

COLLABMEM: A Diagnostic Benchmark for Collaborative Memory Operations in Multi-Agent LLM Systems

Shun Wu^{1,2}, Di Wu^{1,2}, XueYou Zhang^{1,2}, Kang Liu^{1,2*}

¹The Key Laboratory of Cognition and Decision Intelligence for Complex Systems
Institute of Automation, Chinese Academy of Sciences, Beijing, China

²School of Artificial Intelligence, University of Chinese Academy of Sciences
{shun.wu, kliu}@nlpr.ia.ac.cn {di.wu, xueyou.zhang}@ia.ac.cn

Abstract

Multi-agent LLM systems increasingly rely on persistent memory to coordinate role-specialized work. Correctness requires more than recalling a fact: the fact must reach the right role, remain hidden from unauthorized roles, carry the right source, and update when a more authoritative value appears. We introduce COLLABMEM, a diagnostic benchmark for collaborative memory operations. Its matched memory-only harness fixes the trace, target role, answerer, judge, and gold event, localizing errors to sharing, access, routing, provenance, conflict resolution, and retention. COLLABMEM contains 58 hand-authored scenarios (1,118 core probes), a 174-scenario scaled split (3,359 probes), adversarial and compositional hard sets, and six transactive-memory competencies. We also propose TMS-GRAPH++, a scope-aware graph memory framework with role-filtered views, writer/provenance structure, and temporal [CURRENT] marking. Across five evaluation splits, TMS-GRAPH++ achieves the highest non-Oracle average score (69.6%). Evaluations using the official SDKs of Mem0, Letta, and A-Mem show that shared archives help, but reliable collaborative memory still requires explicit visibility, provenance, authority, and retention policies. We will release the dataset, evaluation code, and reproduced outputs to support future work on multi-agent memory systems.¹

1 Introduction

LLM applications are shifting from single assistants to specialized agent teams across software engineering, office workflows, social interaction, and tool-rich planning (Chen et al., 2024; Hong et al., 2024; Wang et al., 2024; Zhu et al., 2025).

This shift changes what memory must do. A back-end agent may observe a breaking API change that QA later needs; a support agent may be barred from seeing the same detail; and a later correction from an authoritative role may supersede the original value. The memory problem is therefore not only long-context recall, but *collaborative memory*: sharing, withholding, routing, resolving, and compressing information across role-governed views. This perspective is consistent with transactive memory theory, which models team memory as distributed knowledge plus meta-knowledge about who knows what and which sources are credible (Wegner, 1987; Hollingshead, 2001).

The corresponding evaluation problem remains open. Long-memory benchmarks test recall, personalization, updates, temporal reasoning, and abstention in mostly single-user histories (Xu et al., 2022; Zhong et al., 2024; Maharana et al., 2024; Wu et al., 2024; Hu et al., 2026; Tan et al., 2025). Multi-agent benchmarks measure task success, social behavior, or interaction trajectories, where memory errors mix with planning, communication, and tool-use errors (Liu et al., 2024; Zhou et al., 2024; Wang et al., 2024; Zhu et al., 2025). Recent multi-party memory benchmarks broaden the setting beyond dyadic user-assistant histories, but they do not make role-governed visibility, source authority, and current-value resolution a controlled requirement for every probe (Yang et al., 2026; Owolabi, 2026). As a result, it is hard to tell whether a multi-agent memory system failed because it did not store the fact, retrieved it for the wrong role, lost provenance, leaked protected content, or compressed away an obsolete value still needed as evidence.

To address this gap, we introduce COLLABMEM (Figure 1), a diagnostic benchmark that turns collaborative memory into a controlled,

*Corresponding author

¹<https://github.com/NLPerWS/COLLABMEM>

COLLABMEM: role-governed memory operations → auditable probes → matched evaluation → design signals

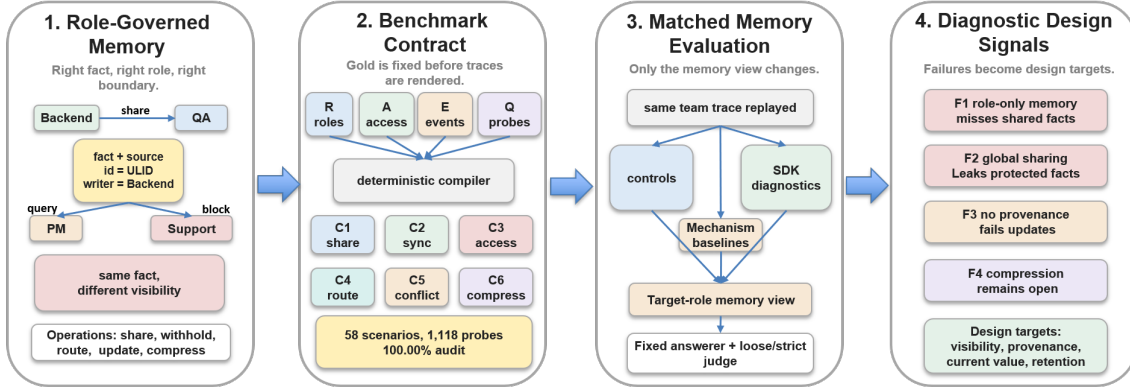


Figure 1: COLLABMEM diagnoses collaborative memory as a role-governed information-flow problem. (1) Facts must be selectively shared, withheld, routed, resolved, and compressed across role-specific views. (2) COLLABMEM fixes a benchmark contract over roles, access rules, events, and probes, with gold computed before trace rendering. (3) A matched memory-only harness holds the trace, target role, answerer, and judge fixed, varying only the memory view. (4) Scores localize failures into design signals for visibility, provenance, current-value resolution, and retention.

memory-only evaluation problem. COLLABMEM fixes the team trace, target role, answerer, judge, and gold event, and varies only the memory context exposed by the evaluated system. This design separates memory failures from planning and tool-use failures, makes visibility, source authority, and current-value resolution explicit, and maps wrong answers to repairable sharing, access, routing, provenance, conflict-resolution, and retention operations. We also propose TMS-GRAPH++, a scope-aware graph memory framework with role-filtered visibility, writer/provenance structure, and temporal [CURRENT] marking to enforce authorized views, preserve source authority, and surface current values after updates.

Contributions. This paper makes four contributions.

- We formulate *collaborative memory operations* as a memory-only task for role-governed multi-agent LLM systems, grounded in six transactive-memory competencies.
- We build COLLABMEM, an auditable benchmark with core, scaled, adversarial, and compositional splits, and will release the dataset, code, outputs, and audit scripts.
- We propose TMS-GRAPH++, a scope-aware graph memory architecture that combines role-filtered views, writer/provenance structure, and temporal [CURRENT] marking.
- We evaluate trivial controls, controlled mechanism baselines, and official SDK evaluation rows under one matched harness, showing

that strong collaborative memory requires selective propagation, access-aware retrieval, source/current-value metadata, and budgeted retention.

2 Related Work

Memory benchmarks. Long-memory benchmarks make persistent interaction measurable by testing recall, personalization, updates, temporal reasoning, and abstention across long histories (Xu et al., 2022; Zhong et al., 2024; Maharana et al., 2024; Wu et al., 2024; Hu et al., 2026; Tan et al., 2025). Agent-memory studies further show that memory streams, reflection, retrieval, and lifelong skill accumulation can improve interactive agents (Park et al., 2023; Shinn et al., 2023; Wang et al., 2023). Multi-party memory benchmarks extend the setting to group conversations and social relations (Yang et al., 2026; Owolabi, 2026). COLLABMEM is complementary but targets a different variable: the memory layer of a role-governed team. It asks whether a system can share, withhold, attribute, resolve, and compress the right facts under fixed access and authority constraints.

Multi-agent benchmarks and systems. AgentBench, SOTOPIA, OfficeBench, AgentVerse, MetaGPT, CAMEL, AutoGen, and MultiAgentBench evaluate teams through task completion, social outcomes, planning, and collaboration (Liu et al., 2024; Zhou et al., 2024; Wang et al., 2024;

Chen et al., 2024; Hong et al., 2024; Li et al., 2023; Wu et al., 2023; Zhu et al., 2025). These benchmarks are necessary for end-to-end evaluation, but they do not identify whether a failure came from memory, planning, communication, reasoning, or tool use (Yao et al., 2022). In parallel, memory systems provide hierarchical buffers, long-term stores, retrieval-augmented context, graph memory, linked notes, and cognitive memory abstractions (Lewis et al., 2020; Packer et al., 2023; Chhikara et al., 2025; Rasmussen et al., 2025; Xu et al., 2026; Sumers et al., 2023); multi-agent memory work adds private/shared tiers, role-aware routing, governed sharing, and team memory services (Cuadros et al., 2026; Liu et al., 2025; Zhang et al., 2026; Hao et al., 2026; Li, 2025; Wu et al., 2026). COLLABMEM fixes the answering layer so these primitives can be evaluated as memory primitives.

Transactive memory and access. Transactive memory theory models team memory as distributed object knowledge plus meta-knowledge about who knows what (Wegner, 1987; Hollingshead, 2001). Work on asymmetric information access shows that participants in the same interaction may legitimately know different facts (Kim et al., 2023). COLLABMEM operationalizes these ideas for LLM systems: every scenario specifies roles, access boundaries, authority, and supersession, turning specialization, credibility, coordination, and shared meta-knowledge into score-bearing probes.

3 Benchmark Design

3.1 Task formulation

COLLABMEM evaluates a memory architecture as a write-index-retrieve layer for a role-governed team. An instance is a tuple (R, A, E, Q) . R is a set of roles with topical responsibilities; A is an access matrix specifying which roles may store, retrieve, or disclose each fact; E is an ordered event schedule with observers, fact payloads, timestamps, authority labels, and supersession pointers; and Q is a deterministic probe set derived from E and A . During evaluation, every system replays the same trace, answers the same role-targeted probes with the same answering LLM, and is scored against the same gold. The benchmark therefore tests the memory substrate rather than a full agent stack: a system may use buffers,

vector stores, graphs, private/shared banks, or tool-mediated retrieval, but it must expose the correct evidence to the queried role at query time.

This formulation gives COLLABMEM its diagnostic character. A probe fixes *who* is asking, *what* they are allowed to know, and *which* prior event makes the answer correct. Because those variables are fixed outside the evaluated system, an error can be mapped to an operation: missed propagation, leakage, lost authority metadata, stale retrieval, or over-aggressive compression.

Design principles. COLLABMEM is built around four design principles. **Memory is isolated from agency.** The benchmark does not ask systems to plan, negotiate, call tools, or decide which agent should speak; these abilities are important, but they would obscure whether a failure comes from memory or from the surrounding agent loop. **Visibility is part of correctness.** A response is not correct merely because it contains the right value; it must be available to an authorized target role and unavailable to excluded roles. **Current-value resolution is source-resolved.** When two roles provide conflicting updates, the gold follows the event’s authority and supersession metadata, not whichever statement appears last in surface text. **Every probe is auditable.** Each question points back to a source event, an access rule, and a competency label, so the benchmark can explain what a failed row needs to fix.

Granularity. Scenarios are containers, but probes are the scoring unit. This choice matters because a single collaborative episode should test more than local recall. The same event schedule can ask whether a fact propagated to the right role (C1), whether another role stayed synchronized after an update (C2), whether an unauthorized role was blocked (C3), whether the system knows which role is authoritative (C4), whether conflicts are resolved with provenance (C5), and whether stale but explanatory state survives compression (C6). Thus, COLLABMEM avoids a common ambiguity in aggregate task scores: a system can succeed on easy recall while failing the memory operation that makes team deployment risky.

Probe families. Each competency is instantiated with value-seeking and evidence-seeking probes. Value probes ask for the operational answer a deployed agent would need, such as the current end-

point format or the resolved medication dose. Evidence probes ask for the writer, source role, or update rationale that makes the value trustworthy. This split is important because collaborative memory errors can be silent: a system may return the right value for the wrong reason, or recall a stale value without exposing that it has been superseded. The strict setting in Section 6 uses this evidence channel to separate recall from governed, provenance-aware memory.

Evaluation contract. The harness separates release-time gold labels from run-time memory. Event identifiers, access labels, authority scores, and supersession links are stored in the benchmark artifact; an evaluated system runs through its declared memory policy, retrieves a context for the target role, and passes that context to the fixed answerer. This contract makes two properties measurable. First, a system cannot compensate for under-sharing or leakage by changing the dialogue, because the trace and query role are fixed. Second, loose and strict scores diagnose different design failures: loose failures indicate missing or unauthorized values, while strict failures indicate missing provenance, authority, or current-state evidence.

3.2 From transactive memory to six competencies

COLLABMEM is grounded in transactive memory systems (TMS) theory (Wegner, 1987; Hollingshead, 2001), which models team memory as distributed object-level knowledge plus a shared directory of who knows what. We convert the four TMS pillars into testable benchmark requirements: **specialization** becomes routing to the authoritative role; **credibility** becomes conflict resolution under source authority; **coordination** becomes directed sharing and access control; and **shared meta-knowledge** becomes cross-role consistency and compression of team state. Table 1 states the resulting six competencies and, critically, the structural requirement that every scenario must satisfy before the competency can be probed. In this sense, COLLABMEM is TMS-grounded by construction rather than labeled after generation.

3.3 Construction pipeline

COLLABMEM uses controlled synthesis because collaborative-memory failures require ground-truth access boundaries, source authority, and su-

Competency	TMS pillar	What every scenario must contain
C1 sharing	Coordination	a fact the access matrix directs to a recipient subset of roles
C2 consistency	Shared meta	an attribute updated so roles observe different versions
C3 access	Coordination	a SENSITIVE fact whose recipients exclude the asked role
C4 routing	Specialization	a topic with exactly one role designated authoritative
C5 conflict	Credibility	a CONTRADICTION carrying a credibility weight and a resolving pointer
C6 compression	Shared meta	a superseded value that must survive an explicit memory budget

Table 1: The TMS $\rightarrow$ COLLABMEM derivation. Two transactive-memory pillars (Coordination and Shared meta) are operationalized *twice*: as a measurable competency (column 1) and as a structural requirement authored into every scenario (column 3). Authoring a COLLABMEM scenario means satisfying column 3 for all six rows; the probe and its gold are then read from the fixed event schedule. This is what makes the dataset TMS-grounded *by construction*.

persession chains that are rarely recoverable from natural logs. Synthetic generation is therefore not a shortcut; it is the mechanism that lets us place evaluation variables at known coordinates and audit them later. Figure 2 summarizes the pipeline. A seed author first specifies roles with disjoint responsibilities, an access matrix, and an event schedule whose FACT, CONTRADICTION, SENSITIVE, ROLE_SPECIFIC, and DEPRECATION events satisfy all six structural requirements in Table 1. Only after this gold-bearing structure is fixed do we render a natural-language trace. The simulator can vary surface wording, but it cannot change the gold: every answer is computed from R , A , and E .

Seed authoring. We hand-author 58 seed scenarios across software development, research collaboration, customer support, clinical coordination, and cross-functional project management. Each seed specifies roles, access permissions, authoritative sources, and an ordered event schedule. Five event types create the benchmark’s controlled failure modes: FACT introduces a value; CONTRADICTION supersedes an earlier value; SENSITIVE marks information that some roles may not see;

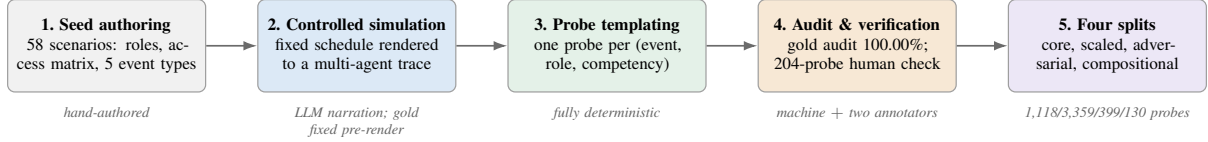


Figure 2: The COLLABMEM construction pipeline. The gold-bearing event schedule is fixed *before* simulation, so LLM narration cannot drift the gold; probe instantiation is fully deterministic; and both a machine audit and a two-annotator human check gate the release.

ROLE_SPECIFIC scopes a fact to one responsibility area; and DEPRECATION retires an attribute while leaving prior state relevant under compression.

Simulation and probe templating. A controlled simulator turns each schedule into a natural-language multi-agent trace. The benchmark never extracts gold from the generated dialogue; it uses the schedule as the source of truth. Deterministic templates then bind each question to a target role and a competency, producing 1,118 core probes. This separation between *surface realization* and *gold construction* makes the dataset auditable: even if a generated utterance is stylistically awkward, the gold label does not move.

Hard and scaled splits. The released benchmark contains three derived stress tests beyond the core seeds. The *scaled* split expands the same schema to 174 scenarios and 3,359 probes. The *adversarial* split applies targeted transformations that should not change the gold but do increase retrieval difficulty: indirect sharing, obfuscated access boundaries, multi-hop routing, and silent conflicts without an explicit update cue. The *compositional* split chains two competencies in one question, such as resolving a conflict and then naming the authoritative source. Table 3 lists split sizes; Table 2 shows how one clinical-coordination scenario yields all six competency probes.

3.4 Auditable gold and human verification

Every probe has a traceable gold anchor in the event schedule. For value probes, the gold resolves to the latest non-superseded value of the relevant (entity, attribute) chain; for access and conflict probes, the resolver additionally checks the access matrix, source authority, and resolving update. This property is machine-checkable: a deterministic audit over all 1,118 core probes passes at 100.00% with 0 residual cases. We also conduct a two-annotator verification on a 204-probe stratified sample. Both-annotator acceptance is 100.00% for answerability, 99.01% for gold cor-

Comp.	Probe (clinical-coordination scenario)	Gold
C1	Cardiology shared the allergy with Primary; ask Primary	penicillin
C2	After the plan revision, ask Scheduling for the current plan	revised
C3	Ask Scheduling (unauthorized) for the HIV status	refuse
C4	Ask Primary who is authoritative on the medication plan	Cardiology
C5	Dose is revised after a contradiction; ask Primary for resolved dose + source	5mg, Cardiology
C6	Under a memory budget, ask for the superseded initial dose	10mg

Table 2: Worked example: one clinical-coordination scenario probed by all six competencies. Each gold answer is derived from the event schedule and access matrix, not from generated dialogue.

Split / audit	Scenario unit	Probes
Core	58	1,118
Scaled	174	3,359
Adversarial	core-derived	399
Compositional	core-derived	130
Human verification sample	probe sample	204

Table 3: Dataset size. Adversarial and compositional sets are probe-level transformations over the core scenarios; they test robustness under controlled perturbations rather than scenario-level generalization.

rectness, and 98.04% for competency label, with all three dimensions accepted simultaneously on 96.57% of the sample.

4 TMS-GRAPH++: A Scope-Aware Memory Architecture

A collaborative-memory architecture needs four properties: it must (1) expose the right facts to the right role without making memory global, (2) keep the latest authoritative value distinguishable from superseded values, (3) preserve a routing index of *who knows what* without leaking the contents themselves, and (4) remain lightweight enough to be paired with a fixed answering LLM. TMS-GRAPH++ addresses all four with two de-

TMS-GRAPH++ retrieval: input (q, r, G)
1. <i>Scope filter:</i> $V \leftarrow \{e \in G : r \in \text{visible}(e)\}$.
2. <i>Authority tag:</i> per (subject, predicate), mark the edge maximal in (contradiction, credibility, time) as [CURRENT].
3. <i>Linearize:</i> emit V in chronological order with writers and [CURRENT] tags.
4. <i>if</i> q targets transactive retrieval (C4): append the writer / specialist registry.
5. A fixed LLM answers q over the emitted context only.

Figure 3: TMS-GRAPH++ retrieval. Step 1 implements role-aware scope filtering, step 2 marks the current authoritative value, and step 4 adds the conditional routing registry used for transactive retrieval. Table 7 removes these components to test whether the gains come from explicit visibility and current-value structure.

sign choices built on a role-filtered visibility graph; Figure 3 shows the resulting retrieval procedure.

Role-aware scope filter. Each event becomes an edge in a visibility graph with writers, a valid-from timestamp, credibility, a contradiction marker, and the set of roles allowed to see it. At query time, TMS-GRAPH++ returns only edges visible to the target role. Crucially, the writer-and-specialist registry (which role owns which topic) is exposed *only when the probe targets transactive retrieval (C4)*, preserving routing without leaking contents on access probes (C3). This satisfies TMS coordination and specialization while respecting the access matrix.

Temporal linearization with [CURRENT]. Within the visible edges, TMS-GRAPH++ computes an authority key for each (subject, predicate) pair as the lexicographic tuple (contradiction-flag, credibility, ingestion-time) and tags the maximal edge as [CURRENT]. The remaining edges stay visible in chronological order with their writers, providing both the resolved value (for C2/C5) and an inspectable history (for C5/C6). This operationalizes the TMS credibility pillar: the answering LLM receives an explicit cue about which version is currently authoritative rather than being left to infer authority from recency alone.

Answering interface. TMS-GRAPH++ is deliberately a *memory layer*: it emits a textual context block and delegates answering to the same fixed LLM used by every other system in our evaluation. This isolation lets Table 4 attribute score differences to the memory architecture rather than to a stronger answerer, and it keeps TMS-GRAPH++’s

inference-time cost to a graph filter and a sort—both $O(|G|)$ in the number of events.

Comparison to prior memory architectures. Hierarchical-buffer systems (Packer et al., 2023), long-term stores (Chhikara et al., 2025; Rasmussen et al., 2025), and Zettelkasten memory (Xu et al., 2026) provide strong primitives for persistence, retrieval, consolidation, and provenance. On their own, however, they do not specify the collaborative policy that COLLAB-MEM probes: which roles receive a fact, which views may expose protected content, and which source is authoritative after a contradiction. TMS-GRAPH++ represents the team’s memory as one shared graph with role-filtered views, achieving private-by-default behavior without sacrificing the team-level routing layer needed for C1/C4. It is also distinct from globally shared logs (FullShare) and lexical retrieval (RAG-Shared): both expose many useful facts but cannot withhold protected ones, and neither carries an authority cue, so they trade C3 leakage for C1/C2 availability rather than satisfying both.

5 Experimental Setup

Systems. We evaluate four groups under a shared answering layer. **Trivial controls:** NoMem (no memory), FullShare (unrestricted shared log), and RAG-Shared (top- k lexical retrieval over one shared store). **Controlled mechanism baselines:** rows named *Mem0-inspired*, *Letta-inspired*, and *A-Mem-inspired* reimplement the published memory *mechanism*—a scoped fact store, core/archival tiers, and linked notes—inside our uniform collaborative-memory protocol. These rows test whether the mechanism supports the six operations when the sharing policy and answerer are held fixed; they are not official product runs. Graph-Shared is a shared graph without TMS-GRAPH++’s scope filter and [CURRENT] tag, while TMS-Graph is kept as an internal ablation without the conditional writer registry. **Official SDK evaluation rows** run released systems through the same memory-only harness: Mem0 (Chhikara et al., 2025) via the published mem0ai SDK, Letta (Packer et al., 2023) via its SDK archival-memory interface, and A-Mem (Xu et al., 2026) via its released AgenticMemorySystem implementation. For Mem0 and Letta, role-only rows maintain one store/archive per role; role+team rows additionally write non-sensitive

events to a shared team namespace/archive. These rows measure the exposed memory primitive under minimal transparent sharing policies, not best-effort native multi-agent deployments. Finally, *Oracle* receives a perfectly accurate, access-respecting, provenance-rich view of the trace and serves as a calibration ceiling.

This separation is important for interpretation. Controlled mechanism baselines ask which design idea improves which competency under the benchmark policy. Official SDK evaluation rows ask what a released memory primitive recovers under a minimal policy that can be reproduced without product-specific agent orchestration.

Scoring. All systems use the same LLM answering layer; only the memory context visible to that model differs. The loose grader is an LLM-as-judge semantic scorer (Zheng et al., 2023). The strict protocol is not simply the loose answers regraded more harshly: the answerer is explicitly asked for value, authoritative source, and, for conflicts, the resolving update; the strict judge then requires exact value tokens, explicit role attribution, temporal evidence, and no unsupported hedging. Oracle is run through the identical strict protocol to establish the attainable ceiling.

Backbones and reproducibility. The main answering backbone is gpt-4o-mini; backbone sensitivity additionally covers claude-3.5-haiku, qwen-2.5-72b, and Llama-3.1-70B, all routed through a single provider for comparability. All numbers in this paper are generated from the released artifact tree (scenarios, traces, probes, raw answers, judge labels). The ranking we report is the one reproduced by the artifacts.

6 Results and Analysis

6.1 Core scorecard

Table 4 reports the primary diagnostic scorecard. Four findings support the benchmark’s intended interpretation. **First, isolated role memory is insufficient for collaborative recall.** Official SDK role-only rows preserve boundaries but often never expose facts observed by another role; adding a team archive lifts Letta SDK to 74.3 overall, whereas Mem0 SDK remains near its role-only score under our minimal policy. **Second, ungoverned sharing is not a solution.** Full-Share and RAG-Shared recover many useful val-

System	C1	C2	C3	C4	C5	C6	Overall
<i>Trivial controls</i>							
NoMem	0.0	0.0	100.0	11.7	0.0	0.0	18.6
Full	67.2	83.0	60.0	78.6	80.1	45.7	69.1
RAG	58.4	68.1	39.0	74.0	69.3	55.6	60.7
<i>Controlled mechanism baselines</i>							
M0-insp.	71.2	77.7	96.8	64.9	82.5	54.3	74.6
Letta-insp.	72.0	76.6	96.8	65.1	77.7	58.0	74.4
A-Mem-insp.	1.6	6.4	100.0	15.7	36.8	55.6	36.0
Graph	73.6	81.9	97.9	94.9	87.3	51.9	81.2
TMS++	81.6	94.7	96.8	99.7	92.2	49.4	85.7
<i>Official-SDK evaluation rows</i>							
M0-r	0.0	0.0	100.0	26.3	0.0	1.2	21.2
M0-r+t	0.0	1.1	100.0	22.6	0.0	1.2	20.8
Letta-r	1.6	5.3	100.0	18.3	28.3	64.2	36.3
Letta-r+t	68.8	78.7	96.8	74.9	72.3	54.3	74.3
A-Mem-r	1.6	5.3	100.0	16.9	29.5	63.0	36.0
<i>Calibration</i>							
Oracle	82.4	100.0	100.0	100.0	93.4	49.4	87.5

Table 4: Core loose scorecard with gpt-4o-mini. Compact names: M0=Mem0; r=role; r+t=role+team; insp.=controlled mechanism baseline.

ues, but they leak protected C3 content on 40.0% and 61.0% of access probes. **Third, graph structure helps only when paired with visibility and current-value cues.** Graph-Shared improves C4/C5, while TMS-GRAPH++ is stronger because it filters by target role and marks the authoritative current value. **Fourth, the benchmark is not saturated.** Even Oracle reaches only 49.4% on C6, indicating that collective compression requires more than perfect access to retrieved facts.

6.2 Harder settings: strict, adversarial, compositional, scaled

The harder splits test whether the same conclusions hold under stronger evidence requirements and perturbed traces. Across all five splits, **TMS-GRAPH++ has the highest non-Oracle average (69.6%)**, 2.5 points above the next architecture. The Letta SDK role+team archive remains competitive (60.2% average), showing that a released shared archival primitive is useful; its gap to TMS-GRAPH++ indicates that explicit role/provenance structure still matters. The strict protocol requires value, authoritative role, and temporal support. Oracle reaches 68.7%, so the task is answerable; the best non-Oracle strict score (53.0%) marks the remaining architecture gap. The compositional split is the sharpest stress test: 52.1% is the best system score, suggesting that chaining conflict resolution with routing or compression exposes failures hidden by single-operation probes.

System	Loose	Strict	Adv.	Comp.	Scaled	Avg.
<i>Trivial controls</i>						
NoMem	18.6	18.4	29.0	22.5	18.6	21.4
Full	69.1	31.0	58.0	37.8	67.6	52.7
RAG	60.7	25.2	50.7	42.4	56.9	47.2
<i>Controlled mechanism baselines</i>						
M0-insp.	74.6	30.0	63.2	39.5	71.5	55.7
Letta-insp.	74.4	30.1	63.6	38.4	71.1	55.5
A-Mem-insp.	36.0	21.5	41.4	37.0	35.0	34.2
Graph	81.2	53.0	73.2	49.5	78.7	67.1
TMS++	85.7	52.9	74.2	52.1	82.9	69.6
<i>Official-SDK evaluation rows</i>						
M0-r	21.2	20.2	34.6	22.4	20.9	23.9
Letta-r	36.3	23.5	39.0	42.7	35.8	35.5
Letta-r+t	74.3	40.4	65.4	47.0	73.9	60.2
A-Mem-r	36.0	24.1	38.6	40.9	33.9	34.7
<i>Calibration</i>						
Oracle	87.5	68.7	75.2	51.2	87.0	73.9

Table 5: Harder evaluations. Loose and Strict are core overall scores; Adv. and Comp. are adversarial and compositional splits.

Backbone	Graph	TMS	TMS-lite	TMS++
gpt-4o-mini	81.2	80.6	81.1	85.7
claude-haiku	85.4	86.0	85.3	87.7
qwen-72b	84.6	86.6	85.1	85.1
llama-70b	78.9	74.6	76.0	81.2

Table 6: Core loose overall across four answering backbones routed through one provider for comparability. Scores vary with the LLM; TMS-GRAPH++ is best in all four runs.

6.3 Backbone sensitivity

Table 6 controls for answerer choice. Absolute scores vary with the LLM, but the ordering is stable: TMS-GRAPH++ is highest under all four backbones. This pattern is consistent with the benchmark design goal: the reported gains trace to what the memory layer stores, filters, and marks, not to hidden reasoning capacity in the answerer.

6.4 Ablation

Removing either the role-aware scope filter (TMS-lite) or the conditional writer registry (TMS-Graph) costs roughly five absolute overall points on core loose (Table 7); Graph-Shared, which lacks the full TMS-GRAPH++ visibility and current-value structure, shows a comparable gap. The ablation therefore supports the central architectural claim: collaborative memory needs both role-scoped visibility and provenance-aware current-state structure.

Variant	Core overall	Δ
Graph-Shared	81.2	-4.5
TMS-Graph	80.6	-5.1
TMS-lite	81.1	-4.6
TMS-Graph++	85.7	–

Table 7: Ablation on core loose (gpt-4o-mini). Δ is relative to full TMS-GRAPH++; variants remove scope filtering, writer routing, and/or current-value marking.

6.5 Failure-mode analysis

Because each probe is tied to a competency, access rule, source event, and gold value, a wrong answer can be interpreted as a failed memory operation rather than an undifferentiated accuracy loss. **F1: isolated role memory cannot support team recall.** SDK role-only rows preserve boundaries but block C1/C2 facts; Letta’s role+team archive confirms that explicit non-sensitive sharing paths matter. **F2: global sharing trades access control for recall.** FullShare and RAG-Shared recover many values but leak protected C3 content on 40.0% and 61.0% of access probes. **F3: retrieval without provenance is brittle under updates.** Graph-Shared improves routing and conflict lookup, yet strict and compositional gaps require writer identity, source authority, temporal order, and current-versus-superseded state. **F4: compression is not solved by perfect retrieval.** Oracle’s 68.7% strict score and 49.4% C6 score show that compression requires retention planning, not only retrieval. These failures redirect optimization from larger contexts alone to visibility, provenance, current-state marking, and retention policy.

7 Conclusion

COLLABMEM establishes collaborative memory as an explicit evaluation target for multi-agent LLM systems. By fixing interaction traces, access rules, respondents, judges, and gold events, it frames role-governed sharing, information withholding, routing, conflict resolution, and data retention as auditable memory operations, instead of side effects of task execution. The results demonstrate the value of this decoupled evaluation. Role-specific memory fails to propagate information across roles; global storage suffers from information leakage; retrieval without provenance breaks after content updates. As diagnostic signals, these limitations point out clear design directions for future multi-agent memory systems.

Limitations

COLLABMEM has four scope boundaries. First, the scenarios are controlled rather than collected from live deployments; this choice supports auditable gold and clean memory-layer attribution, while naturally occurring team traces may contain interaction patterns beyond the current release. Second, the official SDK rows evaluate memory primitives under transparent sharing policies, not exhaustive product-specific multi-agent deployments. Third, the current benchmark covers a finite set of domains and uses a stratified human-verification sample, leaving room for broader coverage as the benchmark grows. Fourth, our method has limitations in compression capability, which lacks mechanism-level optimizations specifically designed for compression. The current C6 metric does not adequately reflect the methods compression capability. We will improve this in future work. Future work will extend COLLABMEM with deployment-derived traces, more domains and languages, and richer native multi-agent memory integrations while preserving the matched memory-only evaluation protocol.

Ethics Statement

Some scenarios are inspired by sensitive domains such as clinical coordination and customer support, but the benchmark uses synthetic examples and should not be interpreted as medical, legal, or operational advice. The role-based access probes are intended to support safer memory-system design by measuring leakage of protected facts. Researchers using COLLABMEM should avoid inserting real personal or confidential information into traces and should report both loose and strict scores when making claims about system safety.

Acknowledgments

This work was supported by the independent research project of the Key Laboratory of Cognition and Decision Intelligence for Complex Systems.

References

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, and 1 others. 2024. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pages 20094–20136.

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*.

Diego F Cuadros, Abdoul-Aziz Maiga, Helen Meskhidze, and Andre Curtis-Trudel. 2026. Governed collaborative memory as artificial selection in llm-based multi-agent systems. *arXiv preprint arXiv:2605.04264*.

Guangya Hao, Yunbo Long, and Zhuokai Zhao. 2026. Self-evolving multi-agent systems via decentralized memory. *arXiv preprint arXiv:2605.22721*.

Andrea B Hollingshead. 2001. Cognitive interdependence and convergent expectations in transactive memory. *Journal of personality and social psychology*, 81(6):1080.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, and 1 others. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pages 23247–23275.

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

Hyunwoo Kim, Melanie Sclar, Xuhui Zhou, Ronan Bras, Gunhee Kim, Yejin Choi, and Maarten Sap. 2023. Fantom: A benchmark for stress-testing machine theory of mind in interactions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14397–14413.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008.

Haichang Li. 2025. Memory as a service (maas): Purpose-bound memory mediation for cooperative agents. *arXiv preprint arXiv:2506.22815*.

Jun Liu, Zhenglun Kong, Changdi Yang, Fan Yang, Tianqi Li, Peiyan Dong, Joannah Nanjekye, Hao Tang, Geng Yuan, Wei Niu, and 1 others. 2025. Rcr-router: Efficient role-aware context routing for multi-agent llm systems with structured memory. *arXiv preprint arXiv:2508.04903*.

- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2024. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pages 52989–53046.
- Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870.
- Olukunle Owolabi. 2026. Socialmembench: Are ai memory systems ready for social group settings? *arXiv preprint arXiv:2605.17789*.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.
- Joon Sung Park, Joseph O’Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulators of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.
- Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.
- Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. 2023. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*.
- Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. 2025. Membench: Towards more comprehensive evaluation on the memory of llm-based agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19336–19352.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.
- Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. 2024. Officebench: Benchmarking language agents across multiple applications for office automation. *arXiv preprint arXiv:2407.19056*.
- Daniel M Wegner. 1987. Transactive memory: A contemporary analysis of the group mind. In *Theories of group behavior*, pages 185–208. Springer.
- Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2024. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.
- Shanglin Wu, Yuyang Luo, Yueqing Liang, Kaiwen Shi, Yanfang Ye, Ali Payani, and Kai Shu. 2026. Scaling teams or scaling time? memory enabled lifelong learning in llm multi-agent systems. *arXiv preprint arXiv:2604.03295*.
- Jing Xu, Arthur Szlam, and Jason Weston. 2022. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 5180–5197.
- Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2026. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604.
- Jingbo Yang, Kwei-Herng Lai, Xiaowen Wang, Shiyu Chang, Yaar Harari, and Evgeniy Gabrilovich. 2026. Groupmembench: Benchmarking llm agent memory in multi-party conversations. *arXiv preprint arXiv:2605.14498*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
- Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. 2026. G-memory: Tracing hierarchical memory for multi-agent systems. *Advances in Neural Information Processing Systems*, 38:12988–13018.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.
- 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.
- Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, and

1 others. 2024. Sotopia: Interactive evaluation for social intelligence in language agents. In *International Conference on Learning Representations*, volume 2024, pages 40975–41019.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Daisy Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and 1 others. 2025. Multiagentbench: Evaluating the collaboration and competition of llm agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8580–8622.

A Supplementary Roadmap

The appendix is organized to make the benchmark’s diagnostic logic auditable. The main paper states the evaluation cell and headline findings; the appendix provides the operational details: what each probe diagnoses, how gold is computed, how the memory-only harness works, and how each adapter row should be interpreted.

Question	Where it is documented
Why is the benchmark diagnostic?	Appendix B locates the benchmark cell; Appendix C maps C1–C6 to memory operations and failure modes.
How are traces and gold labels built?	Appendix D specifies acceptance constraints and gold resolution.
How are systems compared fairly?	Appendix F describes the matched memory-only harness; Appendix G defines adapter rows.
Which capabilities do systems fix or break?	Appendix H provides per-competency and five-split details; Appendix I gives one representative failure trace.

Table 8: Reader map for the appendix.

B Benchmark Positioning

Table 9 summarizes the evaluation cell targeted by COLLABMEM. Existing long-memory benchmarks are essential for recall, update, temporal reasoning, and abstention; multi-agent task benchmarks are essential for planning and tool-use evaluation. COLLABMEM occupies a narrower but underexplored cell: *memory-only* diagnosis for role-governed teams, where every question has an access boundary, a target role, and an auditable source event.

Benchmark family	Roles	Mem.	Access	Prov.	Conflict	Compress
Single-user long mem-ory	–	✓	–	partial	✓	partial
Multi-agent task benchmarks	✓	–	partial	–	partial	–
Multi-party memory benchmarks	✓	✓	partial	✓	partial	partial
Multi-agent memory systems papers	✓	partial	partial	partial	partial	partial
COLLABMEM	✓	✓	✓	✓	✓	✓

Table 9: Benchmark positioning. Citations for the first three rows are given in the related-work section. COLLABMEM isolates collaborative memory operations under a fixed answering layer. “Partial” means that the benchmark family may contain relevant cases but does not make the dimension a controlled, score-bearing requirement for every scenario.

C Diagnostic Probe Semantics

Table 10 makes explicit what the main text compresses: a COLLABMEM probe is not merely a question-answer pair, but also a diagnosis of a named memory operation. This mapping is why

the benchmark can turn a score difference into an engineering target.

Probe	Operation	Gold anchor	Failure localized
C1	Directed sharing	source event plus authorized target role	no path for non-sensitive facts to cross role boundaries
C2	Cross-role consistency	current value after update or supersession	stale local state or unmerged role views
C3	Access control	exclusion set in the access matrix	global retrieval or shared memory leaking protected facts
C4	Transactive routing	authoritative writer or specialist role	missing “who knows what” directory
C5	Conflict resolution	contradiction edge, source authority, current tag	retrieval lacks provenance or current-versus-stale marking
C6	Collective compression	retained stale value and update rationale	compression drops why the current state changed

Table 10: Diagnostic semantics of C1–C6. The last column is the repair signal that a system builder should read from a failure.

D Dataset Construction Details

This section specifies the schema, acceptance gates, file format, and prompts needed to audit or regenerate the benchmark. A seed scenario is a tuple (R, A, E, D) : roles R , role-to-category access matrix A , event schedule E , and domain description D . Events are authored before dialogue rendering, so gold labels are never inferred from the rendered text. Each scenario must satisfy the structural constraints in Table 11.

Requirement	Acceptance condition and diagnostic purpose
Directed sharing	A non-sensitive fact is observed by a strict subset of roles and later queried from another authorized role; this prevents C1 from becoming local recall.
Cross-role consistency	An entity-attribute pair is updated or superseded after earlier exposure; this forces team-current state rather than a bag of facts.
Access boundary	A sensitive event has an explicit exclusion set and unauthorized target-role probe; this makes leakage measurable.
Transactive routing	A topic has an authoritative role without exposing every underlying fact; this tests “who knows what”.
Conflict chain	A contradiction points to a prior event and contains source/credibility metadata; this grounds C5 in authority rather than recency alone.
Compression pressure	A probe requires retaining why a current state changed after stale values are compressed; this separates retrieval from budget-aware retention.

Table 11: Scenario-level acceptance constraints. Each released core scenario contains all six structures, allowing one scenario to generate probes across all six competencies.

Generation pipeline. The released scripts implement four deterministic stages. First, `01_build_scenarios.py` materializes the seed scenarios with roles, access categories, and event schedules. Second, `02_run_simulation.py` renders traces while preserving the event schedule. Third,

03_generate_ga.py creates one or more probes for each eligible event and target role. Fourth, 04_verify.py and 14_build_verification_sample.py audit gold resolution against the post-supersession event map. The scaled, adversarial, and compositional splits reuse this same gold-bearing structure and change either the number of scenarios or the probe transformation.

Gold resolution. For value probes, the resolver selects the latest non-superseded value for the relevant (entity, attribute) chain after applying contradiction and deprecation links. For C3, the resolver additionally checks whether the target role is excluded by the access matrix. For C4, the gold is the authoritative writer or specialist role. For C6, the gold includes the retained value plus the stale state needed to justify why the current state changed. This design allows COLLABMEM to audit every probe mechanically.

E Dataset Format and Example

Table 12 lists the released artifact files. The same structure is used for the packaged reproduced run and for new runs produced by run_full_rebuild.py.

File	Contents
scenarios.json	Roles, access matrix, domain, sessions, and event schedule.
traces.json	Rendered trace, observed events, operations, turns, seed, and simulator stamp.
probes.json	Probe id, competency, target role, question, gold, protected flag, source event, and meta-data.
verification/	Gold audit, annotator sample sheets, pre-screening metadata, and agreement summary.

Table 12: Released artifact layout. Scenario, trace, and probe files live under each split directory.

Example probe record, shortened

```
{
  "scenario_id": "sde-api-...-01",
  "competency": "C1_directed_sharing",
  "target_agent": "PM",
  "question": "As PM, what is the current
users endpoint id format? State the
source.",
  "gold_answer": "ULID",
  "source_event_id": "sdel_f1",
  "metadata": {"acquirer": ["Backend"],
  "attribute": "id_format"}
}
```

Table 13: A shortened probe record from the core split. The full JSON also stores trace id, protected flag, and other metadata needed for audit scripts.

F Evaluation Harness and Prompts

Every system is evaluated through the same memory-only harness. The harness ingests a trace, queries the memory implementation for the target role’s context, and passes that context to a fixed answerer. This is the main fairness constraint: systems differ in stored and retrieved memory, not in the LLM that verbalizes the answer.

Step	Fixed protocol
Replay	Feed the same ordered trace to each memory implementation.
Retrieve	Query the implementation for the target role’s memory context.
Answer	Use the same answerer prompt and backbone for every row.
Judge	Apply loose semantic grading or strict value+source+temporal grading.
Aggregate	Report per-competency and overall scores; core CIs use scenario-cluster bootstrap.

Table 14: Matched memory-only evaluation protocol. This is the procedural basis for the paper’s diagnostic claims.

Answerer skeleton. The loose answerer receives the target role, competency, question, and memory context, then is instructed to answer only from the provided context and to abstain when the context is insufficient. The strict answerer receives the same fields but must separately output (i) value, (ii) authoritative source role or writer, (iii) temporal or update evidence, and (iv) a confidence note. The strict judge then requires exact value support, correct provenance, and no unsupported leakage.

Judge calibration. Loose grading is semantic and accepts paraphrases that preserve the gold value. Strict grading is evidence-aware: a correct value without source and temporal evidence is not sufficient for full credit. The Oracle row is run through the identical answerer and judge; it is a calibration ceiling, not a competing memory architecture.

G System Adapter Details

Table 15 gives the adapter-level details for the systems evaluated in the main paper. The official SDK rows are deliberately minimal memory-primitive evaluations: they test what a common memory API recovers under a uniform harness, not the best possible end-to-end application built around that product. The *inspired* rows are separate controlled mechanism baselines and should not be read as official product measurements.

System	State / boundary	Diagnostic note
NoMem	Empty context.	Lower bound for answerer prior knowledge.
FullShare	Complete shared log; no role boundary.	Availability upper control with expected C3 leakage.
RAG-Shared	Top- k lexical retrieval over one shared store.	Raw retrieval without access-aware memory.
Mem0-inspired	Scoped extracted facts; role plus non-sensitive shared tier.	Controlled mechanism baseline under the benchmark policy.
Letta-inspired	Core/archival tiers; role plus non-sensitive shared tier.	Controlled mechanism baseline under the fixed answerer.
A-Mem-inspired	Linked notes; isolated per role.	Controlled mechanism baseline for linked-note retrieval.
Mem0 SDK	Official mem0ai SDK; role and role+team namespaces.	SDK evaluation; no source modification.
Letta SDK	Official archival-memory interface; per-role archive plus optional team archive.	SDK evaluation; team archive row enters the main table.
A-Mem SDK	Released AgenticMemorySystem implementation; one persistent client per role.	Single-tenant default adapted through persistent-client isolation.
TMS++	Role-filtered graph with writer registry and current-value tag.	Proposed memory layer; same answerer as all rows.

Table 15: Adapter-level evaluation details. *Inspired* rows are controlled mechanism baselines; SDK rows are memory-only evaluation protocols, not exhaustive native multi-agent deployments.

H Additional Results

Figure 4 provides a per-competency view of the core scorecard. The main text keeps the numeric results in Table 4; the appendix plot highlights the same pattern: global sharing improves C1/C2 but hurts C3, shared archival memory is a strong practical primitive, and the proposed role/provenance graph improves C1–C5 while C6 remains difficult.

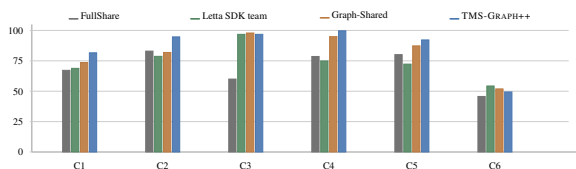


Figure 4: Per-competency loose accuracy for representative rows. Darker bars emphasize the diagnostic pattern: each architecture fixes or breaks different memory operations.

System	Core loose	Core strict	Adversarial	Composit.	Scaled	Avg.
<i>Trivial controls</i>						
NoMem	18.6	18.4	29.0	22.5	18.6	21.4
FullShare	69.1	31.0	58.0	37.8	67.6	52.7
RAG-Shared	60.7	25.2	50.7	42.4	56.9	47.2
<i>Controlled mechanism baselines</i>						
Mem0-inspired store	74.6	30.0	63.2	39.5	71.5	55.7
Letta-inspired tiers	74.4	30.1	63.6	38.4	71.1	55.5
A-Mem-inspired notes	36.0	21.5	41.4	37.0	35.0	34.2
Graph-Shared	81.2	53.0	73.2	49.5	78.7	67.1
TMS-Graph++	85.7	52.9	74.2	52.1	82.9	69.6
<i>Official-SDK evaluation rows</i>						
Mem0 SDK (role)	21.2	20.2	34.6	22.4	20.9	23.9
Letta SDK (role archive)	36.3	23.5	39.0	42.7	35.8	35.5
Letta SDK (role+team archive)	74.3	40.4	65.4	47.0	73.9	60.2
A-Mem SDK (role-isolated)	36.0	24.1	38.6	40.9	33.9	34.7
<i>Calibration Oracle</i>						
	87.5	68.7	75.2	51.2	87.0	73.9

Table 16: Full five-split table repeated for appendix readability. The body uses this table to support the benchmark-level claim; the appendix keeps it near adapter details and failure-mode discussion.

I Qualitative Failure Example

A representative C1/C5 failure occurs in the API-migration scenario. Backend first observes that the user-id format is `UUIDv4`; later, Backend observes the authoritative correction to `ULID`. The PM is authorized to know the current API contract but did not directly observe either message. Isolated role memory fails because the PM’s store lacks both events. FullShare can answer the value but cannot demonstrate that the access-control mechanism would withhold unrelated sensitive events in the same trace. TMS-GRAPH++ succeeds because the PM view includes visible Backend events, marks the superseding `ULID` edge as `[CURRENT]`, and preserves Backend as the source. The same trace therefore yields a directed-sharing probe, a consistency probe, a conflict-resolution probe, and a strict provenance test.