BEWARE OF MEMORY-INDUCED OVERCONFIDENCE: COMMITMENT–COMPETENCE CALIBRATION FOR MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Model-based Multi-Agent Systems have demonstrated impressive capabilities in solving complex tasks through role specialization, memory sharing, and collaborative reasoning. To further enhance their long-term performance, recent studies have introduced self-evolution mechanisms, where agents summarize reflections from historical interactions and reuse them in subsequent tasks. However, we observe that self-evolution does not always lead to consistent improvement: in some cases, apparently relevant memories may provide little benefit for the current task while repeatedly steering agents toward the same ineffective strategy. The key insight is that memory injection has two distinct effects: memory commitment, which characterizes how strongly memory influences behavior by reducing uncertainty about how to act, and memory competence, which estimates how reliably the selected memories support successful execution. These effects can be misaligned: memory may impose a clear and consistent behavioral direction while exhibiting low historical execution reliability. We identify this failure mode as memory-induced overconfidence. To mitigate this challenge, we propose C^2Mem, a memory calibration framework that dynamically attenuates apparently relevant memories that exhibit low execution utility. Experiments across multiple benchmarks show that C^2Mem reduces the propagation of non-beneficial memory guidance, mitigates memory-induced overconfidence, and improves the stability and reliability of multi-agent systems. Our code is available at `https://anonymous.4open.science/r/c2mem-27F3`.

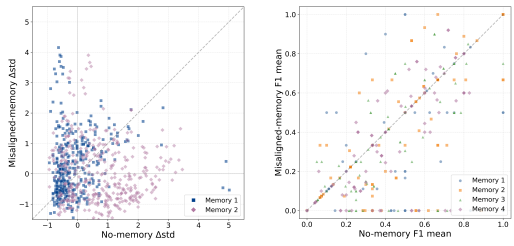
1 INTRODUCTION

Large language model-based multi-agent systems (MAS) have emerged as a promising paradigm for solving complex tasks through division of labor, information exchange, tool use, and collaborative reasoning (Zhang et al., 2025a). To further improve their long-term performance, recent studies have introduced self-evolution mechanisms that allow agents to summarize experiences from interactions and reuse them in subsequent tasks (Wang et al., 2025b). As experience accumulates, however, system reliability depends not only on whether the current reasoning process is correct, but also on whether historical memories can be reliably selected, reused, and updated (Ma et al., 2026a).

Existing memory mechanisms commonly assume that experiences semantically related to the current task provide useful guidance (Xu et al., 2025). Relevance, however, does not necessarily imply utility. Historical memories may contain erroneous reflections (Li et al., 2024), outdated strategies (Fu et al., 2026; Chao et al., 2026), or conclusions that hold only under specific contexts (Wu et al., 2025; Hu et al., 2026; Wu et al., 2026). When repeatedly retrieved, such memories may not only impair task performance but also reinforce the same unproductive strategy (Chen et al., 2024b). In other words, memory may reduce the behavioral uncertainty exhibited by the system without correspondingly improving execution reliability.

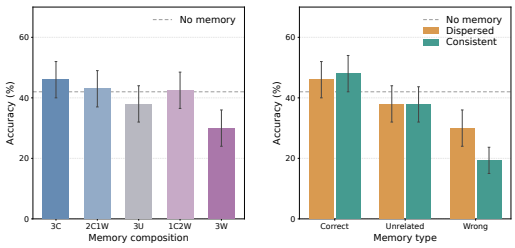
Memory not only provides information for the current task but also guides execution with varying strength. Figure 1 shows that its effects on execution behavior and answer performance do not always align: memory can change how a task is executed without improving the answer. Figure 2 further shows that consistent incorrect memories are more harmful than dispersed incorrect memories.

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



(a) Behavioral variability. (b) Task performance.

Figure 1: Effects of semantically relevant memories of uncertain utility. (a) Execution-length variability, measured as the standard deviation over five runs per task, with and without memory. (b) Answer F1 for different injected memories relative to the no-memory baseline.



(a) Memory quality. (b) Memory consistency.

Figure 2: Effects of memory quality and consistency on task performance. (a) Accuracy with different mixtures of three correct (C), unrelated (U), and wrong (W) memories. (b) Accuracy when memories of the same type provide dispersed or consistent guidance.

Thus, clear and consistent guidance is not necessarily reliable: memory can strongly steer execution in the wrong direction. We therefore distinguish memory guidance strength (**commitment**) from historically estimated execution reliability (**competence**).

Motivated by these observations, we decompose the influence of memory into two dimensions. The first is **memory commitment**, which characterizes the strength of support that retrieved memories provide for a coherent behavioral direction, estimated jointly from total memory support and structural coherence. The second is **memory competence**, which estimates the execution reliability of similar behavioral patterns from historical outcomes. These two dimensions need not be aligned: a memory set can strongly support a particular behavioral direction even when its estimated competence remains below the insufficient-memory-support reference. We define this mismatch between strong guidance and low reliability as **memory-induced overconfidence**.

To mitigate this problem, we propose **C^{2}Mem**, a memory calibration framework for multi-agent systems. C^{2}Mem organizes execution experiences in a hierarchical behavioral tree and maintains a node-level reliability posterior based on historical successes and failures. During execution, it estimates memory commitment from total memory support and the structural consistency of candidate nodes in the behavioral tree. It then estimates memory competence by aggregating the posterior reliability of the retrieved nodes and assesses overconfidence by comparing this estimate with the insufficient-memory-support baseline. When commitment is strong and posterior evidence confidently places aggregated competence below the insufficient-memory-support reference, C^{2}Mem downweights unreliable nodes before injection, reducing their influence on decisions.

This mechanism induces a natural evolutionary process over memory. Repeated unsuccessful exposure lowers a node's estimated competence. When this creates a competence deficit relative to the insufficient-support reference, calibration can attenuate the node's influence in subsequent retrieval contexts. Once their weights fall below the memory injection threshold, these memories cease to influence subsequent execution and are gradually marginalized without requiring explicit deletion. C^{2}Mem therefore focuses not only on accumulating more experience, but also on preventing non-beneficial experiences from being repeatedly retrieved, propagated, and reinforced over time.

Our main contributions are summarized as follows:

- To the best of our knowledge, this work is the first to jointly model **memory commitment** and **memory competence** in multi-agent systems, and to characterize **memory-induced overconfidence** as high commitment without corresponding competence.
- We propose a hierarchical behavioral tree to jointly model these two dimensions of memory: memory commitment is characterized by retrieval support and structural consistency, while memory competence is estimated from historical success and failure evidence.
- Extensive experiments across multiple multi-agent tasks demonstrate that C^{2}Mem achieves state-of-the-art performance.

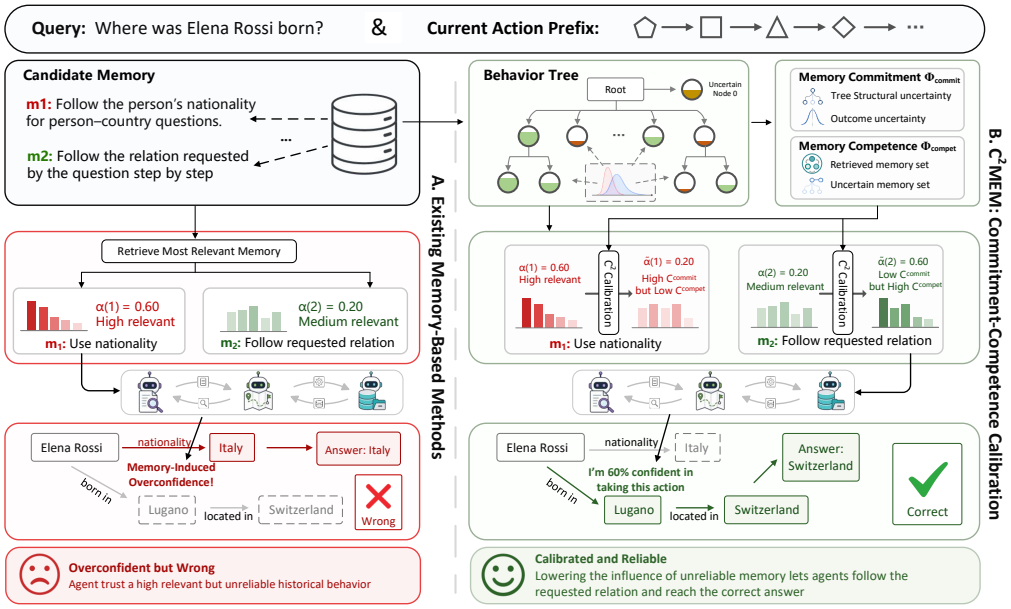


Figure 3: Overview of C^{2}Mem. Left: existing memory-based methods may retrieve a highly relevant but unreliable memory, causing memory-induced overconfidence and leading to a wrong answer. Right: C^{2}Mem calibrates memory by jointly modeling commitment and competence, reducing unreliable memory influence and improving reliability.

2 Methodology

Given a task x_q, let $\mathcal{F}_t$ denote the execution prefix at step t, containing preceding states, actions, observations, and feedback. With the LLM agents frozen, C^{2}Mem retrieves and calibrates behavioral memory during execution, then extracts reusable experience from the completed trajectory and updates memory reliability using the final outcome.

2.1 Behavioral Memory Representation and Retrieval

C^{2}Mem organizes historical experience in a behavioral tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}_\mathcal{T})$, whose depth progresses from general collaboration patterns to specific execution strategies. Each ordinary node $v = (M_v, c_v, a_v, b_v)$ stores reusable experience M_v, its behavioral summary c_v, and success/failure evidence (a_v, b_v), defining

$$\theta_v \sim \mathrm{Beta}(a_v, b_v), \qquad \mu_v = \mathbb{E}[\theta_v] = \frac{a_v}{a_v + b_v}. \tag{1}$$

An analogous unknown node $v = 0$ represents insufficient memory support, with $\mu_0 = a_0/(a_0 + b_0)$ serving as the competence reference. Appendix A.2 details behavioral encoding, experience insertion, node splitting, and posterior inheritance.

Because nodes group behaviorally similar experiences across tasks, at step t we encode the execution prefix and node summaries as $\mathbf{h}_t = \phi_{\mathrm{beh}}(\mathcal{F}_t)$ and $\mathbf{z}_v = \phi_{\mathrm{beh}}(c_v)$, respectively, and compute

$$s_t(v) = \frac{\mathbf{h}_t^\top \mathbf{z}_v}{\|\mathbf{h}_t\|_2 \|\mathbf{z}_v\|_2}. \tag{2}$$

The top-K nodes form $\mathcal{R}_t$, and $\widehat{\mathcal{R}}_t = \mathcal{R}_t \cup \{0\}$. Given $\mathcal{F}_t$ and the retrieved node summaries, an LLM-based evaluator produces $\boldsymbol{\alpha}_t \in \Delta(\widehat{\mathcal{R}}_t)$, where $\alpha_t(v)$ measures behavioral support and $\alpha_t(0)$ represents insufficient memory support. The total ordinary-memory support is

$$m_t = \sum_{v \in \mathcal{R}_t} \alpha_t(v) = 1 - \alpha_t(0). \tag{3}$$

2.2 MEMORY COMMITMENT

Memory commitment combines total memory support with structural coherence to quantify guidance toward compatible behavioral directions.

When $m_t > 0$, we define the conditional distribution over the ordinary memory nodes as

$$\bar{\alpha}_t(v) = \frac{\alpha_t(v)}{m_t}, \qquad v \in \mathcal{R}_t. \tag{4}$$

We measure structural inconsistency through the weighted pairwise tree distance:

$$U_{\text{tree}}^{(t)} = \sum_{u,v \in \mathcal{R}_t} \bar{\alpha}_t(u)\bar{\alpha}_t(v) d_{\mathcal{T}}(u,v). \tag{5}$$

Here, the normalized tree distance $d_{\mathcal{T}}(u,v) \in [0,1]$, with $d_{\mathcal{T}}(v,v) = 0$, operationalizes behavioral incompatibility: nearby nodes provide more compatible guidance.

We define commitment as

$$\Phi_t^{\text{commit}} = m_t \left(1 - U_{\text{tree}}^{(t)}\right). \tag{6}$$

We set $\Phi_t^{\text{commit}} = 0$ when $m_t = 0$; since both factors lie in $[0,1]$, so does commitment. A high score indicates substantial memory support for a coherent behavioral direction. Support concentrated on one node or distributed across structurally compatible nodes yields high commitment, while support spanning incompatible nodes lowers it.

2.3 MEMORY COMPETENCE

Behavioral nodes group experiences with similar execution patterns, and each node's competence estimates the historical task success rate of its pattern from exposure-weighted outcomes. Each retrieved node $v \in \mathcal{R}_t$ has latent competence θ_v and posterior mean μ_v. When $m_t > 0$, we aggregate these estimates using the conditional memory distribution $\{\bar{\alpha}_t(v)\}_{v \in \mathcal{R}_t}$ and define step-level memory competence as

$$\Phi_t^{\text{compet}} = \sum_{v \in \mathcal{R}_t} \bar{\alpha}_t(v)\mu_v. \tag{7}$$

This convex combination estimates the historical execution reliability of the selected behavioral memories. We use μ_0 as an internal reference for insufficient memory support. Appendix D.4 specifies the additional comparability conditions under which a node-level competence deficit implies that injecting the memory reduces conditional task success.

To account for posterior uncertainty, we define

$$c_t^- = \Pr\left(\sum_{v \in \mathcal{R}_t} \bar{\alpha}_t(v)\theta_v < \theta_0 \;\middle|\; \mathcal{D}_t\right), \tag{8}$$

where $\mathcal{D}_t$ denotes the accumulated execution evidence. This probability is approximated by jointly sampling from the Beta posteriors of the retrieved behavioral nodes and the unknown node. When $m_t = 0$, no ordinary memory is used, so memory competence is not evaluated and the overconfidence risk is zero.

2.4 COMMITMENT–COMPETENCE CALIBRATION

Memory-induced overconfidence arises when strong, coherent memory guidance has estimated competence below the insufficient-support reference. For $m_t > 0$, we quantify this risk as

$$O_t = \Phi_t^{\text{commit}} \left[\mu_0 - \Phi_t^{\text{compet}}\right]_+ c_t^-, \quad [x]_+ = \max(0,x). \tag{9}$$

This risk jointly captures the estimated strength and coherence of memory-supported guidance, the extent to which the competence of the retrieved memories falls below the reference competence μ_0, and the posterior confidence that this competence ordering is supported by the accumulated

evidence. Accordingly, O_t is large only when memory strongly supports a coherent behavioral direction, its aggregated competence is lower than μ_0, and the posterior evidence provides high confidence in this ordering.

Based on the estimated overconfidence risk, C^{2}Mem calibrates the retrieved memory nodes before memory injection. For each ordinary node $v \in \mathcal{R}_t$, we define its node-level contribution to the competence deficit as

$$\delta_t(v) = \frac{\bar{\alpha}_t(v)[\mu_0 - \mu_v]_+}{\sum_{u \in \mathcal{R}_t} \bar{\alpha}_t(u)[\mu_0 - \mu_u]_+ + \epsilon}, \tag{10}$$

where $\epsilon > 0$ is a small constant for numerical stability, and we set $\delta_t(0) = 0$.

We calibrate memory weights by solving

$$\widetilde{\boldsymbol{\alpha}}_t = \underset{p \in \Delta(\widehat{\mathcal{R}}_t)}{\arg\min} \left\{ D_{\mathrm{KL}}(p \| \boldsymbol{\alpha}_t) + \kappa O_t \sum_{v \in \widehat{\mathcal{R}}_t} p(v)\delta_t(v) \right\}, \tag{11}$$

Here, $\Delta(\widehat{\mathcal{R}}_t)$ is the probability simplex and $\kappa \geq 0$ controls calibration strength. The KL term preserves retrieval support, while the risk-scaled penalty suppresses nodes with larger deficit contributions. Nodes with zero original support retain zero weight.

This problem has the unique solution

$$\widetilde{\alpha}_t(v) = \frac{\alpha_t(v) \exp\left(-\kappa O_t \delta_t(v)\right)}{\sum_{u \in \widehat{\mathcal{R}}_t} \alpha_t(u) \exp\left(-\kappa O_t \delta_t(u)\right)}. \tag{12}$$

The derivation appears in Appendix D.5. Calibration preserves $\boldsymbol{\alpha}_t$ when $O_t = 0$ or $m_t = 0$; otherwise, larger deficit contributions incur stronger relative penalties, while the unknown node remains unpenalized. The calibrated weights govern memory injection and deactivation under the underlying MAS protocol (Appendix A.5).

2.5 Experience Updating

After task completion, the reflection module extracts reusable experience and its underlying behavioral pattern from the execution trajectory. The experience is then inserted into the behavior node that best matches the extracted pattern. Nodes containing sufficiently heterogeneous experiences are further split into behaviorally distinct child nodes.

For task q with success label $y_q \in \{0, 1\}$, we aggregate calibrated weights across execution steps into node exposures $\rho_q(v)$, normalized such that $\sum_v \rho_q(v) = 1$, and update

$$a_v \leftarrow a_v + \rho_q(v) y_q, \qquad b_v \leftarrow b_v + \rho_q(v)(1 - y_q). \tag{13}$$

Thus, nodes whose memories receive greater cumulative weight obtain more evidence from the final task outcome. The unknown node is updated in the same manner according to its accumulated weight, representing the portion of the execution not supported by existing behavioral memory.

Unsuccessful exposure lowers a node's posterior competence estimate. In subsequent retrieval contexts, calibration can reduce the node's relative weight when it contributes to the competence deficit. Memories whose calibrated weights fall below the injection threshold are excluded from the current execution context, while remaining available for future retrieval.

The update estimates historical task success weighted by each node's exposure and admits a fractional Beta–Bernoulli interpretation, with exposure controlling the amount of evidence contributed by each task. For a fixed node, when exposure–outcome pairs are i.i.d. with positive expected exposure, the estimate converges almost surely to this exposure-weighted reliability, and the generalized posterior concentrates at the same target. Definitions, derivations, and proofs are provided in Appendices D.1–D.3; the training and evaluation protocol is described in Appendix A.3.

MAS	Memory	NQ	BQ	HotpotQA	FEVER	MATH	Avg.
AutoGen	No-memory	72.52$_{\downarrow 0.00}$	82.36$_{\downarrow 0.00}$	40.58$_{\downarrow 0.00}$	65.84$_{\downarrow 0.00}$	75.40$_{\downarrow 0.00}$	67.34$_{\downarrow 0.00}$
	Voyager	74.59$_{\uparrow 2.07}$	86.19$_{\uparrow 3.83}$	36.79$_{\downarrow 3.79}$	55.54$_{\downarrow 10.30}$	75.86$_{\uparrow 0.46}$	65.79$_{\downarrow 1.55}$
	MemoryBank	73.26$_{\uparrow 0.74}$	81.97$_{\downarrow 0.39}$	41.99$_{\uparrow 1.41}$	69.01$_{\uparrow 3.17}$	69.13$_{\downarrow 6.27}$	67.07$_{\downarrow 0.27}$
	Generative	74.35$_{\uparrow 1.83}$	85.82$_{\uparrow 3.46}$	46.14$_{\uparrow 5.56}$	68.44$_{\uparrow 2.60}$	<u>80.10</u>$_{\uparrow 4.70}$	70.97$_{\uparrow 3.63}$
	AMA	<u>75.41</u>$_{\uparrow 2.89}$	<u>86.47</u>$_{\uparrow 4.11}$	43.89$_{\uparrow 3.31}$	65.68$_{\downarrow 0.16}$	74.26$_{\downarrow 1.14}$	69.14$_{\uparrow 1.80}$
	G-Memory	72.25$_{\downarrow 0.27}$	85.24$_{\uparrow 2.88}$	<u>47.91</u>$_{\uparrow 7.33}$	**73.21**$_{\uparrow 7.37}$	79.59$_{\uparrow 4.19}$	<u>71.64</u>$_{\uparrow 4.30}$
	C^{2}Mem	**76.66**$_{\uparrow 4.14}$	**87.74**$_{\uparrow 5.38}$	**50.64**$_{\uparrow 10.06}$	<u>72.06</u>$_{\uparrow 6.22}$	**81.67**$_{\uparrow 6.27}$	**73.75**$_{\uparrow 6.41}$
LLM-Debate	No-memory	69.10$_{\downarrow 0.00}$	84.86$_{\downarrow 0.00}$	36.45$_{\downarrow 0.00}$	54.37$_{\downarrow 0.00}$	58.06$_{\downarrow 0.00}$	60.57$_{\downarrow 0.00}$
	Voyager	72.22$_{\uparrow 3.12}$	<u>89.23</u>$_{\uparrow 4.37}$	37.95$_{\uparrow 1.50}$	56.03$_{\uparrow 1.66}$	55.30$_{\downarrow 2.76}$	62.15$_{\uparrow 1.58}$
	MemoryBank	71.28$_{\uparrow 2.18}$	85.17$_{\uparrow 0.31}$	41.33$_{\uparrow 4.88}$	67.95$_{\uparrow 13.58}$	59.05$_{\uparrow 0.99}$	64.96$_{\uparrow 4.39}$
	Generative	<u>73.52</u>$_{\uparrow 4.42}$	88.00$_{\uparrow 3.14}$	44.61$_{\uparrow 8.16}$	66.16$_{\uparrow 11.79}$	60.15$_{\uparrow 2.09}$	66.49$_{\uparrow 5.92}$
	AMA	71.24$_{\uparrow 2.14}$	86.39$_{\uparrow 1.53}$	41.82$_{\uparrow 5.37}$	<u>69.32</u>$_{\uparrow 14.95}$	64.19$_{\uparrow 6.13}$	66.59$_{\uparrow 6.02}$
	G-Memory	73.22$_{\uparrow 4.12}$	87.48$_{\uparrow 2.62}$	<u>45.39</u>$_{\uparrow 8.94}$	68.02$_{\uparrow 13.65}$	<u>65.30</u>$_{\uparrow 7.24}$	<u>67.88</u>$_{\uparrow 7.31}$
	C^{2}Mem	**73.66**$_{\uparrow 4.56}$	**90.98**$_{\uparrow 6.12}$	**47.32**$_{\uparrow 10.87}$	**70.63**$_{\uparrow 16.26}$	**70.71**$_{\uparrow 12.65}$	**70.66**$_{\uparrow 10.09}$
MacNet	No-memory	71.64$_{\downarrow 0.00}$	83.13$_{\downarrow 0.00}$	41.90$_{\downarrow 0.00}$	62.08$_{\downarrow 0.00}$	<u>78.04</u>$_{\downarrow 0.00}$	67.36$_{\downarrow 0.00}$
	Voyager	66.92$_{\downarrow 4.72}$	81.64$_{\downarrow 1.49}$	39.84$_{\downarrow 2.06}$	74.38$_{\uparrow 12.30}$	76.14$_{\downarrow 1.90}$	67.78$_{\uparrow 0.42}$
	MemoryBank	72.11$_{\uparrow 0.47}$	81.90$_{\downarrow 1.23}$	42.00$_{\uparrow 0.10}$	72.93$_{\uparrow 10.85}$	72.73$_{\downarrow 5.31}$	68.33$_{\uparrow 0.97}$
	Generative	<u>73.19</u>$_{\uparrow 1.55}$	<u>83.56</u>$_{\uparrow 0.43}$	47.84$_{\uparrow 5.94}$	<u>74.52</u>$_{\uparrow 12.44}$	74.38$_{\downarrow 3.66}$	<u>70.70</u>$_{\uparrow 3.34}$
	AMA	72.82$_{\uparrow 1.18}$	80.37$_{\downarrow 2.76}$	37.72$_{\downarrow 4.18}$	70.09$_{\uparrow 8.01}$	74.30$_{\downarrow 3.74}$	67.06$_{\downarrow 0.30}$
	G-Memory	71.46$_{\downarrow 0.18}$	83.27$_{\uparrow 0.14}$	<u>47.95</u>$_{\uparrow 6.05}$	73.59$_{\uparrow 11.51}$	73.50$_{\downarrow 4.54}$	69.95$_{\uparrow 2.59}$
	C^{2}Mem	**74.08**$_{\uparrow 2.44}$	**85.77**$_{\uparrow 2.64}$	**50.55**$_{\uparrow 8.65}$	**75.72**$_{\uparrow 13.64}$	**80.76**$_{\uparrow 2.72}$	**73.38**$_{\uparrow 6.02}$

Table 1: Performance comparison of different memory mechanisms. The colored subscripts indicate absolute performance changes relative to the corresponding No-memory baseline. The best and second-best results within each MAS are highlighted **bold** and <u>underlined</u>, respectively.

3 Experiments

3.1 Experimental Setup

Benchmarks. We evaluate our method on five benchmarks covering diverse knowledge and reasoning tasks: Natural Questions (NQ) (Kwiatkowski et al., 2019) for open-domain question answering, BoolQ (BQ) (Clark et al., 2019) for binary question answering, HotpotQA (Yang et al., 2018) for multi-hop question answering, FEVER (Thorne et al., 2018) for fact verification, and MATH (Hendrycks et al., 2021) for mathematical reasoning. We evaluate task performance using Exact Match (EM) on all five benchmarks and report the results as percentages. Dataset descriptions are provided in Appendix F.1.

Baselines. We compare C^{2}Mem with five representative memory baselines, including Voyager (Wang et al., 2024), MemoryBank (Zhong et al., 2024), Generative (Park et al., 2023), AMA (Huang et al., 2026), and G-Memory (Zhang et al., 2025a). We additionally include a No-memory setting, in which no persistent memory is provided, to quantify the effects of memory injection. Baseline descriptions are provided in Appendix F.2.

MAS and LLM Backbones. We select three representative multi-agent frameworks to integrate with our method and the baselines, including AutoGen (Wu et al., 2024), LLM-Debate (Du et al., 2024), and MacNet (Qian et al., 2025). For instantiating these MAS frameworks, we adopt gpt-4o as the LLM backbone, accessed via OpenAI APIs. Descriptions of the multi-agent frameworks are provided in Appendix F.3.

3.2 Main Results

Table 1 compares C^{2}Mem with five representative memory baselines and a No-memory setting across three multi-agent systems and five benchmarks. C^{2}Mem achieves the highest average performance under all three MAS configurations, reaching 73.75 on AutoGen, 70.66 on LLM-Debate, and 73.38 on MacNet, corresponding to improvements of 6.41, 10.09, and 6.02 points over the respective

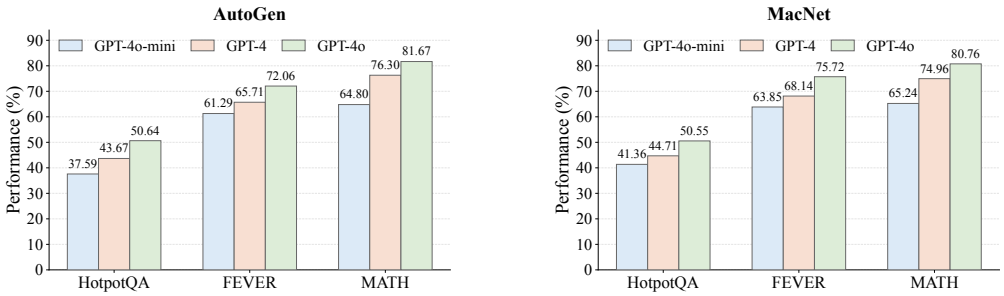


Figure 4: Performance comparison with different LLM backbones on HotpotQA, FEVER, and MATH.

Variant	AutoGen		MacNet	
	HotpotQA	FEVER	HotpotQA	FEVER
No Memory	40.58$_{\downarrow 0.00}$	65.84$_{\downarrow 0.00}$	41.90$_{\downarrow 0.00}$	62.08$_{\downarrow 0.00}$
w/o Calibration	46.98$_{\uparrow 6.40}$	68.29$_{\uparrow 2.45}$	46.20$_{\uparrow 4.30}$	67.31$_{\uparrow 5.23}$
w/o Commitment	49.56$_{\uparrow 8.98}$	71.30$_{\uparrow 5.46}$	49.17$_{\uparrow 7.27}$	74.21$_{\uparrow 12.13}$
C^{2}Mem	**50.64**$_{\uparrow 10.06}$	**72.06**$_{\uparrow 6.22}$	**50.55**$_{\uparrow 8.65}$	**75.72**$_{\uparrow 13.64}$

Table 2: Ablation study of commitment–competence calibration on AutoGen and MacNet. Colored subscripts indicate absolute performance changes relative to No Memory.

No-memory baselines. It consistently outperforms No-memory across all 15 MAS–benchmark combinations and ranks first or second in every setting. These results demonstrate that C^{2}Mem provides robust and generalizable improvements across diverse tasks and multi-agent interaction paradigms.

3.3 Performance Across LLM Backbones

We evaluate C^{2}Mem with GPT-4o-mini, GPT-4, and GPT-4o in AutoGen and MacNet on HotpotQA, FEVER, and MATH. Figure 4 reports task performance for each configuration. GPT-4o achieves the highest scores across all six framework–benchmark combinations. Performance also varies with the interaction framework: for example, GPT-4 achieves 76.30 on MATH with AutoGen and 65.24 with MacNet. These results characterize C^{2}Mem's performance across backbone and collaboration configurations.

3.4 Ablation Study

We compare C^{2}Mem with three variants: No Memory removes persistent memory, w/o Calibration injects retrieved memories without calibration, and w/o Commitment calibrates memory using only competence shortfall and posterior confidence. As shown in Table 2, w/o Calibration improves over No Memory by 4.60 points on average, demonstrating the benefit of behavior-centric memory retrieval. Competence-based calibration provides a further 3.87-point gain, confirming the importance of historical reliability. Adding commitment improves performance in all four settings, with an average gain of 1.18 percentage points over competence-only calibration. This supports using guidance strength to modulate reliability-based penalties. Section 3.5 examines competence along execution trajectories, and Section 3.6 analyzes its interaction with commitment.

3.5 Trajectory-level memory competence.

To assess whether memory competence reflects the reliability of selected memories, we track Φ_t^{compet} along execution trajectories and group them by final task outcome. As shown in Fig. 5, successful and failed tasks begin with similar competence levels but increasingly diverge during execution. Successful trajectories maintain high and gradually increasing competence, whereas failed trajectories progressively rely on less reliable memories. The pronounced terminal-stage separa-

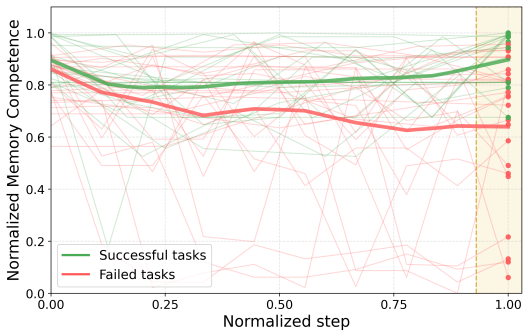


Figure 5: Trajectory-level evolution of memory competence for successful and failed tasks. Thick lines denote averages, and the shaded region marks the terminal execution stage.

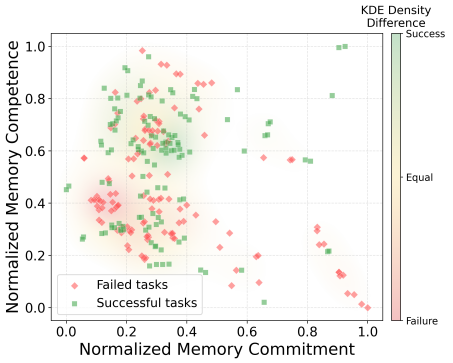


Figure 6: Joint distribution of memory commitment and competence. Markers show outcomes, while KDE colors indicate success- (green) and failure-dominant (red) regions.

tion indicates that Φ_t^{compet} provides an informative step-level reliability estimate and supports the motivation for commitment–competence calibration: strong memory-induced guidance does not necessarily imply reliable execution support.

3.6 Commitment–Competence Relationship.

Figure 6 shows that successful tasks generally have higher memory competence, while failures are more common at lower competence. Failures also appear where commitment is high and competence is low: the retrieved memories provide strong guidance despite low historical reliability. This pattern illustrates why guidance strength should be considered together with reliability. Strong guidance can help agents follow useful strategies, but it can also keep them following ineffective ones. C^{2}Mem therefore increases penalties for unreliable memories that provide strong guidance. Appendix E further analyzes commitment and competence from three perspectives: the temporal dynamics of competence, its outcome-conditioned discriminability, and the binned and continuous structure of the commitment–competence interaction.

3.7 Efficiency Analysis

Figure 7 compares task performance and token cost on HotpotQA. C^{2}Mem achieves the highest performance under both AutoGen and MacNet, using approximately 4.5×10^6 and 8.2×10^6 tokens, respectively. For comparison, G-Memory uses 4.9×10^6 and 8.1×10^6 tokens in the same frameworks. C^{2}Mem therefore achieves higher performance at a lower token cost under AutoGen and a slightly higher token cost under MacNet. Its additional computation includes LLM-based support evaluation, posterior comparison, and weight calibration.

3.8 Case Study

To illustrate how calibration changes multi-agent execution, we compare direct memory injection with C^{2}Mem on representative trajectories. Direct injection allows coherent but insufficiently supported guidance to persist, whereas C^{2}Mem uses commitment–competence calibration to adjust retrieved memory weights while preserving reliable guidance. Appendix B provides the full trajectories and step-by-step analysis.

4 Related Work

LLM-Based Multi-Agent Systems. LLM-based multi-agent systems solve tasks through role specialization, communication, and collaborative reasoning. Prior work has explored role-

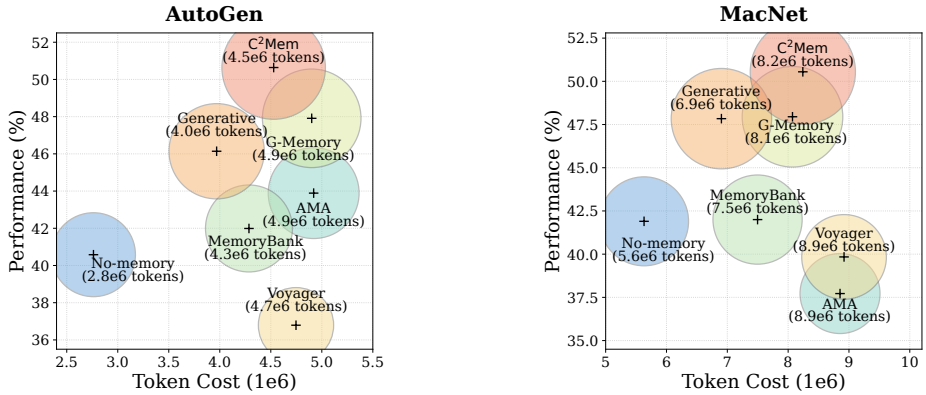


Figure 7: Token cost comparison with different MAS backbones on HotpotQA.

specialized software workflows (Hong et al., 2024; Qian et al., 2024), dynamic agent networks (Liu et al., 2024), optimizable interaction graphs, reflective collaboration, and adaptive routing (Zhuge et al., 2024; Bo et al., 2024; Wang et al., 2025a). Reliability studies have identified unstable debate and vulnerability to faulty agents (Smit et al., 2024; Huang et al., 2025), conformity and failure-attribution difficulties (Choi et al., 2025; Zhang et al., 2025b), and adversarial attacks and cascading uncertainty (Kavathekar et al., 2026; Chen et al., 2026). Despite diversity, these failures share a temporal boundary: they arise and propagate within a single task. Systems break this boundary. Once an unreliable experience is stored, repeated retrieval can carry the failure across tasks and turn it into persistent behavioral guidance. Our work studies this longer-term risk. C^{2}Mem distinguishes the strength of memory guidance from its historical reliability, enabling the system to identify and calibrate influential but insufficiently reliable memories.

Memory for LLM Agents. Memory mechanisms enable LLM agents to accumulate and reuse historical information and execution experience across tasks. Prior work extracts generalizable insights, constructs instruction manuals from environmental interaction, and induces reusable workflows for long-horizon execution (Zhao et al., 2024; Chen et al., 2024a; Sarch et al., 2024). Recent methods improve memory construction and retrieval through reflective refinement, autonomous semantic augmentation, and dynamically linked memory networks (Tan et al., 2025; Salama et al., 2025; Xu et al., 2025). Complementary work studies the structural organization of agent memory (Zeng et al., 2024). Memory also supports continual updates of agent-specific expertise during decentralized collaboration (Yang et al., 2025). These approaches treat memory mainly as an information-management problem: what to store, how to organize it, and what to retrieve. The difficulty begins after retrieval. Agents often follow retrieved experience, allowing inaccurate records to propagate errors across tasks (Xiong et al., 2026). Utility-aware retention estimates which experiences may remain useful (Ma et al., 2026b), but does not compare the strength of memory-induced guidance with its historical reliability. A relevant and repeatedly selected memory may therefore exert more influence than its evidence justifies. C^{2}Mem separates memory commitment from competence to identify and calibrate such high-commitment, low-competence memories.

5 Conclusion

In this work, we identify memory-induced overconfidence as a cross-task reliability failure in which retrieved memories impose strong and consistent guidance despite limited historical reliability. We propose C^{2}Mem, which separates memory commitment from competence, organizes experience in a hierarchical behavioral tree, and calibrates memory influence against an insufficient-memory-support reference. C^{2}Mem uses historical outcome evidence to attenuate retrieved memories that contribute to the estimated competence deficit. Across three multi-agent frameworks and five benchmarks, it outperforms the No-memory baseline in all 15 settings and achieves the highest average performance within each framework. These results support incorporating historical reliability and guidance strength into memory calibration.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation, method implementation, feedback on the research methodology, and the formulation and proof of the conditional interpretation of competence deficits in Appendix D.4. Additionally, we used generative AI tools to assist with literature organization and language polishing. We reviewed all AI-assisted work, including verifying and testing the code and checking translations for accuracy and consistency with the original meaning. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 2 describes the behavioral-memory representation, retrieval, commitment and competence measures, calibration rule, and exposure-weighted update. Section 3.1 identifies the benchmarks, memory baselines, multi-agent frameworks, and language-model backbone used in the experiments. Appendix A provides additional details on memory construction, posterior inheritance, memory injection, and the training and evaluation protocol, including the freezing of memory during test evaluation. Appendix D states the assumptions and provides proofs of the theoretical properties analyzed in this work. The abstract provides a link to an anonymous source-code repository. The repository contains additional implementation details and instructions for reproducing our experiments.

REFERENCES

Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. Reflective multi-agent collaboration based on large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 138595–138631. Curran Associates, Inc., 2024.

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

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. AutoManual: Constructing instruction manuals by LLM agents via interactive environmental learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024a.

Tiejin Chen, Huaiyuan Yao, Jia Chen, Evangelos E. Papalexakis, and Hua Wei. Every response counts: Quantifying uncertainty of LLM-based multi-agent systems through tensor decomposition. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 16204–16218, San Diego, California, United States, July 2026. Association for Computational Linguistics.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *Advances in Neural Information Processing Systems*, volume 37, pp. 130185–130213. Curran Associates, Inc., 2024b.

Min Choi, Keonwoo Kim, Sungwon Chae, and Sangyeop Baek. An empirical study of group conformity in multi-agent systems. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5123–5139, Vienna, Austria, July 2025. Association for Computational Linguistics.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 2024.

Yao Fu, Ran Qiu, Xinhe Wang, Jacob Sansom, Sathvika Ayyappa Prabhu, Huijie Tang, Jaekyeom Kim, Sungryull Sohn, and Honglak Lee. Beyond blind following: Evaluating robustness of LLM

agents under imperfect guidance. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6591–6618, Rabat, Morocco, 2026. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *The Fourteenth International Conference on Learning Representations*, 2026.

Jen-Tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Michael Lyu, and Maarten Sap. On the resilience of LLM-based multi-agent collaboration with faulty agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 26202–26226. PMLR, 2025.

Weiquan Huang, Zixuan Wang, Hehai Lin, Sudong Wang, Bo Xu, Qian Li, Beier Zhu, Linyi Yang, and Chengwei Qin. AMA: Adaptive memory via multi-agent collaboration. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3099–3120, San Diego, California, United States, 2026. Association for Computational Linguistics.

Ishan Kavathekar, Hemang Jain, Ameya Rathod, Ponnurangam Kumaraguru, and Tanuja Ganu. TAMAS: Benchmarking adversarial risks in multi-agent LLM systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 31238–31268, San Diego, California, United States, July 2026. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

Yanhong Li, Chenghao Yang, and Allyson Ettinger. When hindsight is not 20/20: Testing limits on reflective thinking in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3741–3753, Mexico City, Mexico, 2024. Association for Computational Linguistics.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*, 2024.

Weitao Ma, Xiaocheng Feng, Lei Huang, Xiachong Feng, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Bing Qin. Fine-Mem: Fine-grained feedback alignment for long-horizon memory management. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19666–19684, San Diego, California, United States, 2026a. Association for Computational Linguistics.

Wenquan Ma, Jiayan Nan, and WenLong Wu. What deserves memory: Adaptive memory distillation for LLM agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34789–34812, San Diego, California, United States, July 2026b. Association for Computational Linguistics.

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

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, Bangkok, Thailand, 2024. Association for Computational Linguistics.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations*, 2025.

Rana Salama, Jason Cai, Michelle Yuan, Anna Currey, Monica Sunkara, Yi Zhang, and Yassine Benajiba. MemInsight: Autonomous memory augmentation for LLM agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33136–33152, Suzhou, China, November 2025. Association for Computational Linguistics.

Gabriel Sarch, Lawrence Jang, Michael J. Tarr, William W. Cohen, Kenneth Marino, and Katerina Fragkiadaki. VLM agents generate their own memories: Distilling experience into embodied programs of thought. In *Advances in Neural Information Processing Systems*, volume 37, pp. 75942–75985, 2024.

Andries Petrus Smit, Nathan Grinsztajn, Paul Duckworth, Thomas D. Barrett, and Arnu Pretorius. Should we be going MAD? a look at multi-agent debate strategies for LLMs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 45883–45905. PMLR, 2024.

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. 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)*, pp. 8416–8439, Vienna, Austria, July 2025. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Song Wang, Zhen Tan, Zihan Chen, Shuang Zhou, Tianlong Chen, and Jundong Li. AnyMAC: Cascading flexible multi-agent collaboration via next-agent prediction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11555–11567, Suzhou, China, November 2025a. Association for Computational Linguistics.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025b.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen llm applications via multi-agent conversation. In *First Conference on Language Modeling*, 2024.

Zhaofen Wu, Hanrong Zhang, Fulin Lin, Wujiang Xu, Xinran Xu, Yankai Chen, Henry Peng Zou, Shaowen Chen, Weizhi Zhang, Xue Liu, Philip S. Yu, and Hongwei Wang. GAM: Hierarchical graph-based agentic memory for LLM agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34647–34664, San Diego, California, United States, 2026. Association for Computational Linguistics.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts LLM agents: An empirical study of experience-following behavior. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 623–645, San Diego, California, United States, July 2026. Association for Computational Linguistics.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025.

Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. AgentNet: Decentralized evolutionary coordination for LLM-based multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, 2018. Association for Computational Linguistics.

Ruihong Zeng, Jinyuan Fang, Siwei Liu, and Zaiqiao Meng. On the structural memory of LLM agents. *arXiv preprint arXiv:2412.15266*, 2024.

Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. G-Memory: Tracing hierarchical memory for multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76583–76599. PMLR, 2025b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, 2024.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767. PMLR, 2024.

A Additional Method and Evaluation Details

A.1 Supplementary Notation and Memory State

For each benchmark b and multi-agent system configuration m, we construct a separate behavioral-memory state from the corresponding training split and use that state for evaluation on the test split. Let

$$\mathcal{D}_b^{\text{train}} = \{(x_q, g_q)\}_{q=1}^{N_b^{\text{train}}}, \qquad \mathcal{D}_b^{\text{test}} = \{(x_i, g_i)\}_{i=1}^{N_b^{\text{test}}},$$

where x denotes a task instance and g denotes its benchmark-provided gold answer or gold class.

For a fixed pair (b, m), we write the memory state as

$$\mathcal{S}_{b,m} = (\mathcal{T}_{b,m}, \mathcal{M}_{b,m}, \mathcal{P}_{b,m}),$$

where $\mathcal{T}_{b,m}$ is the behavioral-tree topology, $\mathcal{M}_{b,m}$ contains the stored reusable experiences and behavioral summaries, and $\mathcal{P}_{b,m}$ contains the Beta-posterior parameters of the ordinary nodes and the unknown node. Following the main paper, the posterior mean competence of an ordinary node v and the reference competence of the unknown node are

$$\mu_v = \frac{a_v}{a_v + b_v}, \qquad \mu_0 = \frac{a_0}{a_0 + b_0}.$$

The unknown node represents insufficient support from the current ordinary memory set, and μ_0 serves as the reference competence in the calibration mechanism.

A.2 Behavioral Memory Construction

After completing a training task, the reflection module extracts a reusable experience and a behavioral-pattern summary from the execution trajectory. The experience records transferable execution knowledge, while the summary captures the associated planning, coordination, verification, or tool-use strategy. The behavioral tree groups experiences according to these execution patterns.

Behavioral representation and insertion. Let c_q be the behavioral-pattern summary extracted from task q, and let c_v be the summary of an existing ordinary node v. Using the behavioral encoder ϕ_{beh} from the main paper, which is instantiated with BGE-M3, define

$$z_q = \phi_{\text{beh}}(c_q), \qquad z_v = \phi_{\text{beh}}(c_v),$$

and measure their similarity by

$$s(q, v) = \frac{z_q^\top z_v}{\|z_q\|_2 \|z_v\|_2}.$$

The destination node is selected as

$$v_q^\star = \arg\max_{v \in V_{\text{ord}}} s(q, v),$$

where V_{ord} is the current set of ordinary nodes. The reusable experience and its behavioral summary are stored in $v_q^\star$. This behavior-centered assignment allows experiences with similar execution strategies across different task contents to share a node.

Node splitting and posterior inheritance. A node containing sufficiently heterogeneous experiences is divided into behaviorally distinct child nodes. Suppose a parent node v with posterior parameters (a_v, b_v) is split into J children $v_1, \ldots, v_J$. Let $n_j > 0$ be the number of parent experiences assigned to child v_j, and define

$$r_j = \frac{n_j}{\sum_{k=1}^{J} n_k}, \qquad \sum_{j=1}^{J} r_j = 1. \tag{A.1}$$

The child posteriors are initialized as

$$a_{v_j} = r_j a_v, \qquad b_{v_j} = r_j b_v. \tag{A.2}$$

The rule allocates accumulated evidence according to the experience mass of each child. Since $\sum_j r_j = 1$, it preserves the total evidence, and direct substitution shows that every child inherits the parent's posterior mean. Subsequent training trajectories update each child independently through main-text Eq. (13).

For example, consider a parent with $(a_v, b_v) = (10, 5)$ and 15 stored experiences. A split containing 10 and 5 experiences gives $r_1 = 2/3$ and $r_2 = 1/3$, and therefore

$$(a_{v_1}, b_{v_1}) = \left(\frac{20}{3}, \frac{10}{3}\right), \qquad (a_{v_2}, b_{v_2}) = \left(\frac{10}{3}, \frac{5}{3}\right).$$

Both children begin with posterior mean $2/3$, with evidence mass proportional to their assigned experiences.

A.3 Memory Evolution and Evaluation Protocol

For a training instance (x_q, g_q), the MAS produces an execution trajectory τ_q and a prediction $\widehat{g}_q$. The benchmark evaluator computes the success label

$$y_q = \mathrm{Eval}_b(\widehat{g}_q, g_q) \in \{0, 1\}.$$

The reflection module processes the completed trajectory and its task outcome to extract a reusable experience. During execution, calibrated memory weights are aggregated across steps to obtain the normalized node exposure $\rho_q(v) \in [0, 1]$, with

$$\sum_v \rho_q(v) = 1.$$

The posterior update follows main-text Eq. (13):

$$a_v \leftarrow a_v + \rho_q(v) y_q,$$
$$b_v \leftarrow b_v + \rho_q(v)(1 - y_q).$$

The unknown node is updated using the same exposure-weighted rule. Training instances therefore contribute success and failure evidence according to the memory support received along their trajectories.

After the training split is processed, let

$$\mathcal{S}^{\star}_{b,m} = \mathrm{TrainMemory}\left(\mathcal{D}^{\mathrm{train}}_b; m\right)$$

denote the resulting behavioral-memory state. During test evaluation, Eq. (13), experience insertion, node splitting, and all posterior updates are disabled. Test-time retrieval and calibration use this fixed state:

$$\mathcal{S}^{(i)}_{b,m} = \mathcal{S}^{\star}_{b,m} \qquad \text{for every test instance } i. \tag{A.3}$$

For a test input x_i and execution prefix $F_{i,t}$, the system computes the retrieved nodes, support distribution, memory commitment, memory competence, overconfidence risk, and calibrated weights from $\mathcal{S}^{\star}_{b,m}$. The resulting prediction can be written as

$$\widehat{g}_i = f_{b,m}\left(x_i; \mathcal{S}^{\star}_{b,m}, \xi_i\right),$$

where ξ_i denotes the stochasticity of the LLM and MAS execution protocol. The gold target g_i is then used to compute the reported evaluation metric. All compared memory mechanisms use the same benchmark splits, MAS configurations, backbone models, and evaluation instances.

A.4 Interpretation of Memory Commitment

Memory commitment measures the strength and coherence of the behavioral guidance supported by retrieved memories. At step t, main-text Eq. (6) combines two quantities. The ordinary memory mass

$$m_t = 1 - \alpha_t(0)$$

measures the support assigned to the retrieved ordinary nodes. Conditional on ordinary-memory support, the structural inconsistency $U_{\text{tree}}^{(t)}$ is the expected normalized tree distance between two independent draws from $\bar{\boldsymbol{\alpha}}_t$.

Because $d_{\mathcal{T}}(v, v) = 0$, main-text Eq. (5) can be written as

$$U_{\text{tree}}^{(t)} = \sum_{\substack{u,v \in \mathcal{R}_t \\ u \neq v}} \bar{\alpha}_t(u)\bar{\alpha}_t(v) d_{\mathcal{T}}(u, v).$$

This form shows that structural inconsistency depends jointly on the allocation of memory support and the distances between the supported nodes. Concentrating support on one node reduces the off-diagonal mass, while distributing support across structurally compatible nodes keeps the distance-weighted terms small.

A large Φ_t^{commit} indicates substantial ordinary-memory support for a structurally coherent behavioral direction. Memory competence complements this quantity with the historical execution reliability of the selected nodes. Their joint use identifies the high-commitment, low-competence regime associated with memory-induced overconfidence. Since $m_t \in [0, 1]$ and $U_{\text{tree}}^{(t)} \in [0, 1]$, commitment lies in $[0, 1]$. For a fixed conditional distribution and tree structure, it increases with m_t; for fixed m_t, it decreases as $U_{\text{tree}}^{(t)}$ increases.

A.5 Calibrated Injection and Selective Memory Deactivation

The calibrated distribution in main-text Eq. (12) determines the retrieved memories that enter the subsequent MAS context. Let $\tau_{\text{inj}} > 0$ be the memory injection threshold and define the active ordinary-memory set as

$$\mathcal{A}_t = \{v \in R_t : \widetilde{\alpha}_t(v) \geq \tau_{\text{inj}}\}.$$

Memories associated with nodes in $\mathcal{A}_t$ are injected according to the underlying MAS protocol. A node whose calibrated weight falls below the threshold remains outside the current execution context.

For ordinary nodes $u, v \in R_t$, main-text Eq. (12) gives

$$\frac{\widetilde{\alpha}_t(v)}{\widetilde{\alpha}_t(u)} = \frac{\alpha_t(v)}{\alpha_t(u)} \exp\left[-\kappa O_t \left(\delta_t(v) - \delta_t(u)\right)\right].$$

When $\delta_t(v) > \delta_t(u)$, increasing O_t produces stronger relative attenuation for node v. The calibration therefore assigns greater suppression to nodes contributing more strongly to the competence deficit within the same retrieval context.

Repeated unsuccessful exposure increases b_v through main-text Eq. (13) and lowers

$$\mu_v = \frac{a_v}{a_v + b_v}.$$

This increases $[\mu_0 - \mu_v]_+$ and strengthens the node's contribution to the calibration penalty. When its calibrated weight falls below τ_{inj} at step t, the corresponding memory is excluded from the current execution context. Its experience and posterior evidence remain stored in the behavioral tree and may be retrieved in subsequent contexts.

B Qualitative Case Studies

Figures 8 and 9 illustrate how retrieved memories affect an execution trajectory with and without commitment–competence calibration. The first case shows how coherent but insufficiently supported guidance can persist under direct memory injection. The second shows how C^{2}Mem attenuates an initially dominant, low-competence memory while preserving later guidance whose competence exceeds the insufficient-support reference.

Case Study: Failure with Direct Memory Injection

Question: Do the drinks Gibson and Zurracapote both contain gin?

Correct answer: No.

Starting context: The agent retrieves experience from earlier drink-comparison tasks.

Iteration 1

Retrieved memories: M1: "In earlier comparisons of related mixed drinks, first identifying the familiar drink's base spirit helped organize retrieval ... use that ingredient as a search cue for the less familiar drink."

M2: "Recipe descriptions differ in detail across sources, and regional variants may add ingredients ... inspect variants before treating a brief ingredient list as exhaustive."

Action: Lookup[Gibson (cocktail)]

Observation: Gibson is made with gin and vermouth.

Agent conclusion: "Gin is a useful lead for investigating the second drink."

Iteration 2

Retrieved memories: M1: "After extracting a base spirit from the first drink, combine it with the second drink's name to narrow retrieval ... compare recipe descriptions returned by the focused query."

M2: "A short description may cover only the standard recipe ... examine variant recipes before excluding an ingredient that appears in related drinks."

Action: Search[Zurracapote gin recipe]

Observation: The result describes a Spanish mixed drink similar to sangria; it does not confirm gin.

Agent conclusion: "A variant may use gin; inspect the drink before ruling that out."

Iteration 3

Retrieved memories: M2: "Recipe descriptions differ in detail across sources, and regional variants may add ingredients ... a mention of additional alcohol is a reason to inspect variants rather than stop at the standard ingredient list."

Action: Lookup[Zurracapote]

Observation: The page lists red wine, fruit, sugar, and cinnamon. Some recipes add other alcoholic beverages; gin is not named.

Agent conclusion: "The additional alcohol could be gin, consistent with the search lead."

Iteration 4

Retrieved memories: M1: "The familiar drink's base spirit can organize the comparison ... use the focused retrieval results to assess whether the second drink shares that ingredient."

M2: "Recipe summaries may omit regional additions ... consider variant descriptions alongside the standard recipe when comparing ingredients."

Action: Finish[yes]

Agent rationale: "Gibson contains gin, and Zurracapote allows added spirits, so both can contain gin."

Conclusion: Incorrect. A possible variant is substituted for evidence that Zurracapote contains gin.

Outcome: Failure — yes (incorrect). A search cue becomes an unsupported claim about ingredients.

Figure 8: Failure with direct memory injection. Retrieved memories repeatedly promote gin as the organizing search cue. The observations establish that Gibson contains gin but never establish that Zurracapote does, and the agent ultimately converts a possible variant into an affirmative answer.

Case Study: C²Mem Commitment–Competence Calibration

Question: In what city did the "Prince of tenors" star in a film based on an opera by Giacomo Puccini?

Correct answer: Rome.

Starting context: Search returned Antonio Paoli and Franco Corelli. The opened Paoli page says "King of Tenors"; the requested identity is still unverified.

Iteration 1

Retrieved memories: M1: "Earlier retrieval chains progressed by retaining the leading candidate ... refine the missing relation instead of repeatedly reopening identity." [competence 0.20; weight 0.90]

M2: "Similar names or titles can refer to different people ... inspect competing candidates and verify identity before advancing." [competence 0.85; weight 0.05]

Diagnostics: Commitment = 0.90; competence = 0.23 < reference 0.35; risk = 0.09.

Calibration: M1: 0.90 → 0.62; M2: 0.05 → 0.19; insufficient support: 0.05 → 0.19.

Action: Lookup[Franco Corelli]

Conclusion: Confirms the "Prince of tenors" identity and corrects the entity branch.

Iteration 2

Retrieved memories: M1: "Resolve the identifying description against page evidence ... preserve the verified identity as the next hop's starting point." [competence 0.85; weight 0.15]

M2: "Decompose the question into a person–work–location chain ... verify the performer and composer before extracting the city." [competence 0.80; weight 0.80]

Diagnostics: Commitment = 0.82; competence = 0.81 > reference 0.35.

Calibration: Risk = 0.00; retain the retrieved weights.

Action: Search[Franco Corelli Puccini film]

Conclusion: Retrieves Tosca (1956 film) as a candidate; the relation still needs verification.

Iteration 3

Retrieved memories: M1: "A retrieved work is only a candidate until its relations are checked ... verify the performer, composer, and filming location." [competence 0.80; weight 0.90]

M2: "Before accepting a candidate's downstream facts, revisit the identifying evidence ... confirm that the person matches the question." [competence 0.85; weight 0.05]

Diagnostics: Commitment = 0.90; competence = 0.80 > reference 0.35.

Calibration: Risk = 0.00; retain the retrieved weights despite high commitment.

Action: Lookup[Tosca (1956 film)]

Conclusion: Confirms Corelli, the Puccini adaptation, and filming at Cinecittà in Rome.

Iteration 4

Retrieved memories: M1: "After resolving intermediate entities, check the final relation in the source ... ensure that the location belongs to the verified work." [competence 0.80; weight 0.10]

M2: "Additional retrieval is unnecessary once each required link has direct support ... finish with the shortest answer to the requested slot." [competence 0.90; weight 0.85]

Diagnostics: Commitment = 0.81; competence = 0.89 > reference 0.35.

Calibration: Risk = 0.00; retain the retrieved weights.

Action: Finish[Rome]

Conclusion: The verified person–film–city chain supports the correct answer.

Outcome: Success — Rome. Reliable evidence replaces the initially dominant, unreliable guidance.

Figure 9: C²Mem commitment–competence calibration. In the first iteration, calibration suppresses the dominant low-competence memory and reallocates support to reliable identity verification and the insufficient-support node. Subsequent high-competence memories are retained, producing the verified Corelli–*Tosca*–Rome chain.

Failure with direct memory injection. In Figure 8, the retrieved memories consistently frame Gibson's base spirit as a useful cue for investigating Zurracapote. This guidance is useful for forming a query, but it becomes an anchor across the four iterations. The retrieved evidence states that Gibson contains gin and that Zurracapote may admit recipe variants; it never states that a Zurracapote recipe contains gin. Nevertheless, the trajectory repeatedly interprets the absence of disconfirmation as support for the original cue. The final answer therefore substitutes a possible variant for evidence and returns "yes." This case illustrates memory-induced overconfidence: coherent memory guidance stabilizes the execution around a direction whose factual support is insufficient.

Correction through commitment–competence calibration. In the first iteration of Figure 9, the dominant memory receives original weight 0.90 but has competence 0.20. The resulting commitment is 0.90, while aggregated competence is 0.23, below the reference value 0.35, yielding overconfidence risk 0.09. Calibration reduces the dominant memory's weight from 0.90 to 0.62 and raises both the identity-verification memory and insufficient-support mass from 0.05 to 0.19. This rebalancing keeps the identity-verification alternative active, and the next action checks Franco Corelli rather than continuing along the initially opened Antonio Paoli branch. After the identity is corrected, the retrieved memories have competence between 0.80 and 0.90; their aggregated competence exceeds the 0.35 reference, so the overconfidence risk is zero and their weights are retained. The agent then verifies the Corelli–*Tosca*–Cinecittà chain and returns Rome.

Comparison. Together, the cases show that calibration is selective rather than a general penalty on memory use. Direct injection allows a plausible search heuristic to remain influential even after the observations fail to verify its implied answer. C^{2}Mem intervenes when strong guidance is paired with low historical reliability, but leaves equally strong guidance unchanged once its competence is above the reference. Commitment determines how strongly a memory-supported direction can shape execution, while competence determines whether that influence is warranted by historical outcomes.

C LLM Prompt Template

Figures 10–12 show the three prompt templates used for memory construction and retrieval.

Behavior-Tree Node Summarization Prompt Template

Task

Consolidate the member episodes of one behavior-tree node into a reusable description of their shared ordered execution mechanism. Exclude task topics, entities, answers, and correctness from the summary.

Output Format

```
{
  "behavior_pattern":
    "Trigger: ... Early: ... Feedback: ... Transition: ... Stop: ...",
  "reusable_experience": {
    "agent_id": "applicability trigger plus next-action rule"
  }
}
```

Real Input

```
Member behavior patterns:
- {behavior_pattern_1}
- {behavior_pattern_2}
- ...
```

Figure 10: Behavior-tree node summarization prompt template.

Behavior-tree node summarization. The template consolidates the behavior patterns of a node's member episodes into a shared ordered execution mechanism. It removes task-specific topics, entities, answers, and correctness while retaining reusable, agent-specific experience.

Behavioral-Support Evaluation Prompt Template

Task

Estimate how strongly each retrieved behavior-tree node supports the current execution prefix while reserving probability mass for insufficient memory support. Assign a joint probability distribution over the listed ordinary node IDs and node 0.

An ordinary weight is the behavioral support that a node provides for the current execution. Weight on node 0 means that the retrieved memories provide insufficient support.

Judge ordered execution compatibility, not topic or entity overlap. Include every listed ID exactly once. All weights must be nonnegative and sum to 1.

Output Format

```
{
  "weights": {
    "node_id": 0.0,
    "0": 0.0
  },
  "rationale": "short reason"
}
```

Real Input

```
Current execution prefix:
{execution_prefix}

Candidate nodes:
node_id: {node_id}
behavior_pattern: {behavior_pattern}
historical_competence: {historical_competence}

...
```

Figure 11: Behavioral-support evaluation prompt template.

Behavioral-support evaluation. Given the current execution prefix and the retrieved behavioral nodes, the template produces a probability distribution over the ordinary nodes and the insufficient-support node. The distribution measures support from ordered execution compatibility rather than topic or entity overlap.

Episode summarization. Figure 12 shows the template that converts a completed execution into a content summary, an ordered behavior pattern, an outcome summary, and agent-specific reusable experience. It separates task-specific evidence from the transferable execution behavior used to construct behavioral memory.

Coordination across the memory lifecycle. The three templates form a staged interface between trajectory processing, behavioral abstraction, and calibrated retrieval. The memory lifecycle begins with episode summarization after a completed training trajectory. It separates four outputs: a content summary that preserves the task context and decisive evidence, an ordered behavior pattern that records the execution mechanism, an outcome summary that states whether the trajectory succeeded, and agent-specific reusable experience that expresses when and how a strategy should be applied. Keeping these fields separate prevents task-specific entities or answers from defining the identity of a behavioral pattern. The node-summarization template then consolidates the patterns of multiple member episodes into a shared ordered mechanism. It explicitly removes topics, entities, answers, and correctness labels while retaining transferable control structure and agent-specific applicability conditions. The resulting summary becomes the node description used for behavior-centered retrieval across tasks.

At execution time, the behavioral-support evaluator receives the current execution prefix, candidate node identifiers and summaries, and their historical competence metadata. It judges compatibility between the ordered execution prefix and each behavioral mechanism rather than relying on lexical or entity overlap. Its structured output assigns nonnegative mass to every candidate and to the insufficient-support node, providing an abstention channel when memory support is insufficient. This distribution instantiates $\boldsymbol{\alpha}_t$, from which ordinary-memory support m_t and the conditional mem-

Episode Summarization Prompt Template

Task

Convert one completed multi-agent execution into a reusable episode representation. Separate task-specific evidence from transferable execution behavior. The behavior pattern is the primary routing representation and must encode the ordered control mechanism rather than the task topic or answer.

Use actual agent IDs when they are visible. Keep all values entity-free except content_summary. Do not include Markdown.

Output Format

```
{
  "content_summary": "short task and decisive-evidence summary",
  "behavior_pattern":
    "Trigger: ... Early: ... Feedback: ... Transition: ... Stop: ...",
  "outcome_summary": "why this execution succeeded or failed",
  "reusable_experience": {
    "agent_id": "applicability trigger plus next-action rule"
  }
}
```

Real Input

```
Authoritative outcome: {success_or_failure}

Task:
{task}

Trajectory:
{trajectory}

Feedback:
{feedback}
```

Figure 12: Episode summarization prompt template.

ory weights are computed. The evaluator estimates support only; tree-based commitment, posterior competence, uncertainty comparison, and weight calibration remain explicit numerical operations outside the prompt. This separation prevents fluent or confident memory text from being treated as reliability evidence by itself. The fixed identifiers and machine-readable schemas also make the information passed between stages traceable, while the test-time prompt excludes benchmark outcomes and leaves the trained memory state unchanged.

D Theoretical Analysis

This section analyzes the exposure-weighted competence estimator and the optimization view of memory calibration. It also states an additional comparability assumption under which a competence deficit indicates a negative memory effect. Elementary range, monotonicity, and posterior-inheritance properties follow immediately from their definitions and are stated inline in Appendix A.

D.1 Setup for Exposure-Weighted Competence

Fix an ordinary behavioral node v. For training task q, define

$$W_q := \rho_q(v) \in [0,1], \qquad Y_q := y_q \in \{0,1\}.$$

Here, W_q is the trajectory-level exposure of node v and Y_q is the final task outcome. For the fixed node v, let

$$\mathcal{D}_{v,n} := \{(W_q, Y_q)\}_{q=1}^{n}.$$

Let the initial prior be

$$\theta_v \sim \mathrm{Beta}\left(a_v^{(0)}, b_v^{(0)}\right), \qquad a_v^{(0)} > 0, \quad b_v^{(0)} > 0. \tag{D.1}$$

After n training tasks, main-text Eq. (13) yields

$$a_{v,n} = a_v^{(0)} + \sum_{q=1}^{n} W_q Y_q, \tag{D.2}$$

$$b_{v,n} = b_v^{(0)} + \sum_{q=1}^{n} W_q (1 - Y_q). \tag{D.3}$$

The corresponding posterior mean is

$$\widehat{\theta}_{v,n} := \frac{a_{v,n}}{a_{v,n} + b_{v,n}} = \frac{a_v^{(0)} + \sum_{q=1}^{n} W_q Y_q}{a_v^{(0)} + b_v^{(0)} + \sum_{q=1}^{n} W_q}.$$

Define the population-level exposure-weighted historical execution reliability of node v as

$$\theta_v^\star := \frac{\mathbb{E}[W_q Y_q]}{\mathbb{E}[W_q]}, \qquad \mathbb{E}[W_q] > 0. \tag{D.4}$$

Because Y_q is binary, this target can equivalently be written as

$$\theta_v^\star = \frac{\mathbb{E}[W_q \Pr(Y_q = 1 \mid W_q)]}{\mathbb{E}[W_q]}.$$

It measures historical task success associated with node v, weighted by the degree to which the node participates in the corresponding trajectories.

For later use, define

$$S_n := \sum_{q=1}^{n} W_q Y_q, \qquad T_n := \sum_{q=1}^{n} W_q, \qquad c_0 := a_v^{(0)} + b_v^{(0)}, \qquad \mu_v^{(0)} := \frac{a_v^{(0)}}{c_0}. \tag{D.5}$$

D.2 Fractional Beta–Bernoulli Interpretation

For training task q, define the fractional likelihood contribution

$$L_q(\theta_v) := \left[\theta_v^{Y_q}(1 - \theta_v)^{1 - Y_q}\right]^{W_q}. \tag{D.6}$$

Multiplying these fractional likelihoods by the prior in Eq. (D.1) gives

$$\theta_v \mid \mathcal{D}_{v,n} \sim \mathrm{Beta}(a_{v,n}, b_{v,n}),$$

where $a_{v,n}$ and $b_{v,n}$ are given by Eqs. (D.2) and (D.3). For $W_q = 1$, the task contributes one Bernoulli observation; for $W_q = 0$, the posterior remains unchanged. Values in $(0, 1)$ contribute fractional evidence. Main-text Eq. (13) therefore corresponds to a generalized Bayesian update under a power likelihood, with W_q controlling the amount of task-level evidence assigned to node v.

D.3 Strong Consistency of the Competence Estimator

For an asymptotic characterization under an i.i.d. task–exposure process, consider a stationary regime in which the node-exposure and outcome pairs are independent and share a common distribution.

Theorem D.1 (Strong consistency)**.** *Assume that $\{(W_q, Y_q)\}_{q \geq 1}$ is independent and identically distributed, with*

$$0 \leq W_q \leq 1, \qquad Y_q \in \{0, 1\}, \qquad \mathbb{E}[W_q] > 0.$$

Then

$$\widehat{\theta}_{v,n} \xrightarrow{\text{a.s.}} \theta_v^\star = \frac{\mathbb{E}[W_q Y_q]}{\mathbb{E}[W_q]}. \tag{D.7}$$

Therefore, the posterior mean induced by main-text Eq. (13) *is a strongly consistent estimator of the exposure-weighted historical execution reliability of node v.*

Proof. Since $0 \le W_q Y_q \le 1$ and $0 \le W_q \le 1$, both random variables have finite first moments. By the strong law of large numbers,

$$\frac{S_n}{n} \xrightarrow{\text{a.s.}} \mathbb{E}[W_q Y_q], \qquad \frac{T_n}{n} \xrightarrow{\text{a.s.}} \mathbb{E}[W_q].$$

Because $\mathbb{E}[W_q] > 0$, $T_n \to \infty$ almost surely and

$$\frac{S_n}{T_n} = \frac{S_n/n}{T_n/n} \xrightarrow{\text{a.s.}} \frac{\mathbb{E}[W_q Y_q]}{\mathbb{E}[W_q]} = \theta_v^\star. \tag{D.8}$$

Directly from the posterior mean,

$$\widehat{\theta}_{v,n} = \frac{c_0}{c_0 + T_n}\mu_v^{(0)} + \frac{T_n}{c_0 + T_n}\frac{S_n}{T_n}.$$

As $T_n \to \infty$ almost surely, the first coefficient converges to zero and the second converges to one. Combining these limits with Eq. (D.8) proves the result. □

Posterior concentration. The same effective sample size T_n also controls posterior uncertainty. Since $a_{v,n} + b_{v,n} = c_0 + T_n$ and $xy \le (x+y)^2/4$ for nonnegative x, y,

$$\mathrm{Var}(\theta_v \mid \mathcal{D}_{v,n}) \le \frac{1}{4(c_0 + T_n + 1)} \xrightarrow{\text{a.s.}} 0.$$

Together with Theorem D.1, this implies that the generalized posterior concentrates at $\theta_v^\star$. This result applies to a fixed node under the stated stationary i.i.d. regime; it does not by itself characterize adaptive exposure, node splitting, or distribution shift.

D.4 Conditional Interpretation of Competence Deficits

Historical execution reliability and the effect of injecting a memory are distinct quantities. Behavioral retrieval motivates their connection by selecting experiences with execution patterns similar to the current prefix. The following assumption makes the additional comparability conditions explicit.

Fix the current memory state, an ordinary node v, and a pre-injection behavioral state z. We suppress the execution-step index in the competence estimates. Let $Y_v(z)$ and $Y_\varnothing(z)$ denote the binary outcomes of two continuations that respectively inject the memory associated with v or omit that memory. Both continuations start from the same state and use the same otherwise-specified MAS protocol and background memory configuration; subsequent actions may differ as a consequence of the intervention. Probabilities average over execution randomness and task instances represented by z. Define

$$p_v(z) = \Pr(Y_v(z) = 1), \qquad p_\varnothing(z) = \Pr(Y_\varnothing(z) = 1), \qquad \Delta_v(z) = p_v(z) - p_\varnothing(z). \tag{D.9}$$

Thus, $\Delta_v(z) < 0$ means that injecting this memory lowers the conditional success probability relative to omitting it.

Assumption D.2 (Behavior-conditioned reference comparability)**.** *For the node and state under consideration, the historical competence estimates satisfy*

$$|\mu_v - p_v(z)| \le \epsilon_v(z), \qquad |\mu_0 - p_\varnothing(z)| \le \epsilon_0(z), \tag{D.10}$$

where $\epsilon_v(z), \epsilon_0(z) \ge 0$ bound the combined effects of estimation error, residual task and behavior mismatch, and differences between historical exposure and the specified interventions.

Proposition D.3 (Competence deficit and negative memory effect)**.** *Under Assumption D.2, let $E_v(z) = \epsilon_v(z) + \epsilon_0(z)$. Then*

$$\mu_v - \mu_0 - E_v(z) \le \Delta_v(z) \le \mu_v - \mu_0 + E_v(z). \tag{D.11}$$

In particular, $\mu_0 - \mu_v > E_v(z)$ implies $\Delta_v(z) < 0$.

Proof. Decompose the conditional effect as

$$\Delta_v(z) = (\mu_v - \mu_0) + (p_v(z) - \mu_v) - (p_\varnothing(z) - \mu_0).$$

Applying the two error bounds in Assumption D.2 gives Eq. (D.11) and the stated implication. □

Scope of the assumption. Proposition D.3 provides a conditional interpretation for a single-memory intervention under Assumption D.2. Applying this interpretation requires μ_0 to approximate the matched no-memory success probability and the competence gap to exceed the combined approximation errors. The unknown node is estimated from shared task outcomes, so this reference comparability requires an additional assumption beyond behavioral similarity, the exposure-weighted update, and Theorem D.1. Our experiments do not estimate these error bounds; c_t^- captures posterior uncertainty within the competence model, with reference mismatch outside its scope. Joint injection also introduces possible interactions among memories beyond the single-memory proposition. We therefore use the aggregated competence deficit as a calibration risk signal, without claiming empirical identification of causal memory effects.

D.5 Optimization View of Memory Calibration

At execution step t, the original support distribution $\boldsymbol{\alpha}_t$, overconfidence score O_t, and node-level deficit contributions $\delta_t(v)$ are computed before calibration and held fixed during the optimization. Define the supported node set as

$$\mathcal{S}_t = \left\{ v \in \widehat{\mathcal{R}}_t : \alpha_t(v) > 0 \right\}.$$

Because $\boldsymbol{\alpha}_t$ is a probability distribution, $\mathcal{S}_t$ is nonempty. We restrict the calibrated distribution to this support:

$$\Delta_t = \left\{ p : p(v) \geq 0, \quad \sum_{v \in \widehat{\mathcal{R}}_t} p(v) = 1, \quad p(v) = 0 \text{ for } v \notin \mathcal{S}_t \right\}.$$

This restriction also follows from the standard convention that $D_{\mathrm{KL}}(p \| \boldsymbol{\alpha}_t) = +\infty$ whenever p assigns positive mass to a node with zero original support.

Proposition D.4 (Optimal calibrated distribution)**.** *Let $\kappa \geq 0$, $O_t \geq 0$, and $\delta_t(v) \geq 0$ be finite. The optimization problem*

$$\min_{p \in \Delta_t} \left\{ D_{\mathrm{KL}}(p \| \boldsymbol{\alpha}_t) + \kappa O_t \sum_{v \in \widehat{\mathcal{R}}_t} p(v) \delta_t(v) \right\} \tag{D.12}$$

has a unique solution:

$$p^\star(v) = \frac{\alpha_t(v) \exp[-\kappa O_t \delta_t(v)]}{Z_t}, \qquad Z_t = \sum_{u \in \widehat{\mathcal{R}}_t} \alpha_t(u) \exp[-\kappa O_t \delta_t(u)]. \tag{D.13}$$

This solution is the calibrated distribution in main-text Eq. (12)*.*

Proof. On $\mathcal{S}_t$, the objective can be written as

$$J_t(p) = \sum_{v \in \mathcal{S}_t} p(v) \log \frac{p(v)}{\alpha_t(v)} + \kappa O_t \sum_{v \in \mathcal{S}_t} p(v) \delta_t(v).$$

Introducing a multiplier ν for the normalization constraint gives the Lagrangian

$$\mathcal{L}(p, \nu) = J_t(p) + \nu \left(\sum_{v \in \mathcal{S}_t} p(v) - 1 \right).$$

For an interior stationary point,

$$\frac{\partial \mathcal{L}}{\partial p(v)} = \log \frac{p(v)}{\alpha_t(v)} + 1 + \kappa O_t \delta_t(v) + \nu = 0.$$

Therefore,

$$p(v) = \alpha_t(v) \exp[-\kappa O_t \delta_t(v)] \exp[-1 - \nu].$$

Normalization yields $\exp[-1 - \nu] = Z_t^{-1}$, giving Eq. (D.13). Since all penalties are finite, $Z_t > 0$, and this candidate is positive on $\mathcal{S}_t$.

To establish global optimality and uniqueness, substitute $p^\star$ into the objective. For every $p \in \Delta_t$,

$$\begin{aligned} J_t(p) &= \sum_{v \in \mathcal{S}_t} p(v) \log \frac{p(v)}{\alpha_t(v) \exp[-\kappa O_t \delta_t(v)]} \\ &= D_{\mathrm{KL}}(p \| p^\star) - \log Z_t. \end{aligned}$$

The KL divergence is nonnegative and equals zero only when $p = p^\star$. Hence $p^\star$ is the unique minimizer. □

The solution preserves the original distribution when $O_t = 0$. For any two supported nodes u and v,

$$\frac{\widetilde{\alpha}_t(v)}{\widetilde{\alpha}_t(u)} = \frac{\alpha_t(v)}{\alpha_t(u)} \exp\left[-\kappa O_t\big(\delta_t(v) - \delta_t(u)\big)\right].$$

Thus, for $\kappa > 0$, increasing O_t decreases the relative weight of a node with a larger deficit contribution. Calibration preserves the relative weights of nodes with equal deficit contributions. In particular, the unknown node and ordinary nodes with zero deficit receive the same zero penalty.

E Commitment–Competence Interaction Analysis

We further characterize how commitment interacts with competence on HotpotQA. The association between commitment and success changes with the underlying competence level. Table 3 and Figures 13–15 examine the temporal separation of competence, its outcome-conditioned distribution, and the competence-dependent association between commitment and success.

E.1 Competence Dynamics

The outcome-conditioned competence trajectories end with a substantial gap, but the evolution of this difference is equally informative. We divide normalized execution progress into three equal intervals and summarize the mean competence of successful and failed cases in each stage.

Table 3: Stage-wise mean competence of successful and failed cases on HotpotQA. The competence gap is computed between the two outcome groups.

Progress interval	Success mean competence	Failure mean competence	Competence gap
Early: $[0, 1/3]$	0.81	0.76	0.05
Middle: $[1/3, 2/3]$	0.81	0.69	0.12
Late: $[2/3, 1]$	0.85	0.64	0.21

The competence gap grows from 0.05 in the early interval to 0.12 in the middle and 0.21 in the late interval. The successful-task competence curve first remains stable and then rises, whereas the failed-task curve declines. The resulting progressive divergence shows that the outcome gap is formed throughout execution rather than appearing only at the final step. Late-stage competence is therefore substantially more discriminative of successful and failed executions than early-stage competence.

E.2 Competence Discriminability

Figure 13 complements the trajectory summary with the share of successful and failed cases that reach or exceed each competence threshold.

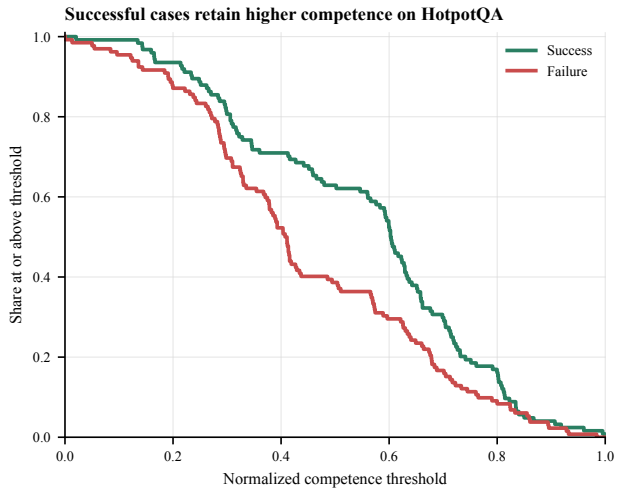


Figure 13: Complementary cumulative distributions of normalized competence on HotpotQA. At each threshold, a higher curve indicates that a larger fraction of cases retain competence at or above that level.

The success curve remains above the failure curve across most of the mid-to-high range. Its median is 0.603, compared with 0.409 for failed cases, and successful cases place substantially more mass above 0.6. Together with Table 3, this shows that competence provides a stable signal for distinguishing successful from failed executions.

E.3 Commitment–Competence Interaction

Figure 14 retains the joint two-dimensional structure and reports the success percentage within each equal-width commitment–competence cell.

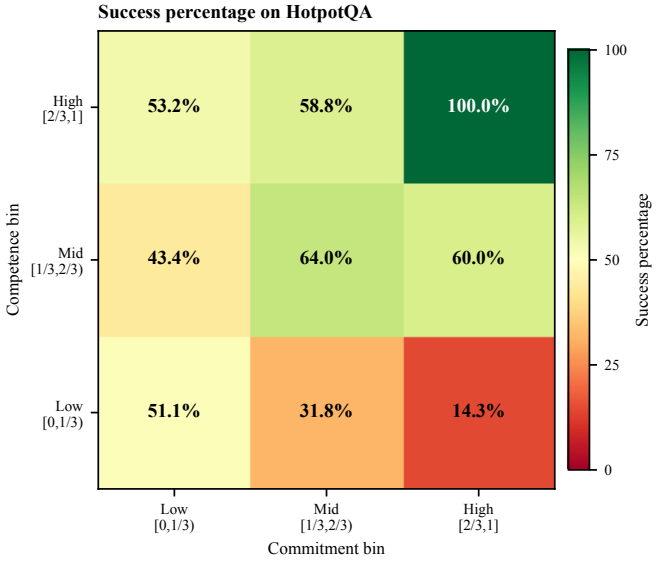


Figure 14: Success percentage across the joint commitment–competence space on HotpotQA. Both axes use three equal-width bins.

Within the low-competence row, the success percentage decreases from 51.1% in the low-commitment bin to 31.8% in the middle bin and 14.3% in the high bin. The same monotone decrease does not appear in the middle- or high-competence rows. In particular, the high-competence/high-commitment cell reaches the highest observed success percentage of 100.0%. The association between commitment and success is therefore conditional: stronger commitment is associated with lower success when competence is low, while this negative association disappears as competence increases. This variation across competence levels identifies commitment as an interaction variable whose meaning depends on competence.

This pattern shows why commitment supplies interaction information beyond a competence-only score. Low competence identifies unreliable memory behavior, whereas commitment indicates how strongly the policy is organized around those memories. Commitment modulates how a competence shortfall is translated into risk: stronger commitment amplifies the risk signal carried by low competence, while under high competence it is associated with higher predicted success.

Figure 15 provides a continuous view of this interaction by varying competence while holding commitment at two observed reference percentiles.

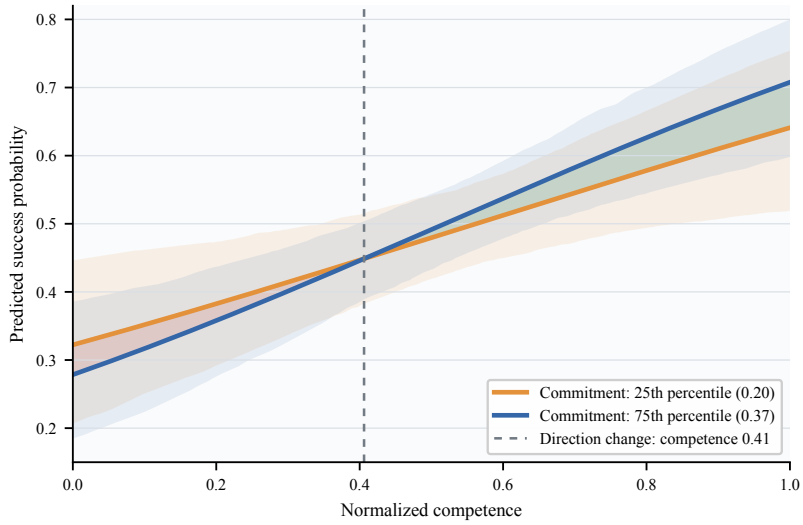


Figure 15: Continuous logistic-interaction estimates on HotpotQA. The orange and blue curves fix commitment at its observed 25th (0.20) and 75th (0.37) percentiles, respectively, while the pale ribbons show 90% bootstrap intervals. The dashed line marks the fitted direction change at competence 0.41. To its left, higher commitment is associated with lower predicted success; to its right, higher commitment is associated with higher predicted success. The red and green regions show the corresponding gaps between the two reference curves.

F Baselines, Datasets, and Multi-Agent Frameworks

F.1 Datasets

The five benchmarks cover factual question answering, yes/no reasoning, multi-hop reasoning, claim verification, and mathematical problem solving.

Natural Questions (NQ). Natural Questions (Kwiatkowski et al., 2019) contains real queries submitted to the Google search engine. The original dataset pairs each question with a Wikipedia page and provides long-answer and short-answer annotations when an answer is present. In our experiments, NQ serves as an open-domain question-answering benchmark, testing the ability to identify the facts needed to answer naturally occurring questions.

BoolQ (BQ). BoolQ (Clark et al., 2019) is a reading-comprehension dataset of naturally occurring yes/no questions. Each example includes a question, a passage, and a binary answer. Many questions require interpreting whether the passage supports a proposition rather than locating a matching answer span. BQ complements open-ended question answering with binary decisions based on textual evidence.

HotpotQA. HotpotQA (Yang et al., 2018) contains Wikipedia-based questions that require combining information from multiple supporting documents. It includes questions that connect facts across documents and questions that compare entities. Sentence-level supporting-fact annotations identify the evidence needed for an answer. The benchmark tests the ability to gather and combine evidence across multiple reasoning steps.

FEVER. FEVER (Thorne et al., 2018) evaluates factual claims against Wikipedia evidence. Its original label set consists of SUPPORTED, REFUTED, and NOTENOUGHINFO. For supported and refuted claims, the dataset includes evidence sentences supporting the judgment. It tests whether a system can distinguish evidence that supports a claim, evidence that contradicts it, and insufficient evidence.

MATH. MATH (Hendrycks et al., 2021) consists of competition mathematics problems with worked solutions and final answers. Solving these problems requires choosing suitable mathematical techniques and carrying out multi-step derivations. It extends the evaluation from knowledge and evidence-based tasks to mathematical reasoning, where the usefulness of a recalled solution strategy depends on the structure of the current problem.

F.2 BASELINES

We compare C^2Mem with five memory methods and a No-memory setting. The following descriptions summarize the memory mechanisms introduced in the original studies.

No-memory. This setting runs the multi-agent system without persistent memory from previous tasks. Agents retain the context available within the current task. It provides a reference for assessing the effect of reusing experience across tasks.

Voyager. Voyager (Wang et al., 2024) is an agent originally developed for open-ended exploration in Minecraft. It builds a growing library of reusable skills represented as executable programs. Skills are indexed by embeddings of their descriptions and retrieved for subsequent tasks. Environment feedback and execution errors guide iterative program refinement. Its memory mechanism centers on accumulating and reusing procedural knowledge.

MemoryBank. MemoryBank (Zhong et al., 2024) maintains long-term memory from past interactions and retrieves relevant information to support later responses. It also summarizes interaction history to capture information about the user. Its memory updating mechanism draws on the Ebbinghaus forgetting curve, allowing memories to fade or be reinforced over time. The method emphasizes selective retention and retrieval during sustained interaction.

Generative. Generative denotes the memory architecture of Generative Agents (Park et al., 2023). It records observations in a natural-language memory stream and synthesizes them into higher-level reflections. Retrieval combines relevance to the current situation, recency, and importance. Retrieved observations and reflections then inform planning and subsequent behavior.

AMA. Adaptive Memory via Multi-Agent Collaboration (AMA) (Huang et al., 2026) coordinates specialized agents to construct, retrieve, verify, and refresh memory. The Constructor and Retriever organize memory at multiple granularities and route queries to relevant content. The Judge checks relevance and consistency, while the Refresher updates conflicting information or removes outdated entries. This design supports adaptive retrieval and maintenance of memory consistency.

G-Memory. G-Memory (Zhang et al., 2025a) organizes multi-agent interaction history as a three-tier hierarchy of insight, query, and interaction graphs. Its bidirectional traversal retrieves both generalizable cross-trial insights and condensed collaboration trajectories. After execution, the hierarchy evolves by assimilating new trajectories, supporting agent-specific memory and continual team evolution.

F.3 Multi-Agent Frameworks

We evaluate memory methods within AutoGen, LLM-Debate, and MacNet. These frameworks provide different ways to organize communication and combine agents' reasoning during task execution.

AutoGen. AutoGen (Wu et al., 2024) provides a framework for building applications through conversations among customizable agents. Agents can combine LLM reasoning, tool execution, and human input, while their roles and interaction rules determine how a task progresses. The framework supports programmable conversation patterns in which agents exchange messages and respond to intermediate results. It provides a flexible setting for studying memory within conversational workflows.

LLM-Debate. LLM-Debate (Du et al., 2024) uses multiple language-model instances to propose answers and exchange reasoning over successive rounds. Agents first produce individual responses, then consider other agents' answers and arguments when revising their own. The process aims to reach a common final answer through repeated critique and refinement. This framework emphasizes comparison and revision of competing reasoning paths.

MacNet. MacNet (Qian et al., 2025) organizes multi-agent collaboration as a directed acyclic graph. The graph specifies communication dependencies, and interactions proceed in topological order so that intermediate solutions can inform subsequent reasoning. Its design supports different network topologies and agent counts. MacNet therefore provides a setting for examining memory within collaboration structured by a communication graph.