Probing Stability–Plasticity Tradeoffs in Agent Memory through Cognitive Experimental Paradigms

Jiaqi Ding
Department of Computer Science
UNC-Chapel Hill
jiaqid@cs.unc.edu

Guorong Wu
Department of Computer Science
Department of Psychiatry
UNC-Chapel Hill
guorong_wu@med.unc.edu

Abstract

Agent memory systems are increasingly used to maintain long-term user preferences, task states and evolving facts, but current evaluations often collapse memory behavior into final-answer accuracy. We introduce MEMPROBE, a cognitive-science-inspired framework for diagnosing stability–plasticity tradeoffs in agent memory. The framework is motivated by a core insight from cognitive memory research: memory is reconstructive and shaped by interference, source reliability, reinforcement, and reactivation. MEMPROBE turns this insight into four reusable experimental paradigms (interference, misinformation, consolidation strength, and reconsolidation window) that manipulate when a memory should be updated, preserved, or treated as uncertain. It further decomposes correctness into behavioral profiles that reveal how systems update, preserve, attribute, and temporally organize information. We instantiate these paradigms in a 56-episode diagnostic suite and evaluate six incremental memory systems under a unified protocol. Results show that systems with similar aggregate scores exhibit distinct behavioral profiles. MEMPROBE provides such a diagnostic lens, turning aggregate performance into interpretable profiles of memory maintenance over time.

1 Introduction

Two agent memory systems can achieve identical accuracy but behave very differently when updating memory: one may aggressively overwrite prior memories upon seeing new evidence; another may retain outdated information unless explicitly corrected. Identical scores can thus mask very different risks for long-term memory (Figure 1).

This distinction has become increasingly important because modern agent memory systems differ substantially in how they store, retrieve, compress, invalidate, and update information. Retrieval-only and Full-context systems often defer conflict resolution to query time, where a reader model interprets

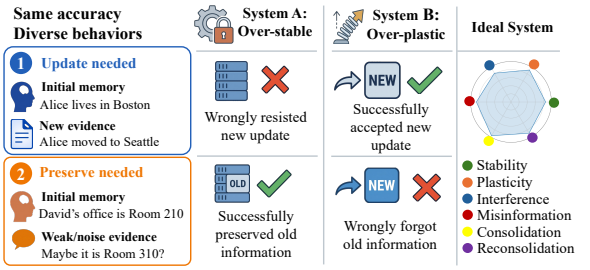

Figure 1: Illustrative depiction of how the same accuracy can mask distinct memory-update behaviors, motivating behavioral evaluation beyond final-answer correctness.

the available evidence. Other systems maintain memory during ingestion by extracting structured facts, compressing interaction histories into evolving summaries, or applying explicit operations such as ADD, UPDATE, DELETE, and NOOP. Therefore, these mechanisms can produce different update tendencies even when their final answers are similar.

Existing benchmarks have substantially expanded the scope of long-term memory evaluation. Long-context and conversational memory benchmarks such as LongMemEval (Wu et al., 2024), LoCoMo (Maharana et al., 2024) and MemBench (Tan et al., 2025a) evaluate temporal reasoning, knowledge updates, and multi-session understanding. More recent memory-agent benchmarks such as MemoryAgentBench (Hu et al., 2026b), MemoryStress (Sosa, 2026), AMA-Bench (Zhao et al., 2026), MemoryArena (He et al., 2026) and EverMemBench (Hu et al., 2026a) further move toward incremental memory maintenance, contradiction handling, agentic trajectories, and temporally evolving decisions. These benchmarks are valuable and have pushed the field beyond simple recall.

However, their primary outcomes remain task-level correctness measures such as accuracy, F1, recall or task success rate. Such measures tell us whether a system produced the correct final answer, but not what memory-update behavior produced

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 56130–56149
October 24-29, 2026 ©2026 Association for Computational Linguistics

that answer. A system may answer a contradiction case correctly because it retrieves the right evidence, because it has learned a reliable update policy, or simply because the test distribution favors update-needed examples. Conversely, a system with high aggregate accuracy may still be over-plastic or over-stable. As agent memory systems become more diverse, evaluating only final-answer correctness risks collapsing distinct update profiles into a single score.

We therefore introduce MEMPROBE, a cognitive-science-inspired framework for behaviorally evaluating stability–plasticity tradeoffs in agent memory systems. Rather than assuming access to an agent's internal memory, MEMPROBE controls the input sequence and observes the resulting behavior. This follows the experimental logic of cognitive memory research: when internal memory states cannot be directly observed, controlled behavioral paradigms can reveal systematic memory tendencies.

MEMPROBE is not intended as another fixed leaderboard benchmark. Its primary contribution is behavioral measurements and a set of reusable paradigm specifications for diagnosing memory-update dynamics. We make three contributions:

First, we introduce a paradigm-specification framework for evaluating agent memory update behavior. MEMPROBE defines four diagnostic paradigms: Interference (Underwood, 1957; Barnes and Underwood, 1959), Misinformation (Loftus, 1975), Consolidation Strength (Ebbinghaus, 1913; McClelland et al., 1995), and Reconsolidation Window (Nader et al., 2000). Each paradigm specifies what information is encoded, what perturbation is introduced, what variables are controlled, what probes are asked, and what behavioral signatures are expected.

Second, we define behavioral metrics that decompose aggregate correctness into multi-dimensional stability–plasticity profiles. Based on these metrics, MEMPROBE further generates a systematic behavioral profile for each evaluated memory system, summarizing its strengths, weaknesses, dominant failure modes and stability–plasticity tendencies in a structured report.

Third, we instantiate the paradigms in a compact diagnostic suite and apply them to existing memory systems. Empirically, MEMPROBE shows that high-scoring systems can still differ markedly in how they maintain evolving information, motivating behavioral profiles rather than single-score evaluation. At the same time, MEMPROBE is not limited to this fixed suite: we provide the trial structure, latent specification, controlled variables, probe types, gold-label requirements, and scoring protocol needed for researchers to construct customized suites in their own domains.

The rest of the paper is organized as follows: Sec.3 introduces the MEMPROBE paradigm specification. Sec.4 describes the evaluated systems, input protocols and behavioral metrics. Sec.5 reports results and analyzes stability–plasticity profiles, paradigm-wise behavior and failure modes.

2 Related Work

Agent Memory Systems. Modern agent memory systems (Park et al., 2023; Liu et al., 2026; Zhong et al., 2024; Packer et al., 2024; Fang et al., 2026; Tan et al., 2025b) differ substantially in how they store, retrieve, compress, invalidate, and update information. Extractive systems such as Mem0 (Chhikara et al., 2025) and Zep (Rasmussen et al., 2025) instead maintain structured memories by writing, merging or invalidating facts during ingestion. Compression-based systems like Observational Memory (Barnes, 2026) update an implicit summary or observation state through repeated summarization. Policy-based systems such as Memory-R1 (Yan et al., 2026) and AgeMem (Yu et al., 2026) learn explicit memory operations. These mechanisms can produce different update tendencies even when their final answers are similar, motivating evaluation that looks beyond task accuracy to the underlying memory behavior.

Agent Memory Benchmarks. Recent benchmarks (Hu et al., 2026a; Wang et al., 2026; Bian et al., 2026) evaluate how LLMs and agents retain and use information across long interaction histories. LoCoMo (Maharana et al., 2024) and LongMemEval (Wu et al., 2024) probe long-horizon conversational recall through multi-session dialogue tasks, while MemoryAgentBench (Hu et al., 2026b) and MemBench (Tan et al., 2025a) move toward incremental memory maintenance, treating memory as something accumulated through interaction rather than read from a static long context.

Another group of benchmarks stresses memory systems with contradiction, noise, and evolving facts. MemoryStress simulates 1,000 sessions with contradictions, fading memories, and accumulated noise (Sosa, 2026), and MemoryArena evaluates interdependent multi-session tasks where later success depends on distilling earlier actions

and feedback (He et al., 2026). Concurrent with our work, MemConflict (Tao et al., 2026) examines whether long-term memory systems retrieve and apply memories that are temporally valid, factually correct, and contextually applicable under conflicting evidence. MEMPROBE is complementary, rather than scoring task accuracy under long horizons or conflict, it treats conflict as one manifestation of a broader stability–plasticity trade-off and uses cognitive-science-inspired paradigms to diagnose memory update behaviors.

3 From Cognitive Memory Paradigms to Agent Memory Diagnosis

MEMPROBE formulates evaluation as a behavioral diagnosis problem: controlled contexts manipulate memory conditions, and targeted probes reveal patterns of update behaviors.

3.1 Diagnostic Paradigms from Cognitive Memory Research

Many agent memory systems differ in how they store, retrieve, compress or update information. Therefore, following the logic of cognitive memory experiments, MEMPROBE diagnoses memory behavior through observable responses rather than implementation-specific memory states (Figure 2).

Paradigm I: Interference. This paradigm is motivated by classic $AB - AC$ paired-associate learning studies (Underwood, 1957; Barnes and Underwood, 1959). Participants first learn an association A–B and later learn a competing association A–C. When probed with A, producing the old value B after learning C reflects proactive interference, whereas producing C when asked for the earlier association reflects retroactive interference. The paradigm therefore separates successful updating from preservation of historical memory.

In agent memory, the same logic captures competition among an established memory, a legitimate update and a later distractor. This paradigm probes excessive stability, where valid updates fail, and excessive plasticity, where distractors intrude as new memories.

Paradigm II: Misinformation. This paradigm studies how post-event misleading information can distort memory for an original event (Loftus and Palmer, 1974; Loftus, 1975). Its key insight is that later information affects memory not only through its content, but also through its source and credibility: misleading details from trusted sources are more likely to be incorporated, whereas unreliable sources are more likely to be discounted.

In agent memory, conflicting values may come from users, assistants, tools, retrieved documents, stale trackers or third-party statements. The central question is whether the system distinguishes evidence from mere mention. A reliable user correction or trusted tool update may warrant revision, while a hallucinated assistant statement, rumor or non-target value should not overwrite memory. This paradigm tests source-sensitive updating and misinformation-driven over-updating.

Paradigm III: Consolidation Strength. This paradigm reflects that strongly reinforced memories are more resistant to disruption. Overlearning improves retention (Ebbinghaus, 1913), and interference studies suggest that repeatedly learned associations are harder to overwrite than weakly learned ones (Underwood, 1957). Related reconsolidation work further shows that more consolidated memories can be less susceptible to modification even after reactivation (Milekic and Alberini, 2002).

For agent memory, this means update behavior should depend on prior support. A fact repeatedly confirmed across interactions should be harder to overwrite than a one-off statement, especially under weak or unreliable challenges. However, strong reliable evidence should still allow revision. This paradigm tests whether a system treats all memories as equally plastic or adjusts stability and plasticity according to memory strength.

Paradigm IV: Reconsolidation Window. This paradigm shows that recalling a consolidated memory can temporarily make it modifiable (Nader et al., 2000; Schiller et al., 2010). After reactivation, new information may update or distort the original memory before it stabilizes again. This effect can also be selective: directly reactivated memories are more susceptible to modification than indirectly activated ones (Deębiec et al., 2006).

In agent memory, an old memory may be explicitly recalled before new evidence appears. Reactivation can help compare old and new information, but it should not make the system indiscriminately accept weak speculation or misinformation. This paradigm tests whether reactivation changes update sensitivity, and whether that sensitivity is selective to reliable evidence rather than causing reactivation-induced over-updating.

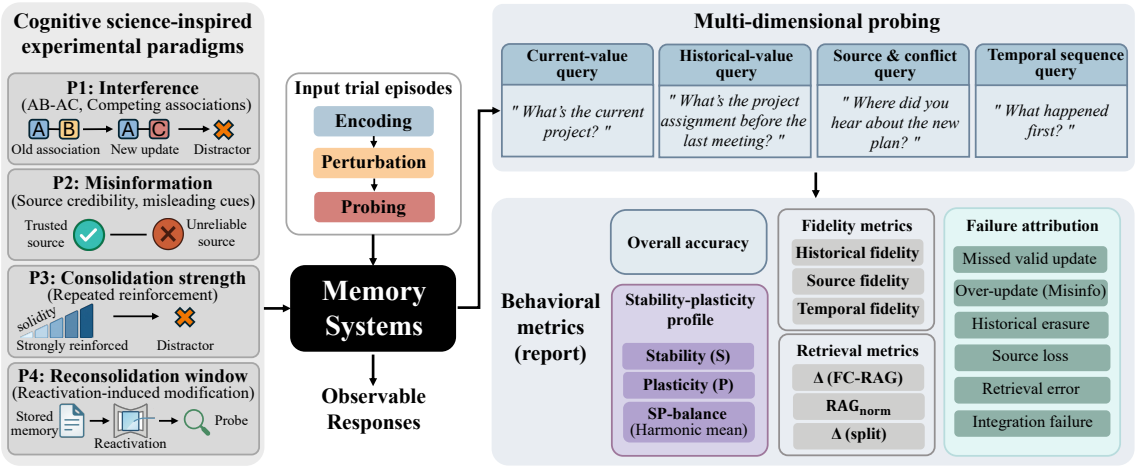

Figure 2: Overview of the MEMPROBE evaluation framework. Cognitive-science-inspired paradigms generate controlled trial episodes that are processed by a memory system and queried through multi-dimensional probes. Observable responses are converted into behavioral stability–plasticity profiles.

3.2 Controlled Trial Design

Each MEMPROBE trial is an ordered episode in the *encoding–perturbation–probing* structure designed to expose a specific memory-update behavior. In an *encoding* stage, an initial target memory is established. It may then include filler interactions, near-miss facts or distractors to create a realistic memory context and prevent the task from reducing to simple recency matching.

The central manipulation occurs in the *perturbation* stage. Depending on the paradigm, the perturbation may introduce a valid update, a conflicting but unreliable value, a weak challenge to an established memory or an explicit reactivation of a prior memory. Some trials also include post-perturbation context, which tests whether the system is overly sensitive to surface similarity, later mentions or irrelevant recency cues.

Finally, the episode ends with targeted *probes*. These probes query the system about the current state, previous state, source of information, conflict status or temporal update sequence. What must be preserved is the controlled relation among the initial memory, the perturbation, the expected behavior and the probe. App. B provides the construction details of our demonstration suite.

4 MEMPROBE Evaluation Protocol

We now specify how MEMPROBE trials are evaluated in practice. The goal is not to collapse systems into a single ranking, but to make their responses comparable through a shared input protocol and a common set of behavioral measurements.

4.1 Systems and Input Protocol

MEMPROBE supports several input protocols (Figure 3). The first set of input protocols are used as references/baselines (yellow boxes in Figure 3). A **Full-context reader** receives the complete episode at probe time and serves as an upper-reference for whether the trial is answerable under complete evidence. Retrieval-based baselines (Lewis et al., 2020), including **Naive RAG** (retrieves the top-k sessions independently for each probe), **Time-aware RAG** (additionally incorporates session order and temporal metadata) and **Oracle RAG** (provides gold-relevant evidence sessions to distinguish retrieval failure from reader reasoning failure), index sessions as retrievable units and answer each probe from retrieved evidence. We also include **heuristic baselines** that use simple rule-based candidate selection, such as preferring the most recent value or selecting values from specific source types. These heuristics are not intended as memory systems, but serve as sanity checks for shortcut vulnerability. App. E reports schema checks, probe-level validation, heuristic baselines, and cross-model validation results.

Our primary evaluation setting is **incremental memory maintenance** (purple boxes in Figure 3) because it requires the system to decide during ingestion whether new evidence should update, preserve or coexist with prior memory. For each episode, the memory store is reset and sessions are provided in chronological order. At probe time, the system must answer using its maintained memory state, retrieval interface or internal memory mechanism. Formally, for an episode e with sessions

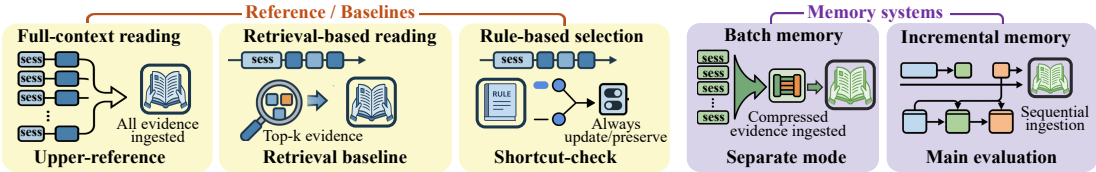

Figure 3: Input protocols used in MEMPROBE. Full-context reading, retrieval-based reading, heuristic baselines, batch memory, and incremental memory correspond to different computational interpretations of memory and are therefore reported separately. Incremental memory is the primary setting for evaluating stability–plasticity behavior.

$\{s_t\}_{t=1}^{T}$ and probes $\{q_j\}_{j=1}^{m}$, the protocol is:

$$\begin{aligned} M_0 &\leftarrow \mathrm{Reset}(), \\ M_t &\leftarrow \mathrm{Ingest}(M_{t-1}, s_t), \quad t = 1, \ldots, T, \\ \hat{a}_j &\leftarrow \mathrm{Query}(M_T, q_j), \quad j = 1, \ldots, m. \end{aligned} \tag{1}$$

We evaluate six incremental memory systems under the same sequential-ingestion protocol: **Mem0** (Chhikara et al., 2025), **Zep/Graphiti** (Rasmussen et al., 2025), **LangMem** (The LangChain Team, 2025), **Cognee** (Markovic et al., 2025), **A-MEM** (Xu et al., 2025), and **MemoryOS** (Kang et al., 2025). These systems cover extraction-based memory, temporal knowledge-graph memory, framework-level memory primitives, graph-vector hybrid memory, agentic self-organizing memory and hierarchical memory-OS designs. For consistency, all systems are fed the same ordered sessions and are probed using a unified reader over their retrieved or extracted memory. When a system requires an internal LLM for memory extraction or consolidation, we use the same internal model whenever the interface permits.

4.2 Behavioral Metrics

All metrics are computed from structured probe scores. A probe score records whether the system's answer matches the structured gold fields required by that probe, such as the current/previous values. We define the full probe taxonomy and scoring procedure in App. C. Let $\mathcal{Q}$ denote the set of all probes and let $c(q, S)$ be an indicator of whether system S answers probe q correctly under this scoring procedure. We define accuracy over any probe subset $\mathcal{D}$ as:

$$A(S; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{q \in \mathcal{D}} c(q, S). \tag{2}$$

Overall accuracy is computed by applying Eq. (2) to all probes: $\mathrm{Acc}(S) = A(S; \mathcal{Q})$.

Unless noted otherwise, confidence intervals are episode-level bootstrap intervals (2,000 resamples, fixed seed), and system comparisons use paired bootstrap on the same resampled episodes.

Stability–plasticity metrics. Let $\mathcal{U}$ denote probes from trials where the expected behavior is to accept a valid update, and let $\mathcal{P}$ denote probes from trials where the expected behavior is to preserve the existing memory. Using Eq. (2), we define plasticity and stability as:

$$\begin{aligned} \mathrm{Plasticity}(S) &= A(S; \mathcal{U}), \\ \mathrm{Stability}(S) &= A(S; \mathcal{P}). \end{aligned} \tag{3}$$

The complementary condition-level error rates are:

$$\begin{aligned} \mathrm{UpdateError}(S) &= 1 - \mathrm{Plasticity}(S), \\ \mathrm{PreserveError}(S) &= 1 - \mathrm{Stability}(S). \end{aligned} \tag{4}$$

To provide a compact summary across the two conditions, we report their harmonic mean:

$$\text{SP-Balance}(S) = \frac{2\,\mathrm{Plasticity}(S)\,\mathrm{Stability}(S)}{\mathrm{Plasticity}(S) + \mathrm{Stability}(S)}. \tag{5}$$

Eq. (5) penalizes systems that perform well on only one side of the stability–plasticity tradeoff.

Historical, source and temporal fidelity. We derive fidelity metrics from the corresponding probe types defined in App. C.1. Historical fidelity is measured by previous-value accuracy, source fidelity by source and conflict-status accuracy, and temporal fidelity by temporal-probe accuracy. These metrics measure whether a system preserves not only the current answer, but also the prior state, provenance and update sequence behind that answer.

Retrieval and evidence-integration metrics. For retrieval-based diagnostic baselines, we compare retrieved-context performance against full-context reading. Let $\mathrm{Acc}_{\mathrm{FC}}$ denote full-context accuracy and let Acc_r denote the accuracy of a retrieval-based condition r. We define $\Delta_{\mathrm{FC}}(r) = \mathrm{Acc}_{\mathrm{FC}} - \mathrm{Acc}_r$, and $\mathrm{RAG}_{\mathrm{norm}}(r) = \frac{\mathrm{Acc}_r}{\mathrm{Acc}_{\mathrm{FC}}}$. Here,

Metric	Oracle RAG (diag.)	Full-context	Mem0	Graphiti	LangMem	Cognee	A-MEM	MemoryOS	Naive RAG	Time-aware RAG
Overall [95% CI]	93.3 [89.7, 96.4]	88.8 [84.4, 92.9]	41.5 [35.7, 46.9]	62.5 [55.8, 68.8]	65.2 [57.6, 72.3]	81.7 [76.3, 86.6]	88.4 [83.9, 92.4]	60.3 [53.6, 66.5]	68.3 [60.3, 76.3]	68.8 [61.2, 75.9]

Table 1: Overall behavioral performance across references, incremental memory systems and retrieval baselines.

Δ_{FC} measures the performance loss relative to full evidence, while RAG_{norm} measures the fraction of full-context performance recovered by retrieved evidence. Values above 1 are possible for diagnostic evidence-selection baselines such as Oracle RAG, because they remove filler and near-miss sessions and provide only clean evidence.

For split-perturbation episodes, where relevant evidence is distributed across sessions, we define $\Delta_{\text{split}} = \text{Acc}_{\text{split}} - \text{Acc}_{\text{non-split}}$. Negative values therefore indicate a split penalty, while positive values indicate better performance on split episodes.

Failure attribution. Incorrect predictions are assigned to diagnostic failure categories based on the structured scores described in App. C.2. Retrieval-based systems may fail because relevant evidence is missing, only part of a split perturbation is retrieved, or the reader misinterprets retrieved evidence. Incremental systems may fail by missing valid updates, over-updating on weak evidence, overwriting historical values, losing source information, or retrieving the wrong memory at query time. These categories are reported separately from aggregate accuracy.

5 Results and Analysis

All incremental memory systems are evaluated with a shared protocol: each episode uses a fresh memory store, sessions are ingested sequentially, retrieved/extracted memory is passed to a unified Gemini-3-Flash reader, and answers are scored using the same structured JSON scorer. §5.4 verifies that the resulting profiles do not depend on which reader is used. For systems requiring an internal LLM for extraction, we use Gemini-2.5-Flash when configurable. The LLM-selection rationale and full implementation details are provided in App. A.

5.1 Overall Memory-System Performance

Table 1 provides a coarse entry point into system behavior. We report overall accuracy to summarize performance, but do not treat it as the final evaluation criterion. We separate diagnostic references from real memory systems. Among incremental memory systems, A-MEM performs best, reaching 88.4% overall accuracy and nearly matching the Full-context upper reference at 88.8%. Cognee ranks second with 81.7% overall accuracy, followed by LangMem, Graphiti, and MemoryOS in the middle range.

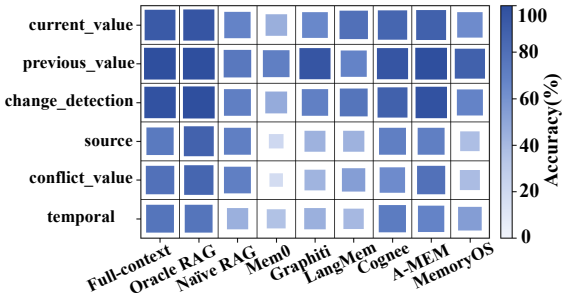

Figure 4: Probe-level performance across systems. Many systems perform well on current-value, previous-value and change-detection probes, but degrade on source, conflict-value and temporal probes.

Figure 4 further decomposes performance by probe type (Probe definitions are provided in App. C.1). Most systems perform better on current-value, previous-value, and change-detection probes than on source, conflict-value, and temporal probes. This suggests that many memory systems can retain or retrieve factual values, but often fail to preserve provenance and update-sequence information.

These aggregate scores provide only a coarse summary. As the following sections show, systems with similar overall accuracy can differ sharply in their update tendencies, historical retention, provenance preservation, temporal reasoning, and robustness to split evidence. We therefore decompose the results into stability–plasticity profiles, paradigm-wise behavior, and failure modes below.

5.2 Stability–Plasticity Profiles

Aggregate accuracy alone cannot distinguish systems that update too aggressively from systems that preserve outdated memories too strongly. We therefore compare systems by plasticity, stability and the corresponding condition-level error rates. Plasticity measures whether a system accepts valid updates, while stability measures whether it resists

weak or unreliable evidence.

System	Plast.	Stab.	SP-Bal	PreserveErr.	UpdateErr.
Mem0	28.6	54.5	37.5	45.5	71.4
Graphiti	58.9	66.1	62.3	33.9	41.1
LangMem	63.4	67.0	65.1	33.0	36.6
Cognee	75.0	88.4	81.1	11.6	25.0
A-MEM	91.1	85.7	88.3	14.3	8.9
MemoryOS	54.5	66.1	59.7	33.9	45.5

Table 2: Stability–plasticity metrics for incremental memory systems.

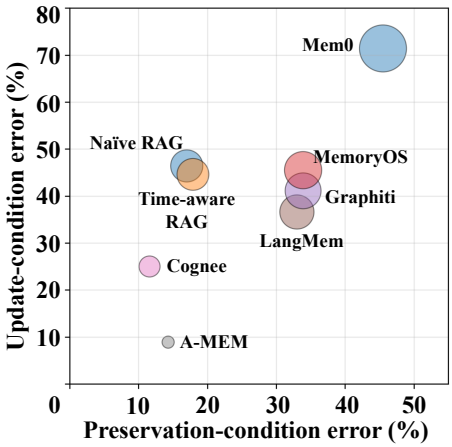

Figure 5: Stability–plasticity plane. Systems closer to the lower-left corner have better SP-balance.

Table 2 reports the stability–plasticity metrics. A-MEM has the best balance among memory systems, with 91.1% plasticity, 85.7% stability, and an SP-Balance of 88.3%. Cognee is more stability-biased: it achieves the highest stability at 88.4%, but lower plasticity at 75.0%. LangMem and Graphiti occupy a middle region with different behaviors: LangMem consolidates aggressively and loses history, while Graphiti retains historical values but struggles to update and preserve source information.

Figure 5 plots systems in the stability–plasticity plane. The ideal region is the lower-left corner, where errors are low under both conditions. A-MEM is closest to this region. Cognee is also strong but more conservative, with fewer errors when preservation is required and more when an update is required. Mem0 lies in the least desirable region.

The three mid-range systems (Graphiti, LangMem, MemoryOS) are statistically indistinguishable in overall accuracy (all pairwise $p > 0.37$), and thus directly fall into the "same accuracy" condition. But their behavior differs once correctness is decomposed by probe type. Graphiti preserves historical values far better than LangMem (previous-value 96.4 vs. 67.9, 95% CI $[+9.1, +48.3]$, $p = 0.002$), a gap that survives Bonferroni correction across all 18 system–probe comparisons, while the current-value gap runs in the opposite direction (66.1 vs. 80.4, $\Delta = -14.3$pp). Two systems with the same aggregate score therefore realize opposite maintenance strategies: one biased toward retaining the past, the other toward tracking the present.

These results illustrate the central motivation of MEMPROBE: final accuracy alone cannot reveal whether a memory system is over-plastic, over-stable or balanced.

5.3 Paradigm-wise Analysis

We next analyze system behavior across the four MEMPROBE paradigms. Each paradigm contains 14 episodes and 56 scored probes. Table 3 reports paradigm-wise accuracy.

Several patterns emerge. **First**, interference is the hardest paradigm for the strongest references: Full-context reaches 82.1%, A-MEM 82.1%, and Oracle RAG 89.3%. This suggests that competing similar facts stress memory behavior even when clean evidence is available. **Second**, consolidation sharply separates systems. A-MEM reaches 96.4%, and Graphiti achieves its best paradigm score at 75.0%, suggesting that note-based and graph-based structures help retain accumulated evidence. Mem0 performs less well on consolidation in our setting, suggesting that repeated reinforcement is not always preserved by flat extracted-memory representations. **Third**, MemoryOS shows a different pattern from most other systems: it performs best on interference but worst on misinformation and consolidation. This suggests that its heat-promotion mechanism may keep recent or competing items accessible while failing to preserve reinforced history. **Finally**, misinformation favors systems that preserve raw or well-attributed evidence. Retrieval-style access performs relatively well, A-MEM and Cognee remain strong. In contrast, systems that aggressively extract, merge or rewrite memories often blur provenance during memory construction. App. D further reports how much independent signal each decomposition axis carries.

5.4 Reader Ablation

To check whether the profiles reflect the memory representation rather than the reader's ability to

System	Interference	Misinformation	Consolidation	Reconsolidation	Overall
Oracle RAG (diag.)	89.3	92.9	98.2	92.9	93.3
Full-context	82.1	91.1	91.1	91.1	88.8
Mem0	41.1	46.4	33.9	44.6	41.5
Graphiti	53.6	60.7	75.0	60.7	62.5
LangMem	60.7	66.1	69.6	64.3	65.2
Cognee	80.4	80.4	85.7	80.4	81.7
A-MEM	82.1	85.7	96.4	89.3	88.4
MemoryOS	66.1	55.4	57.1	62.5	60.3
Naive RAG	67.9	75.0	60.7	69.6	68.3
Time-aware RAG	67.9	73.2	66.1	67.9	68.8

Table 3: Paradigm-wise accuracy across the four paradigms. Interference is difficult even for strong references, while consolidation sharply separates structured memory systems from flatter extraction or retrieval systems.

interpret it, we re-ran all six memory systems on the same stored memories with a second reader, Qwen3-32B, drawn from a different model family than both the main reader (Gemini) and the generator (GPT) used for dialogue surface form. So any difference is attributable to the reader alone.

System	Gemini	Qwen3
A-MEM	88.4	72.8
Cognee	81.7	68.3
LangMem	65.2	60.3
Graphiti	62.5	61.6
MemoryOS	60.3	51.8
Mem0	41.5	41.5

Table 4: Reader ablation. The same stored memories are answered by two readers from different model families. Absolute scores drop under the weaker reader, but the ordering is largely preserved.

Absolute scores drop under Qwen3-32B, which is expected given its weaker source and conflict-status accuracy (Table 4). However, the ordering is essentially unchanged (Spearman $\rho = 0.94$, Kendall $\tau = 0.87$). The single exception is LangMem and Graphiti, which swap positions, whose aggregate scores are statistically indistinguishable (§5.2), and whose order is therefore not expected to be stable under any perturbation.

A reader with weaker provenance reasoning lowers every system's score, but it does not change which system retains more provenance than another. The behavioral profiles reported here are therefore properties of the maintained memory rather than artifacts of the selected reader. Full cross-reader results are given in App. E.5.

5.5 Failure Taxonomy

MEMPROBE is designed to identify not only whether a system fails, but how it fails. We therefore group errors into several recurring failure modes: provenance loss, temporal reconstruction failure, distributed-evidence retrieval miss, historical overwrite, under-update, over-update, and extraction fragility.

System	Hist.	Source	Temp.
Oracle RAG (diag.)	100.0	75.8	77.0
Full-context	100.0	66.1	77.0
Mem0	71.0	17.7	36.0
Graphiti (Zep)	96.0	41.9	45.0
LangMem	68.0	46.8	41.0
Cognee	96.0	64.5	73.0
A-MEM	100.0	64.5	68.0
MemoryOS	89.0	33.9	55.0
Naive RAG	75.0	64.5	45.0
Time-aware RAG	86.0	59.7	45.0

Table 5: Historical, source and temporal fidelity across systems

The sharpest systematic failure is provenance loss. This indicates that these memory systems can store values but often fail to retain who introduced them, whether they were accepted and whether they applied to the target fact. This could in principle reflect a reader that fails to recover provenance rather than a memory that fails to retain it. Two observations separate the two. First, as shown in Table 5, Full-context uses the same reader on complete evidence and reaches 66.1%, so part of the difficulty is inherent to source recovery. But systems fall far below this same-reader ceiling (Graphiti 41.9%, Mem0 17.7%), and a gap of this size under an identical reader cannot be a read-out artifact. Second, we manually inspected every source error and labeled it as a storage-side loss (provenance absent from the retrieved memory, unrecoverable by any reader) or a read-out failure (provenance present but not recovered). As shown in Table 6, read-out

errors are roughly constant across systems (6–9), so the reader does not explain the large differences in source fidelity. What varies is storage-side loss, which grows as systems retain less of the original context. Temporal fidelity is also weak: most memory systems fall below the Full-context temporal score of 77%, suggesting that they do not reliably reconstruct update order, trigger and reason.

System	Read-out	Storage-side
A-MEM	6	2
Cognee	9	1
LangMem	9	10
Graphiti	8	11
MemoryOS	9	13
Mem0	7	20

Table 6: Manual classification of source-probe errors. Systems are ordered by how much of the original context they preserve for the reader. Read-out errors are roughly constant, whereas storage-side losses grow as systems compress memory more aggressively.

Another clear failure mode appears in the split-perturbation setting, where the update trigger/source and the accepted current value are placed in separate sessions. Solving such episodes requires retrieving and integrating both perturbation parts, rather than finding a single decisive update sentence. In Table 7, Time-aware RAG adds true temporal position labels to retrieved sessions, but improves only marginally over Naive RAG: 68.8% overall versus 68.3%, and 33.3% split accuracy versus 29.2%. Split current-value (Split CV) and change-detection (Split CD) accuracy remain stuck at 8.3% for both Naive and Time-aware RAG. This shows that the problem is not missing temporal order information.

In contrast, Oracle RAG reaches 93.3% overall and 89.6% on split episodes. Its split current-value accuracy is 91.7%, and its split change-detection accuracy is 100%. Since Oracle RAG uses the same reader but receives clean non-filler evidence, the gap between Naive RAG and Oracle RAG isolates retrieval failure as the main bottleneck. In other words, split failures occur because the retriever often fails to recover both perturbation parts, not because the reader cannot reason over the evidence.

Finally, Figure 6 summarizes split penalties across systems. Most systems degrade when evidence is distributed across sessions. MemoryOS is the only system with a positive split delta, but this should be interpreted cautiously because the split subset contains fewer source and conflict probes, which are its weakest categories. Overall, the split results show that distributed evidence integration remains a major weakness for retrieval-based and incremental memory systems.

Metric	Naive RAG	Time-aware RAG	Oracle RAG	Full-context
Δ_{FC}	20.5	20.0	-4.5	–
$\mathrm{RAG}_{\mathrm{norm}}$	0.769	0.775	1.051	–
Non-split	79.0	78.4	94.3	90.9
Split	29.2	33.3	89.6	81.2
Split CV	8.3	8.3	91.7	66.7
Split CD	8.3	8.3	100.0	91.7

Table 7: Diagnostic RAG ablation for split-perturbation episodes. Δ_{FC} measures the loss relative to full-context reading, and $\mathrm{RAG}_{\mathrm{norm}}$ measures the fraction of full-context performance recovered by retrieved evidence. Split CV denotes current-value accuracy, and Split CD denotes change-detection accuracy.

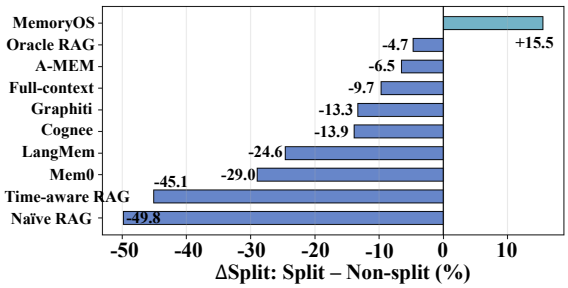

Figure 6: Split penalty/gain by systems. Negative values indicate that performance drops when perturbation evidence is distributed across sessions.

6 Discussion and Conclusion

MEMPROBE provides a cognitive-style diagnostic framework for evaluating agent memory systems beyond aggregate final-answer accuracy. By organizing memory evaluation around interference, misinformation, consolidation, and reconsolidation paradigms, MEMPROBE exposes how systems balance stability and plasticity under controlled update and preservation conditions.

MEMPROBE is not intended to be an exhaustive leaderboard. Our current suite is a compact instantiation of the proposed paradigm specifications, and future work can instantiate the same trial structure in other domains, modalities and real user histories. Nevertheless, the framework demonstrates that agent memory evaluation should move from asking only whether a system answers correctly to asking how it updates, preserves and distorts information over time.

Limitations

MEMPROBE is a diagnostic framework rather than an exhaustive benchmark. While our new generated suite demonstrates that the paradigm specifications can be systematically scaled beyond the 56-episode reference suite, the current evaluation still covers only a limited range of domains, languages, user histories and deployment settings. We use a unified reader and structured scoring protocol to enable controlled comparison across systems, and additional analysis suggests that the main findings are not driven by this choice, although native end-to-end deployments may differ. Finally, because part of the dialogue surface form is generated with LLM, the suite may still contain generator-specific biases in wording, scenario realization, or interaction style.

References

Jean M Barnes and Benton J Underwood. 1959. " fate" of first-list associations in transfer theory. *Journal of experimental psychology*, 58(2):97.

Tyler Barnes. 2026. Announcing observational memory. https://mastra.ai/blog/observational-memory. Mastra Blog. Accessed: 2026-05-24.

Haonan Bian, Zhiyuan Yao, Sen Hu, Zishan Xu, Shaolei Zhang, Yifu Guo, Ziliang Yang, Xueran Han, Huacan Wang, and Ronghao Chen. 2026. Realmem: Benchmarking llms in real-world memory-driven interaction. *Preprint*, arXiv:2601.06966.

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

Jacek Deębiec, Valérie Doyère, Karim Nader, and Joseph E LeDoux. 2006. Directly reactivated, but not indirectly reactivated, memories undergo reconsolidation in the amygdala. *Proceedings of the National Academy of Sciences*, 103(9):3428–3433.

Hermann Ebbinghaus. 1913. *Memory: A Contribution to Experimental Psychology*. Teachers College Press.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. 2026. Lightmem: Lightweight and efficient memory-augmented generation. In *International Conference on Learning Representations*, volume 2026, pages 98706–98729.

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

Chuanrui Hu, Tong Li, Xingze Gao, Hongda Chen, Yi Bai, Dannong Xu, Tianwei Lin, Xiaohong Li, Yunyun Han, Jian Pei, and Yafeng Deng. 2026a. Evaluating long-horizon memory for multi-party collaborative dialogues. *Preprint*, arXiv:2602.01313.

Yuanzhe Hu, Yu Wang, and Julian McAuley. 2026b. Evaluating memory in llm agents via incremental multi-turn interactions. In *International Conference on Learning Representations*, volume 2026, pages 156259–156291.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory os of ai agent. *Preprint*, arXiv:2506.06326.

Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. 2025. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects. *Preprint*, arXiv:2512.12818.

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. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jiaqi Liu, Yaofeng Su, Peng Xia, Yiyang Zhou, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. 2026. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*.

Elizabeth F Loftus. 1975. Leading questions and the eyewitness report. *Cognitive psychology*, 7(4):560–572.

Elizabeth F Loftus and John C Palmer. 1974. Reconstruction of automobile destruction: An example of the interaction between language and memory. *Journal of verbal learning and verbal behavior*, 13(5):585–589.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*.

Vasilije Markovic, Lazar Obradovic, Laszlo Hajdu, and Jovan Pavlovic. 2025. Optimizing the interface between knowledge graphs and llms for complex reasoning. *Preprint*, arXiv:2505.24478.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

Maria H Milekic and Cristina M Alberini. 2002. Temporally graded requirement for protein synthesis following memory reactivation. *Neuron*, 36(3):521–525.

Karim Nader, Glenn E Schafe, and Joseph E Le Doux. 2000. Fear memories require protein synthesis in the amygdala for reconsolidation after retrieval. *Nature*, 406(6797):722–726.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. *Preprint*, arXiv:2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *In the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, UIST '23, New York, NY, USA. Association for Computing Machinery.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: A temporal knowledge graph architecture for agent memory. *Preprint*, arXiv:2501.13956.

Samarth Sarin, Lovepreet Singh, Bhaskarjit Sarmah, and Dhagash Mehta. 2025. Memoria: A scalable agentic memory framework for personalized conversational ai. *Preprint*, arXiv:2512.12686.

Daniela Schiller, Marie-H Monfils, Candace M Raio, David C Johnson, Joseph E LeDoux, and Elizabeth A Phelps. 2010. Preventing the return of fear in humans using reconsolidation update mechanisms. *Nature*, 463(7277):49–53.

Jason Sosa. 2026. Why i built memorystress. https://omegamax.co/blog/why-we-built-memorystress. Accessed: 2026-05-13.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. 2025a. Membench: Towards more comprehensive evaluation on the memory of llm-based agents. *Preprint*, arXiv:2506.21605.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. 2025b. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8416–8439, Vienna, Austria. Association for Computational Linguistics.

Zhen Tao, Jinxiang Zhao, Peng Liu, Dinghao Xi, Yanfang Chen, Wei Xu, and Zhiyu Li. 2026. Memconflict: Evaluating long-term memory systems under memory conflicts. *Preprint*, arXiv:2605.20926.

The LangChain Team. 2025. Langmem sdk for agent long-term memory. https://www.langchain.com/blog/langmem-sdk-launch. Accessed: 2026-05-13.

Benton J Underwood. 1957. Interference and forgetting. *Psychological review*, 64(1):49.

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

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2024. Longmemeval: Benchmarking chat assistants on long-term interactive memory.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. 2026. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *Preprint*, arXiv:2508.19828.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. 2026. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *Preprint*, arXiv:2601.01885.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, Yuandong Tian, and Jishen Zhao. 2026. Ama-bench: Evaluating long-horizon memory for agentic applications. *Preprint*, arXiv:2602.22769.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 19724–19731.

A Implementation and Reproducibility Details

A.1 Code and Data Availability

We provide a complete reproducibility package in our Github (https://github.com/jq-ding/MemProbe). The archive includes all code used to generate and validate the demonstration suite, the full 56-episode dataset and latent specifications, the runner scripts for all evaluated memory systems, raw model outputs, computed metrics, result tables, and analysis scripts. It also contains detailed implementation notes documenting environment setup, model configurations, system-specific adaptations, excluded systems, and troubleshooting details. In addition, we include instructions for constructing customized MEMPROBE suites, including how to define new fact types, create latent specifications, generate dialogues, validate gold labels, and evaluate memory systems under the same protocol.

A.2 Environment and Model Configuration

The main environment used a unified Gemini-based evaluation stack. We used `gemini-3-flash-preview` as the final reader for all systems and `gemini-2.5-flash` as the internal LLM for memory extraction, consolidation, graph construction, or memory promotion whenever a system required an LLM component and allowed this configuration. When a system required embeddings, we used the system-supported embedding backend; for Gemini-compatible systems, we used `gemini-embedding-001`. For systems relying on local sentence-transformer embeddings, we used the default or recommended local embedding model.

Model-selection rationale. We use Gemini-3-Flash as the primary reader rather than a GPT-family model because a substantial portion of the demonstration-suite dialogue text was generated with GPT-5.4 assistance, although the latent specifications, gold labels, and validation checks were manually reviewed. Using a non-GPT reader reduces generator and reader coupling in the main evaluation. We also ran cross-model validation with GPT-5.4, Gemini-3-Flash, and Qwen3-32B (App. E.5). The qualitative diagnostic patterns were consistent across models, but Qwen3-32B was weaker on source attribution and conflict-status fields, making it less appropriate as the default reader. We therefore select Gemini-3-Flash as a strong independent reader. For internal memory operations, we use Gemini-2.5-Flash when configurable because it is supported by most evaluated systems, provides stable structured extraction, and avoids the extra runtime and output-budget complications associated with thinking-mode models.

The main goal of this configuration was to make final answering comparable across memory systems. Each system was allowed to use its own memory mechanism, but probe answering was standardized through the same reader and structured scorer.

A.3 Common Evaluation Protocol

All incremental memory systems were evaluated under the same session-by-session protocol. For each episode, we created a fresh isolated memory store, fed the episode sessions in chronological order, retrieved the system's memory or context at probe time, and then asked the unified reader to answer the probe using that context.

Formally, for each episode, the evaluation followed:

1. Reset or create a fresh memory store.
2. Ingest sessions sequentially in their original order.
3. For each probe, retrieve memory/context from the system rather than giving the full episode.
4. Pass the retrieved memory/context to the unified reader.
5. Score the structured answer using the same typed-field scorer.

Whenever a memory system exposed a context-only retrieval interface, we used it instead of the system's own final answer generation. This ensures that differences in final answers primarily reflect the retrieved or maintained memory rather than differences in the answer generator. If a system did not expose a clean retrieval interface, we treated it as an end-to-end memory system and documented this separately.

A.4 Evaluated Systems

We successfully evaluated six incremental memory systems:

- **Mem0**, representing extraction-based long-term memory.

- **LangMem**, representing framework-level memory extraction and consolidation.
- **Graphiti/Zep**, representing temporal knowledge-graph memory.
- **Cognee**, representing graph-vector hybrid memory.
- **A-MEM**, representing agentic self-organizing memory.
- **MemoryOS**, representing hierarchical memory-OS style memory.

We also evaluated retrieval-based diagnostic baselines, including Naive RAG, Time-aware RAG, and Oracle RAG. Naive RAG retrieves top-ranked sessions for each probe. Time-aware RAG additionally annotates retrieved sessions with their original temporal position. Oracle RAG bypasses retrieval and supplies clean non-filler evidence sessions, serving as a diagnostic upper bound rather than a deployable baseline.

A.5 Systems Considered but Excluded

We attempted to evaluate several additional memory systems but excluded them due to implementation or protocol incompatibilities.

- **MemGPT** (Packer et al., 2024) was excluded because the available implementation required a different version of python/server setup and its tool-based memory mechanism was incompatible with our Gemini-controlled function-calling protocol.
- **Hindsight** (Latimer et al., 2025) was excluded due to database and system dependency constraints, including PostgreSQL/pgvector and glibc incompatibilities in our environment.
- **Memory-R1** (Yan et al., 2026) was excluded because a runnable implementation was not available at the time of our experiments, and the method is an RL fine-tuning framework rather than an inference-time memory system compatible with our protocol.
- **Memoria** (Sarin et al., 2025) was excluded due to dependency conflicts with the unified reader stack and the absence of a clean fixed-transcript ingestion API under our setting.

These exclusions do not change the goal of our evaluation, which is to diagnose representative memory mechanisms rather than exhaustively rank all available systems. MEMPROBE is intended to diagnose representative memory mechanisms rather than exhaustively rank all available memory products or research prototypes.

A.6 Artifacts

For reproducibility, we release the episode suite, latent specifications, runner scripts, raw prediction reports, metric computation scripts, and the full implementation log. The full implementation log contains exact versions, installation notes, system-specific patches, runtime characteristics, and detailed failure modes for excluded systems.

B Demonstration Suite Details

Section 3 defines MEMPROBE as a set of diagnostic paradigm specifications. This appendix describes one concrete instantiation of these specifications: a 56-episode demonstration suite designed to evaluate stability–plasticity behavior in agent memory systems. The suite is not intended to exhaust all possible memory scenarios. Rather, it serves as a controlled and reproducible testbed for showing how the MEMPROBE paradigms can be instantiated, validated, and used to produce diagnostic behavioral profiles.

B.1 Suite Overview

The demonstration suite contains 56 episodes, 1,246 ordered sessions, and 224 probe questions. Each episode instantiates one of the four MEMPROBE paradigms: Interference, Misinformation, Consolidation Strength, and Reconsolidation Window. Each episode contains an initial target memory, a controlled perturbation condition, naturalistic filler interactions, and probes targeting different behavioral readouts.

Suite property	Value
Number of episodes	56
Number of sessions	1,246
Number of probes	224
Paradigms	4
Domains	Personal, Work, Agentic
Split-perturbation episodes	12
Non-split episodes	44
Probe types	Current, Previous, Change, Source, Conflict, Temporal

Table 8: Summary statistics of the 56-episode MEMPROBE demonstration suite.

The suite is balanced across the four diagnostic paradigms, with 14 episodes per paradigm. We also

Domain	Example fact types
Personal / lifestyle	Current gym, dietary preference, coffee preference, commute plan, routine preference
Work / productivity	Submission deadline, project owner, team standup time, deployment cadence, review schedule
Agentic / tool-state	Reported port number, active branch, selected database, retry limit, configuration value, tool-reported state

Table 9: Domains and representative fact types used in the demonstration suite.

balance update-oriented and preserve-oriented conditions so that simple strategies such as always updating or always preserving cannot solve the suite. In addition, the suite contains both local-evidence episodes, where the relevant perturbation is contained in a single interaction, and split-evidence episodes, where the trigger and accepted new value are distributed across multiple sessions.

The 56-episode suite is a reference instantiation of MEMPROBE rather than the specification itself. Other researchers may instantiate the same paradigms with different domains, languages, interaction styles, or memory systems, as long as the controlled relation among the initial memory, perturbation, expected behavior, and probes is preserved.

B.2 Domains and Fact Types

To avoid limiting evaluation to lifestyle preferences, we instantiate the paradigms across three broad domains: personal memory, work or productivity memory, and agentic or tool-state memory. This reflects the fact that modern agent memory systems are used not only to remember user preferences, but also to maintain project state, configuration values, tool outputs, and task-specific commitments.

Each fact type is chosen to support both update and preserve conditions. For example, a project owner can change through a confirmed handoff, but a teammate's temporary involvement should not be treated as an ownership transfer. Similarly, a reported port number may be updated by an authoritative deployment log, while a stale tool output should not be accepted as the current state. These distinctions allow similar surface values to play different roles depending on source, status, and target applicability.

B.3 Construction Pipeline

The suite was constructed using a staged pipeline that separates experimental design from surface dialogue generation. This separation is important because the main object of evaluation is not the wording of any single episode, but the controlled relationship between initial memory, perturbation, expected behavior, and probes.

The construction pipeline consists of five stages:

1. **Fact-type design.** We manually define fact types compatible with one or more MEMPROBE paradigms. Each fact type must support clear initial values, plausible updates, plausible distractors, and unambiguous current-state labels.

2. **Value-pair design.** For each fact type, we define initial values, candidate new values, distractors, and non-target values. Value pairs are reviewed to avoid ambiguity, trivial templates, and obvious shortcuts.

3. **Latent specification.** Each episode is first represented as a structured latent specification. The specification fixes the paradigm, condition, expected behavior, canonical values, source type, support level, reactivation status, and required probe types.

4. **Dialogue generation.** Dialogue sessions are generated from the latent specification. Generation is constrained so that the target memory, perturbation, fillers, and probes remain consistent with the structured design. Some filler interactions were adapted from or inspired by public long-memory benchmark materials, including LongMemEval (Wu et al., 2024), LoCoMo (Maharana et al., 2024), and MemoryArena (He et al., 2026), while the target memory manipulations, perturbations, probes, and gold labels were constructed according to our MEMPROBE latent specifications.

5. **Validation and cleanup.** We apply rule-based checks, structured gold checks, LLM-judge consistency checks, Full-context reader validation, retrieval-based baselines, and heuristic baselines. Episodes with schema inconsistencies, ambiguous gold labels, or accidental updates are revised.

This pipeline keeps the manipulated experimental variable separate from surface dialogue form.

It also makes the suite extensible: new domains or fact types can be added by creating new latent specifications that satisfy the same paradigm constraints.

B.4 Latent Specifications

Each episode is grounded in a latent specification that records the controlled variables of the trial. The latent specification is used both for generation and for evaluation. It prevents the dialogue from being treated as unstructured text and ensures that each episode has a well-defined expected behavior.

Each latent specification contains the following fields:

- **Paradigm and condition**: the MEMPROBE paradigm and the specific condition being instantiated, such as `authoritative_update`, `decoy_no_update`, `assistant_noise`, `stale_tool`, `high_support_weak_challenge`, or `reactivated_weak_evidence`.
- **Expected behavior**: whether the system should update the target memory or preserve the existing memory.
- **Canonical values**: the initial value, accepted new value if applicable, conflict value, distractor values, and previous value.
- **Source and status**: whether the perturbation comes from the user, assistant, third party, tool, stale tool output, or another entity.
- **Support and reactivation variables**: for consolidation and reconsolidation trials, the support level, evidence strength, and whether the prior memory is explicitly reactivated.
- **Split perturbation metadata**: whether the perturbation evidence is contained in one session or split across multiple sessions.
- **Structured gold**: canonical answers for current value, previous value, change status, source, conflict status, and temporal sequence.

B.5 Dialogue and Perturbation Structure

Each episode consists of ordered sessions. Sessions are assigned both a global index and a phase label. The main phases are encoding, filler, perturbation, post-perturbation context, and probing.

Encoding sessions establish the initial memory. Filler sessions introduce natural background interactions and near-miss facts. Confusable sessions mention values that are semantically or lexically similar to the target fact but should not determine the gold answer. Perturbation sessions introduce the critical update, conflict, weak evidence, or reactivation. Post-perturbation sessions test whether systems are overly sensitive to recency or surface similarity.

A standard non-split episode follows the pattern:

$$\begin{aligned}&\text{Encoding} \rightarrow \text{Filler} \rightarrow \text{Perturbation}\\&\rightarrow \text{Post-perturbation} \rightarrow \text{Probing.}\end{aligned} \tag{6}$$

In split-perturbation episodes, the perturbation is intentionally distributed:

$$\begin{aligned}&\text{Encoding} \rightarrow \text{Filler} \rightarrow P_1\\&\rightarrow \text{Interleaving filler} \rightarrow P_2\\&\rightarrow \text{Post-perturbation} \rightarrow \text{Probing.}\end{aligned} \tag{7}$$

Here, P_1 may contain the source, trigger, or reason for change, while P_2 contains the accepted current value or final decision. This design prevents a retrieval system from solving the episode by retrieving a single decisive update sentence. Instead, the system must integrate evidence across multiple sessions.

Split perturbation is included to test distributed evidence integration. In many real interactions, an update is not expressed as a single sentence of the form "the value changed from x to y." A user may first indicate that a plan has changed, then later mention the accepted new value. Alternatively, a tool may provide a trigger, while a user later confirms how it should be interpreted.

The split design separates three pieces of information that are often collapsed in simple benchmarks:

- **Change trigger**: evidence that some update or reconsideration occurred.
- **Source or authority**: who or what introduced the relevant evidence.
- **Accepted current value**: the value that should be treated as current.

A system that only retrieves P_1 may know that a change occurred but fail to identify the new value. A system that only retrieves P_2 may identify the new value but miss the trigger or source. A robust memory system should connect these pieces of evidence while preserving the previous state.

B.6 Structured Gold Labels

Each probe is paired with structured gold labels. We avoid relying only on free-form textual answers because many correct predictions differ in wording. Structured gold fields specify the intended answer at the level of value, source, status, target applicability, and temporal relation.

For current-value probes, the gold label specifies the accepted current value. For previous-value probes, it specifies the earlier value that should remain historically accessible. For change-detection probes, it specifies whether a confirmed change occurred. For source probes, it specifies who or what introduced the relevant evidence. For conflict-value probes, it specifies the conflicting value, its source, its status, and whether it applies to the target fact. For temporal probes, it specifies the old-to-new direction, trigger, reason, and whether an exact time is required.

For example, a conflict gold label has the following abstract structure:

```
{
  "conflict_value": "...",
  "mentioned": true,
  "source": "...",
  "status": "<not_accepted | stale |
          other_entity | hypothetical>",
  "applies_to_target": true or false,
  "correct_current_value": "..."
}
```

Similarly, a temporal gold label has the following structure:

```
{
  "change_direction": "<old_to_new |
                  no_confirmed_change>",
  "old_value": "...",
  "new_value": "...",
  "trigger": "...",
  "reason": "...",
  "exact_time_required": false
}
```

Structured labels make scoring more robust to paraphrase and allow partial subscores. For example, a system may identify the conflict value but fail to classify whether it was accepted, stale, hypothetical, or associated with another entity.

C Probe Taxonomy and Scoring

This section describes the probe taxonomy and scoring procedure used in MEMPROBE. The main text reports aggregate and behavioral metrics, while the details below specify how individual model responses are matched against structured gold labels.

C.1 Probe Types

MEMPROBE uses targeted probes as behavioral readouts rather than ordinary question-answering items. Each probe type isolates a different aspect of memory behavior.

Current-value probes. Current-value probes ask for the currently valid value of the target fact. They evaluate whether the system reaches the appropriate final memory state after the episode, either by accepting a valid update or preserving the original value when no valid update occurred.

Previous-value probes. Previous-value probes ask for the earlier value after a change. They evaluate whether historical information remains accessible after an update, rather than being erased or overwritten by the current value.

Change-detection probes. Change-detection probes ask whether a real update occurred. They distinguish systems that simply output a plausible value from systems that recognize the update status of a memory.

Source probes. Source probes ask who or what introduced a relevant value. They evaluate whether the system preserves provenance information, such as whether a value came from the user, an assistant statement, a tool result, a retrieved document, or another source.

Conflict-value probes. Conflict-value probes ask whether a competing value was mentioned, where it came from, and whether it applied to the target fact. They test whether the system distinguishes valid evidence from distractors, stale information, misinformation, or values belonging to another entity.

Temporal probes. Temporal probes ask about the update sequence, including the original value, current value, trigger, and reason for change. They evaluate whether the system can reconstruct the temporal structure behind a memory update, not only the final state.

Together, these probes separate memory behaviors that would otherwise be collapsed by final-answer accuracy. A system may recover the current value but lose the previous value, detect that a conflicting value was mentioned but misattribute

its source, or identify a change without correctly explaining why the change occurred.

C.2 Scoring Procedure

We score six probe types: current value, previous value, change detection, source, conflict value, and temporal sequence.

Current-value scoring. A current-value response is correct if it identifies the accepted current state of the target fact according to the episode gold label.

Previous-value scoring. A previous-value response is correct if it recovers the earlier value after an update. If no update occurred and the probe asks for a previous value, the expected answer is determined by the episode label, such as "no previous value" or the original value when the wording asks for the initially established state.

Change-detection scoring. A change-detection response is correct if it correctly determines whether a confirmed update occurred. This score evaluates update-status recognition rather than value extraction alone.

Source scoring. A source response is correct if it identifies the source associated with the relevant value or update. Acceptable answers may include normalized source categories such as user, assistant, tool, document, third party, stale record, or other entity, depending on the episode label.

Conflict-value scoring. Conflict-value probes are decomposed into multiple fields: value identification, source identification, status classification, and target applicability. A response may receive partial credit if it identifies the competing value but misclassifies whether that value was accepted, rejected, stale, hypothetical, unreliable, or non-applicable to the target entity.

Temporal scoring. Temporal probes are scored using the old value, current value, update direction, trigger, and reason for change. Exact dates or timestamps are required only when explicitly specified by the gold label. Otherwise, a correct response must capture the update sequence and the event or source that triggered the change.

C.3 Normalization and Partial Credit

Before scoring, we normalize common surface variants that do not change meaning. For example, optional prepositions in time expressions may be ignored, and numeric values may be compared with or without common units when the unit is unambiguous. For sentence-level predictions, we extract candidate values when possible before comparison.

Partial credit is used only for structured probes with multiple fields, such as conflict-value and temporal probes. This prevents formatting differences from being counted as semantic errors while preserving important distinctions among value identification, source attribution, status classification, and temporal reasoning.

C.4 Execution Errors and Exclusions

API failures, parsing failures, or execution errors unrelated to episode content are marked separately and excluded from the corresponding accuracy denominators. These exclusions are not treated as model reasoning errors. All exclusions are reported in the validation tables associated with the experiment.

D Inter-Axis Correlation

MEMPROBE reports results along two decomposition axes: the four cognitive paradigms and the six probe types. A natural question is how much independent signal each axis carries. Treating the six memory systems as samples, we compute pairwise correlations between axes and summarize the off-diagonal entries.

	Pearson		Spearman	
Axis	**mean**	**min**	**mean**	**min**
4 paradigms	0.93	0.85	0.88	0.77
6 probe types	0.78	0.43	0.75	0.43

Table 10: Off-diagonal correlation summary across the two decomposition axes, computed over the six incremental memory systems.

The paradigm axis is highly collinear: all twelve off-diagonal correlations exceed 0.85, so a system that is strong on one paradigm tends to be strong on all four. Paradigm-level accuracy therefore largely reflects general competence rather than four separable capabilities. This is expected, and it is consistent with the role the paradigms play in MEMPROBE: they construct mechanistically distinct controlled conditions under which memory behavior can be observed, not four independent scoring dimensions.

The probe-type axis is measurably less collinear. Its minimum off-diagonal correlation (0.43) is well

below the paradigm minimum (0.85), indicating that some probe types capture behavior the others do not predict. Temporal fidelity is the clearest case: it correlates only $r = 0.64$ with current-value accuracy, so reconstructing the update sequence is substantially less correlated to reporting the correct current value. Source fidelity, by contrast, correlates more strongly with current value ($r = 0.89$), but this reflects a uniform deficit rather than a shared capability: every system scores far lower on source than on current value (Table 5). The independent diagnostic signal in MEMPROBE therefore comes primarily from decomposing correctness by probe type.

E Suite Quality Control and Validation

We validate the suite at three levels: schema consistency, gold-label consistency, and behavioral sanity checks. The goal is to ensure that the episodes are internally consistent, answerable under complete evidence, and not solvable by trivial shortcuts.

E.1 Schema and Gold-Label Validation

All episodes are checked for required top-level fields, latent-spec fields, session fields, and metadata fields. Each session has a session index, phase label, session type, dialogue content, and metadata. Split episodes explicitly record the number of perturbation sessions and the identities of perturbation parts.

We use structured gold labels for all probe types and manually audit cases where gold labels and dialogue content disagree. We also normalize common value variants, such as optional prepositions in time expressions and unit-bearing numeric values. This reduces false negatives caused by surface mismatch rather than model error.

E.2 Probe-Level Validation

Table 11 reports validation accuracy by probe type. High judge and Full-context scores indicate that most gold labels are recoverable under complete evidence. The lower RAG scores on current-value and change-detection probes suggest that the suite selectively stresses retrieval and evidence integration rather than merely increasing surface ambiguity.

E.3 Behavioral Sanity Checks

We run several reference baselines before using the suite to evaluate memory systems. A Full-context reader checks whether the episode is solvable when all evidence is visible. A naive RAG baseline checks whether retrieval and evidence integration are non-trivial. Heuristic baselines such as latest-value, always-update, and always-preserve test whether the suite can be solved by simple shortcuts.

Probe type	Judge	Full-context	Naive RAG
Current value	98.2%	96.4%	71.4%
Previous value	100.0%	100.0%	92.9%
Change detection	100.0%	94.6%	65.5%
Temporal	86.4%	85.7%	81.8%
Source	97.1%	97.1%	97.1%
Conflict value	100.0%	96.4%	100.0%

Table 11: Probe-level validation accuracy for the 56-episode suite. Judge and Full-context scores are high for most probe types, indicating that structured gold labels are largely recoverable when all evidence is visible. Naive RAG shows the largest degradation on current-value and change-detection probes, suggesting that retrieval-based systems struggle most when they must identify the accepted current state and determine whether a real update occurred.

Table 12 summarizes the current validation results. The judge and Full-context reader indicate that the episodes are largely readable and internally consistent. The RAG gap and low heuristic scores indicate that the suite is not solved by simple retrieval or recency shortcuts.

Baseline	Accuracy
LLM judge	97.8%
Full-context reader	95.5%
Naive RAG	81.2%
Latest-value heuristic	18.8%
FC–RAG gap	14.3 pp

Table 12: Validation results for the 56-episode demonstration suite. The judge and Full-context reader indicate that the episodes are largely readable and internally consistent, while the RAG gap and low heuristic score indicate that the suite is not solved by simple retrieval or recency shortcuts.

E.4 Split-Perturbation Validation

The split-perturbation subset provides an additional diagnostic check. Full-context performance remains high on split episodes, indicating that these episodes are still answerable when all evidence is visible. However, naive RAG drops substantially on split episodes, showing that retrieval systems struggle when evidence is distributed across sessions.

Subset	Judge	Full-context	Naive RAG
Split episodes	95.8%	93.6%	56.2%
Non-split episodes	98.3%	96.0%	88.0%

Table 13: Split versus non-split validation results. Split perturbations preserve Full-context readability but sharply reduce RAG performance, indicating that they diagnose distributed evidence integration rather than simply making the text ambiguous.

Baseline	GPT-5.4	Gemini-3-Flash	Qwen3-32B
Judge	96.9	94.2	90.2
Full-context	92.4	88.8	78.1
Naive RAG	68.8	68.3	59.4
FC–RAG gap	23.6	20.5	18.7

Table 14: Cross-model validation summary under structured JSON scoring. All three readers show the same qualitative trend: Full-context performance is substantially higher than naive RAG, indicating that the suite exposes retrieval and evidence-integration difficulty beyond Full-context readability.

E.5 Cross-model validation

To check whether the validation results depend on a single reader model, we repeat the suite validation with three readers: GPT-5.4, Gemini-3-Flash, and Qwen3-32B. All models are evaluated with the same structured JSON scoring protocol. The probes remain natural-language questions, but model responses are normalized into structured fields before scoring. This reduces surface-form bias across models while preserving the semantic requirements of each probe.

Table 14 summarizes the overall results. All three readers achieve high judge accuracy, indicating that the structured gold labels are largely recoverable across models. Full-context performance is consistently higher than naive RAG, showing that retrieval-based reading introduces a substantial additional bottleneck. Qwen3-32B is weaker overall, especially on source and conflict-status reasoning, but it preserves the same qualitative pattern: Full-context reading outperforms naive RAG.

Table 15 breaks down Full-context performance by probe type. Across models, current-value, previous-value, and change-detection probes are consistently strong, showing that the core factual memory state is readable under full evidence. The largest cross-model differences appear in source and conflict-value probes. Qwen3-32B performs competitively on state-oriented probes but drops sharply on provenance and conflict-status probes, suggesting that source attribution and target-applicability classification are harder than factual value extraction for weaker open-weight readers.

Probe type	GPT-5.4	Gemini-3-Flash	Qwen3-32B
Current value	96.4	92.9	94.6
Previous value	100.0	100.0	96.4
Change detection	96.4	98.2	94.6
Source	76.5	73.5	41.2
Conflict value	92.9	78.6	42.9
Temporal	86.4	77.3	72.7
Overall	92.4	88.8	78.1

Table 15: Full-context accuracy by probe type across reader models. State-oriented probes are robust across models, whereas source and conflict-value probes reveal larger differences, especially for Qwen3-32B. This suggests that provenance and conflict-status reasoning are more demanding than extracting current or previous values.

Overall, the cross-model results support two conclusions. First, the suite is not specific to the GPT-5.4 reader: Gemini-3-Flash and Qwen3-32B recover the main factual memory states under full context, and Gemini closely matches GPT-5.4 in the overall validation pattern. Second, the split-perturbation effect is robust, retrieval-based reading fails much more severely on split episodes than on non-split episodes, especially for probes that require identifying the accepted current value or detecting that a true update occurred. This validates split perturbation as a diagnostic stressor for distributed evidence integration.

F A Second Suite Instantiation

MEMPROBE is specified as a construction protocol rather than a fixed dataset. To verify that the protocol is portable rather than tailored to the 56-episode suite, we instantiated a second and independent suite using the same five-stage pipeline (App. B) and evaluated the same six memory systems on it.

F.1 Construction

The second suite contains 40 episodes, 10 per paradigm, balanced across the same three domains. Two properties make it an independent test of the protocol rather than a re-wording of the original suite. First, it uses 12 new fact types with no overlap with the original suite, extending the work and agentic domains in particular. Second, dialogue surface form was generated with a different generator model (Gemini-3.1-Pro), so the instantiation does not inherit the wording conventions of the

original suite. As before, all controlled variables (latent specifications, perturbations, expected behaviors and gold labels) were human-designed and audited rather than generated.

F.2 Results

Because the second suite was generated with a Gemini-family model, we report results under a GPT-5.6 reader, keeping generation and reading in different model families as in the main evaluation.

System	Interf.	Misinf.	Consol.	Recons.	Overall [95% CI]
A-MEM	97.5	89.5	97.3	97.5	95.5 [91.7, 98.7]
Cognee	86.5	92.1	92.5	94.6	91.4 [86.3, 96.1]
MemoryOS	84.6	76.9	68.4	80.0	77.6 [70.4, 84.3]
LangMem	82.5	70.3	86.8	63.2	75.8 [69.4, 82.6]
Graphiti	75.0	71.1	66.7	83.8	74.0 [66.4, 81.8]
Mem0	45.0	34.2	43.2	48.6	42.8 [36.7, 48.4]

Table 16: Paradigm-wise and overall accuracy on the second suite. Confidence intervals are episode-level bootstrap intervals over the 40 episodes.

The core diagnostic conclusions replicate. A-MEM and Mem0 again occupy the top and bottom of the ranking, and the source accuracy again falls well below current-value accuracy for every system. Absolute scores are higher than on the original suite, which is consistent with the second suite having shorter episodes and a stronger reader.

One difference is that MemoryOS ranks third here but fifth on the original suite. The second suite is weighted toward agentic, shorter-horizon fact types, which suits MemoryOS's hierarchical short/mid/long-term organization, whereas the original suite contains longer episodes with more competing-fact interference.

G Limitations of the Demonstration Suite

The demonstration suite is LLM-assisted and intentionally controlled. This improves interpretability and validation, but does not capture the full diversity of real user logs. The covered domains are representative rather than exhaustive, and the suite does not include all possible uses of long-term agent memory.

The current instantiation is also text-only and English-only. It does not evaluate multimodal memory, multilingual interaction, or memory updates grounded in long-running external environments.

These limitations motivate larger and noisier instantiations rather than changing the role of MEMPROBE. Future work can instantiate the same paradigm specifications in multilingual, multimodal, user-derived, or application-specific settings while preserving the controlled variables that make stability–plasticity behavior measurable.

H License and Artifact Usage

H.1 License

We use standard licenses from the community and provide the following links to the licenses for the datasets, codes, and models that we used in this paper:

Mem0: Apache

Zep/Graphiti: Apache

LangMem: MIT

Cognee: Apache

A-MEM: MIT

MemoryOS: Apache

LoCoMo: Attribution-NonCommercial 4.0 International

LongMemEval: MIT

MemoryArena: CC-BY-4.0

H.2 Artifact Usage

The use of existing artifacts is consistent with their intended use in this work. We will make our code and models publicly accessible and all created artifacts will be only for research purposes and should not be used outside of research contexts.

H.3 AI Assistants Usage

We strictly follow the ARR rules in AI Assistants usage and carefully check all the produced artifacts.