4-26B-A4B 101%. Denominators: 27B n =231; 9B n =211–213; Gemma-4-26B-A4B n =229.

- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper is primarily empirical. The E-P-R framework (Section 3.1) provides quantitative definitions rather than theoretical results requiring proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.

- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 describes the temporal injection protocol, memory construction, and E-P-R metrics. Section 3.3 specifies models, serving configurations, and benchmark details. Appendix A provides additional environment and infrastructure details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We use open-weight models (Qwen3.5, Gemma-4) and the open-source WebArena benchmark. The MemTrapBench tasks, all memory passages (MemTrapBench and WebArena), the agent, and the evaluation code are publicly available at `https://github.com/Schuture/Compliance-Trap`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3.3 specifies model sizes, serving parameters (context length, precision), and benchmark details (task count, timeout, max steps). Appendix A provides serving infrastructure and statistical testing details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report sign-flip permutation tests with 10,000 iterations, p-values, and 95% confidence intervals for key comparisons (Section 4). We explicitly note which comparisons reach significance and which do not. Appendix A details the testing methodology.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix A specifies GPU type (NVIDIA A6000 48GB), tensor parallelism configurations, and serving framework (vLLM with bfloat16).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work studies model behavior under controlled experimental conditions using publicly available models and benchmarks. No human subjects, private data, or dual-use concerns are involved.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 5 discusses design implications for memory-augmented agent systems, including the risk that models with higher no-memory baselines can suffer greater absolute damage from conflicting memory—a finding with direct relevance to the safe deployment of autonomous agents.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not release new models or datasets with misuse risk. We release only analysis code and a controlled diagnostic suite built on the existing WebArena benchmark.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original papers for WebArena, all model families (Qwen, Gemma), and vLLM. All models used are publicly released under open licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We introduce MemTrapBench, a controlled benchmark of 247 long-horizon browser tasks (231 main and 16 pilot) built from 8 structural templates, described in Sections 3.3 and 4.4 and Appendix D. The tasks, memory passages, and evaluation code are documented and released at `https://github.com/Schuture/Compliance-Trap` under CC BY 4.0 (tasks and memories) and MIT (code).

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are the core experimental subjects of this work. Section 3.3 specifies all five models (Qwen3.5-9B/27B, Gemma-4-E4B/26B, Gemini-3-Flash) and their serving configurations. Memory construction uses an LLM (described in Section 3.3).

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.