CogMem: A Cognitive Memory Architecture for Sustained Multi-Turn Reasoning in Large Language Models

Yiran Zhang[1], Jincheng Hu[2], Mark Dras[1], Usman Naseem[1]
Macquarie University[1], Independent Researcher[2]
{yiran.zhang, mark.dras, usman.naseem}@mq.edu.au

Abstract

Large language models (LLMs) excel at single-turn reasoning but often lose accuracy and coherence over extended, multi-turn interactions. Recent evaluations such as TurnBench–MS highlight recurring failure modes, including reasoning bias, task drift, hallucination, overconfidence, and memory decay. Current approaches typically append full conversational histories, causing unbounded context growth, higher computational costs, and degraded reasoning efficiency. We introduce CogMem, a cognitively inspired, memory-augmented LLM architecture that supports sustained iterative reasoning through structured, persistent memory. CogMem incorporates three layers: a Long-Term Memory (LTM) that consolidates cross-session reasoning strategies; a Direct Access (DA) memory that maintains session-level notes and retrieves relevant long-term memories; and a Focus of Attention (FoA) mechanism that dynamically reconstructs concise, task-relevant context at each turn. Experiments on TurnBench–MS show that this layered design mitigates reasoning failures, controls context growth, and improves consistency across extended reasoning chains, suggesting that structured hierarchical memory is a promising direction for improving long-horizon multi-turn reasoning in large language models.

1 Introduction

Reasoning is central to human intelligence (Wason and Johnson-Laird, 1972; Holyoak and Morrison, 2012) and serves as a core benchmark for artificial systems. Large language models (LLMs) have demonstrated remarkable performance on single-turn reasoning tasks (Brown et al., 2020; Touvron et al., 2023; Comanici et al., 2025). Yet real-world applications demand iterative, multi-turn reasoning: models must integrate new information, revise assumptions, and maintain coherent thinking over extended interactions. Sustaining long-horizon reasoning remains a key challenge.

Recent benchmarks have highlighted this gap. TurnBench-MS (Zhang et al., 2025) requires models to infer hidden rules through sequential interactions, integrating feedback across turns. Even top-performing LLMs achieve only 84% accuracy in classic mode and 18% in nightmare mode, with early errors often cascading through subsequent steps. Other benchmarks, including AvalonBench (Light et al., 2023) and MT-Bench (Bai et al., 2024), confirm that sustained dialog-based reasoning remains a significant challenge. Chain-of-Thought prompting (Wei et al., 2022) improves single-turn inference but is insufficient for long-term, multi-turn reasoning, highlighting the need for persistent, structured memory.

Memory-augmented LLMs offer one path forward. Retrieval-Augmented Generation (Lewis et al., 2020) extends context via retrieval but lacks persistent lifecycle management. Later systems, including MemBank (Zhong et al., 2024), Mem0 (Chhikara et al., 2025), Mem1 (Zhou et al., 2025), A-Mem (Xu et al., 2025), and MemOS (Li et al., 2025), introduced structured, evolving memory architectures that enable dynamic extraction, updating, and long-term storage of relevant knowledge. However, these approaches typically treat memory and reasoning as loosely coupled.

We propose a cognitively inspired, memory-augmented reasoning architecture that tightly integrates reasoning and structured memory. The system maintains a hierarchical memory: a long-term memory stores distilled reasoning strategies and cross-session knowledge, a direct-access memory preserves session-relevant plans and information, and a focus-of-attention module reconstructs minimal working context at each turn. Reasoning and memory agents collaborate continuously, allowing models to recover from early errors, maintain coherence, and efficiently manage context across sessions. This framework combines insights from prior memory-augmented LLMs with a unified,

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 552–562
November 6-10, 2026 ©2026 Association for Computational Linguistics

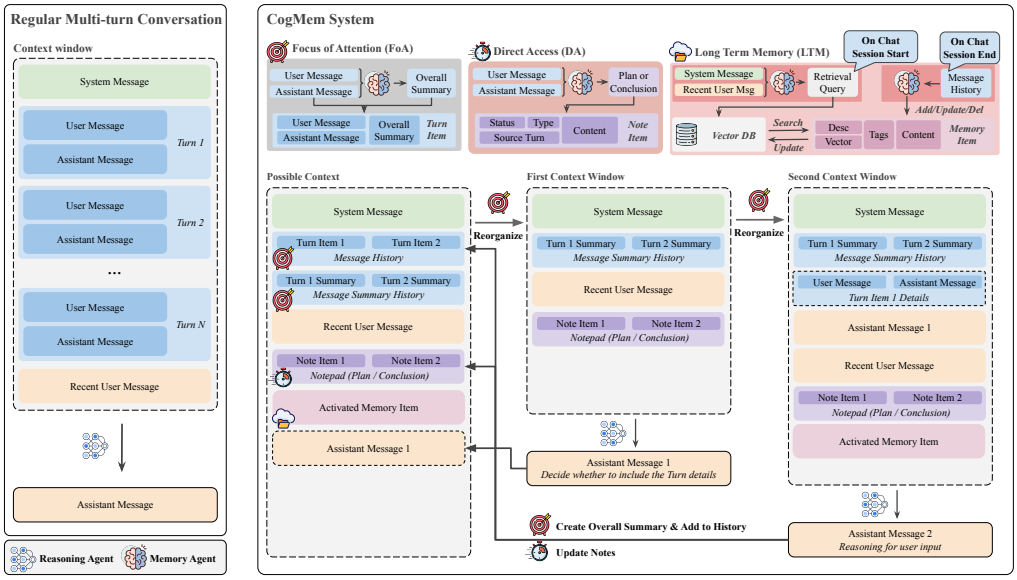


Figure 1: Overview of the CogMem framework.

cognitively grounded approach to persistent, adaptive reasoning. Our contributions are:

- A structured, multi-level memory architecture that enables coherent, long-horizon reasoning while controlling context growth.
- Integration of modular reasoning and memory agents for continuous inference, knowledge accumulation, and recovery from reasoning errors.
- Empirical validation on TurnBench-MS demonstrating improved reasoning accuracy, reduced failure modes, and enhanced efficiency in extended multi-turn reasoning tasks.

2 Method

2.1 System Overview

We propose a memory-augmented reasoning framework designed to enable LLMs to maintain coherent, adaptive, and efficient reasoning across multiple turns. The system comprises two cooperative agents: **the reasoning agent** and **the memory agent**, coordinated by a **session manager** and a **memory manager** (Section 2.4), with definitions and case studies provided in Appendix B. These components operate on a hierarchical memory structure inspired by cognitive models of working memory, consisting of three layers with distinct levels of accessibility and persistence: long-term memory (LTM), direct-access memory (DA), and focus of attention (FoA) (Figure 1).

Long-Term Memory (LTM) stores distilled reasoning strategies and reusable problem-solving patterns accumulated across sessions, serving as a persistent repository that evolves as new insights are integrated.

Direct-Access Memory (DA) functions as session-level working memory, maintaining concise notes of intermediate conclusions, sub-goals, and ongoing plans. At the start of each session, the LTM supplies the DA with the most relevant long-term memories, enabling the reasoning agent to access prior knowledge without repeatedly querying the full LTM.

Focus-of-Attention (FoA) dynamically reconstructs the minimal reasoning context for each turn. It selectively integrates current notes, retrieved long-term memories, summarized dialogue history, and new user input to produce a compact, informative prompt. This ensures a bounded, interpretable context while retaining essential information for accurate reasoning.

This hierarchical design allows iterative reasoning without unbounded context expansion, preserving critical reasoning cues, recovering from earlier mistakes, and maintaining continuity across extended reasoning chains while controlling computational cost.

2.2 Processing Workflow

Upon receiving user input, the session manager determines whether an existing session can be reused or extended. If a matching session exists, stored results and memory states are reused directly. Otherwise, the system searches for an *inheritable session*, where a prior dialogue forms the prefix of the current input. In these cases, corresponding

memories and notes are restored to continue reasoning seamlessly. If no session is reusable, a new session is initialized: the DA is created empty, and the memory agent queries the LTM for relevant information by rewriting the current situation into a descriptive query for semantic retrieval.

Reasoning begins with constructing the first FoA, the *First Context Window* (Figure 1), which includes session notes, summarized dialogue history, and new user input. The reasoning agent evaluates whether this context is sufficient; if incomplete, it returns identifiers of missing turns, allowing FoA to retrieve details and construct a *Second Context Window*. The agent then performs inference over this refined context, integrating retrieved turn details, user input, and existing notes. Together, these cycles constitute a single turn in the dialogue flow.

After reasoning, the memory agent summarizes the generated content. The summary, user input, and model response form a new turn record, updating the DA asynchronously. This ensures the user receives immediate responses while the memory remains coherent and current. Turn records and updated memory states are stored in the session cache for efficient reuse.

Inactive sessions trigger a cleanup procedure: expired sessions and cached DA entries are deleted, while FoA turn data are reference-counted to enable turn-level reuse. Before final removal, the LTM reviews the session to determine whether distilled knowledge should be merged, added, or excluded, ensuring only meaningful reasoning traces persist.

2.3 Interaction among Memory Layers

The memory layers interact through ongoing cycles of retrieval, reconstruction, and refinement. FoA selects activated memories and relevant DA notes to construct the next reasoning context. The DA retrieves relevant entries from the LTM and updates its notes as new reasoning outcomes are summarized. At session conclusion, the LTM distills essential insights from DA and FoA summaries, integrating them into its persistent store. This bidirectional flow allows the model to accumulate experience and maintain stable reasoning across sessions.

2.4 Implementation and Efficiency

The framework is modular and compatible with the OpenAI SDK, operating with any base LLM. The memory agent is a lightweight model responsible for summarization, retrieval, and note generation, enabling a separation of complex inference and efficient memory management.

Short-term data, including notes, turns, and session metadata, are stored in RAM for fast access, optionally supported by external caches such as Redis. The LTM resides in a vectorized database for high-speed semantic retrieval and selective updates. The memory manager orchestrates interactions among FoA, DA, and LTM, while the session manager handles identification, caching, expiration, and garbage collection. Event-triggered garbage collection ensures consistent latency regardless of session length, maintaining bounded computational cost across long dialogues.

2.5 Design Rationale and Error Mitigation

The framework addresses recurring multi-turn reasoning failures, including reasoning bias, task misconception, hallucination, overconfidence, memory decay, and unbounded context growth.

FoA and DA reduce reasoning bias, hallucination, and memory decay by reconstructing a concise input window at each step, retaining only essential reasoning traces. DA maintains structured notes of current plans and intermediate conclusions rather than full histories, ensuring reasoning is grounded in distilled information. Iteratively revising assumptions in notes reduces bias over time.

LTM accumulates refined reasoning patterns and task-specific strategies across sessions, guiding task understanding and planning, and enhancing cross-session adaptability. Persistent memory supports incremental improvement and long-term stability.

The session manager and lifecycle controller bound computational cost. Reusable or inheritable sessions avoid redundant computation, and selective refinement into LTM ensures only meaningful reasoning traces persist. Overall, the framework transforms multi-turn reasoning from an ever-growing sequence of appended contexts into structured, iterative refinement, yielding more reliable, interpretable, and temporally consistent reasoning.

3 Experiment Setup and Results

3.1 Experimental Settings

We evaluate our framework on the TurnBench-MS benchmark (Zhang et al., 2025), which is designed to assess iterative reasoning and adaptation in multi-turn interactions. TurnBench–MS simulates rule-discovery games where models infer hidden patterns through sequential queries and feedback. Fol-

Models	Average Accuracy			
	Total	Easy	Medium	Hard
Baseline - gemini-2.5-flash	0.76	0.87	0.93	0.47
Baseline + FoA	0.78	0.93	0.87	0.53
Baseline + FoA + DA	0.84	0.93	0.93	0.67
Baseline + FoA + DA + LTM	**0.93**	**1.00**	**1.00**	**0.80**
Random Guess	0.0085	0.0079	0.0098	0.0077

Table 1: Performance comparison between the CogMem framework and the baseline model under the TurnBench–MS classic mode settings. A random-guessing baseline is included for comparison. All results are obtained using the chain-of-thought (CoT) prompting strategy. The **bold** text represents the best performance.

lowing the official evaluation protocol, we use the classic mode consisting of 45 game setups across three difficulty levels, including easy, medium, and hard.

All experiments employ chain-of-thought prompting. The reasoning agent is Gemini 2.5 Flash (Comanici et al., 2025), while the memory agent uses a lightweight Gemini 2.5 Flash Lite variant for summarization and retrieval. External vector storage is managed via Milvus (Wang et al., 2021), and accuracy is computed following TurnBench–MS standards (Pedregosa et al., 2011). Full experimental and implementation details are provided in Appendix C and Appendix D.

3.2 Baselines

We evaluate four incremental configurations to isolate the contribution of each memory component:

- **Baseline:** Gemini 2.5 Flash with chain-of-thought prompting, no memory.
- **Baseline + FoA:** Adds focus-of-attention for bounded context reconstruction.
- **Baseline + FoA + DA:** Incorporates direct-access memory for structured session notes.
- **Baseline + FoA + DA + LTM:** Full system with long-term memory for cross-session adaptation.

All other settings (prompts, decoding parameters, temperature) are identical. A random-guessing baseline is included to indicate lower-bound performance. Detailed experiment setups are in Appendix C.

3.3 Results and Analysis

Table 1 summarizes the performance for the classic mode of TurnBench-MS (see Appendix D.1 for additional experiments). The baseline uses the Gemini 2.5 Flash results, achieving an overall accuracy of 0.76. Introducing the FoA alone brings modest improvements on the easy and hard setups, while performance declines on the medium setups. This suggests that dynamic context reconstruction helps maintain focus, though its impact remains limited without the notes provided by the DA.

Adding the direct-access memory yields a substantial gain, raising total accuracy in the classic mode to 0.84. This indicates that structured note keeping and short-term summarization improve reasoning consistency and help the model recover from earlier mistakes by preserving key conclusions between turns. Finally, integrating the long-term memory produces the strongest results: 0.93 total accuracy in the classic mode, with perfect scores on easy and medium difficulties and 0.80 on hard setups. The improvement confirms that cross-session accumulation of distilled reasoning patterns allows the model to generalize strategies and stabilize its decision process over extended reasoning chains. This effect is further illustrated through a qualitative case study in Appendix E.

Overall, the ablation trend demonstrates clear additive benefits from each memory layer. FoA mitigates token growth and focus drift; DA enhances within-session consistency; and LTM extends adaptation across sessions. Together, they transform the model's behavior from reactive, context-bound reasoning to structured, self-referential reasoning capable of maintaining coherence under both standard and adversarial conditions. A detailed analysis of latency and total computational cost for each memory component is provided in Appendix F.2.

4 Conclusion

We introduced CogMem, a cognitively inspired memory architecture that enables large language models to sustain accurate, coherent, and efficient reasoning over extended multi-turn interactions. CogMem integrates three complementary layers: long-term memory, direct-access memory, and focus of attention to manage reasoning information at different temporal scales. This design allows the model to retain essential context, prevent redundant expansion, and recover earlier errors. Experiments on TurnBench-MS show that each memory layer contributes incrementally, with the full CogMem system achieving substantial gains over the baseline Gemini 2.5 Flash model. These results demonstrate that structured, cognitively grounded memory can transform LLMs from reactive, context-dependent reasoners into adaptive, self-consistent systems capable of sustained multi-turn inference.

Limitations

While CogMem demonstrates clear improvements in reasoning stability and coherence, its evaluation remains limited in scope. All experiments are conducted on the TurnBench-MS benchmark with a small number of base models (Gemini 2.5 Flash and Qwen3-14B), and component-wise ablations are performed only in the classic mode, where each difficulty level contains 15 setups. These factors constrain the generalizability of our findings. Future work will extend the evaluation to a wider range of reasoning benchmarks, task types, and language models, and will include repeated runs to further validate CogMem's robustness across diverse multi-turn reasoning scenarios.

Ethics Statement

This work focuses on improving the reasoning stability and efficiency of large language models through structured memory design. All experiments were conducted using publicly available benchmark data (TurnBench-MS) and proprietary language models accessed through standard APIs, and open-weight models (such as Qwen3). No personally identifiable or sensitive information was used in any stage of development or evaluation. The goal of CogMem is to enhance the reliability and interpretability of multi-turn reasoning systems, which may reduce unintended hallucinations and overconfident outputs in deployed AI applications. Nonetheless, improved long-term reasoning capabilities also raise considerations regarding information persistence and user privacy. Future work will further examine safeguards for memory management, data retention, and user consent to ensure that cognitively inspired memory systems remain transparent, controllable, and aligned with ethical standards for responsible AI development.

References

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and 1 others. 2024. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. *arXiv preprint arXiv:2402.14762*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

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

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Keith J. Holyoak and Robert G. Morrison. 2012. *The Oxford Handbook of Thinking and Reasoning*. Oxford University Press.

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

Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, and 1 others. 2025. Memos: An operating system for memory-augmented generation (mag) in large language models. *arXiv preprint arXiv:2505.22101*.

Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. 2023. Avalonbench: Evaluating llms playing the game of avalon. *arXiv preprint arXiv:2310.05036*.

Klaus Oberauer. 2002. Access to information in working memory: Exploring the focus of attention. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 28(3):411–421.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and 1 others. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, and 1 others. 2021.

Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 international conference on management of data*, pages 2614–2627.

Peter Cathcart Wason and Philip Nicholas Johnson-Laird. 1972. *Psychology of Reasoning: Structure and Content*. Harvard University Press, Cambridge, MA, USA.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

Yiran Zhang, Mo Wang, Xiaoyang Li, Kaixuan Ren, Chencheng Zhu, and Usman Naseem. 2025. TurnBench-MS: A benchmark for evaluating multi-turn, multi-step reasoning in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 19892–19924, Suzhou, China. Association for Computational Linguistics.

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

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. 2025. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*.

A Related Work

Multi-Turn Reasoning and Benchmarks As LLMs are applied to increasingly complex tasks, benchmarks have evolved to test multi-turn and multi-step reasoning beyond single-query question answering. TurnBench-MS (Zhang et al., 2025) is a recent benchmark explicitly designed for iterative reasoning: models must infer hidden rules through sequential interactions and integrate feedback across turns. Results reveal a large performance gap between humans and current LLMs, especially when long reasoning chains or corrections are required. Analyses further show that reasoning errors made early in the process often persist or cascade through later steps. Other benchmarks such as AvalonBench (Light et al., 2023) and MT-Bench (Bai et al., 2024) confirm that sustained dialog-based reasoning remains a significant challenge for LLMs.

To mitigate such failures, Chain-of-Thought (CoT) prompting (Wei et al., 2022) encourages models to articulate intermediate steps, improving single-instance reasoning but remaining insufficient for multi-turn or session-level reasoning, where memory continuity and selective recall are essential. Our work builds upon these insights, using TurnBench–MS as a testbed while introducing an explicit structured memory framework to improve reasoning persistence and efficiency.

Memory-Augmented LLMs and Long-Term Interaction Recent research increasingly views memory as a core component of scalable LLM systems. Early approaches such as Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) expanded effective context length by incorporating external retrieval, but lacked persistent storage or lifecycle management. Subsequent systems formalized memory as a managed and evolving resource rather than a static cache.

Among these, MemBank (Zhong et al., 2024) introduced a unified external memory store for continual distillation and adaptive overwriting. Mem0 (Chhikara et al., 2025) and its graph-based variant extended this paradigm to conversational agents, dynamically extracting, updating, and organizing salient facts from dialogue. Mem1 (Zhou et al., 2025) and A-Mem (Xu et al., 2025) further explored agentic and self-organizing memories, enabling models to autonomously create and refine memory entries as they interact. MemOS (Li et al., 2025) conceptualizes memory as a first-class operating resource, establishing unified governance and lifecycle management across parametric, activation, and external memories.

These developments collectively illustrate a transition from context extension to structured, persistent, and self-adaptive memory architectures that support long-term reasoning and continuity across sessions. But they generally treat reasoning and memory as loosely coupled components. In contrast, our approach integrates these elements within a unified, cognitively grounded framework. The memory hierarchy is explicitly inspired by Oberauer (2002)'s model, mapping the relationships among focus, working, and long-term memory to guide reasoning dynamics. Through tight interaction between the reasoning and memory agents, the system continuously refines and reconstructs context at each turn, rather than updating memory only after reasoning is complete. Furthermore, we introduce session-level lifecycle management that handles session inheritance, recycling, and selective long-term refinement, thereby enabling reasoning continuity and governance beyond what previous systems support.

B Method Details

B.1 Agent Definition

The **reasoning agent** is solely responsible for task-level inference and decision making, while the **memory agent** performs only lightweight memory operations, such as summarization, information extraction, and semantic retrieval, without participating in reasoning.

B.2 Memory Update Policy

The Memory Update Policy is determined directly by the LLM, during distillation, the Memory Agent decides whether an incoming memory trace should update an existing entry or create a new one. This choice was made intentionally to avoid fixed heuristics and leverage the model's semantic understanding.

B.3 Two-Stage Context Construction

Our framework includes a validation step in which the reasoning agent checks whether the summary produced by the memory agent contains enough information for the current task. If the reasoning agent detects missing or incomplete information, it will request that specific turns be expanded to obtain more detailed content. This prevents errors

caused by possible information loss during summarization and ensures that reasoning quality is not affected (as illustrated in the example below).

Example: Some parts are shortened as [...].

```
{"role": "user", "content": "You will be
↪  given the summary of chat history with
↪  turn id:\n\nChat history: ```\n**turn:**
↪  aa56a8af3c07eb95\nThe user entered the
↪  Proposal Stage, [...]\n\n**turn:**
↪  593da13289755cd8\nThe user informed the
↪  assistant that the current code (1, 2, 3)
↪  resulted in a FAIL for Verifier <4>,
↪  leading the assistant to analyze how this
↪  failure updates the constraints for V<4>
↪  based on previous results, and
↪  subsequently decide to test a new, highly
↪  informative Verifier <1> to gather
↪  orthogonal information.\n\n
↪  [...]```\n\nUser input:```[...there is
↪  the user input]```\n\nYour task:\n1.
↪  Analyze the user input carefully.\n2.
↪  Decide whether the summary and notes
↪  already provide enough context to respond
↪  accurately.\n3. **If they are
↪  sufficient**, you do not need to inspect
↪  any detailed chat turns, leave the
↪  `\"turn_ids\"` list empty (`[]`).\n4.
↪  **If you need specific details from one
↪  or more turns** to reason correctly, list
↪  the IDs of those turns in
↪  `\"turn_ids\"`.\n\nOutput json
↪  format:\n{\"reasoning\": \"Your
↪  reasoning about whether more details are
↪  needed\", \"turn_ids\": [\"ID of the
↪  history turn or keep list empty\",
↪  ...]}\n"}
```

```
{"role": "assistant", "content": "{\n
↪  \"reasoning\": \"The current turn
↪  provides the result [...], I need to
↪  re-verify the exact result from turn
↪  `593da13289755cd8` to ensure the previous
↪  deduction about V<4> was accurate and not
↪  based on a misunderstanding of how (1, 2,
↪  3) satisfies or fails
↪  criteria.\\n\\nSpecifically, I need to
↪  check the result for V<4> with code (1, 2,
↪  3) in turn `593da13289755cd8` to confirm
↪  the FAIL result.\",\n  \"turn_ids\":
↪  [\n\"593da13289755cd8\"\n]\n}"}
```

```
{"role": "user", "content": "**detailed turn
↪  items:** ```\n**turn:**
↪  593da13289755cd8\nuser_input: [...the
↪  detailed user
↪  input]\n\nassistant_response: [...the
↪  detailed assistant
↪  response]\n\n```\n\n**Existing
↪  notes**:\n[...detailed
↪  notes]\n\n**Related strategy**: \n[...
↪  detailed long term memory]\n\n**User
↪  input**:\n[... current user input /
↪  task]\n"}
```

C Experimental Setup

C.1 Benchmark

We follow the TurnBench-MS protocol (Zhang et al., 2025) with 45 setups for classic mode and three difficulty levels (easy, medium, hard). Each setup simulates a rule-discovery game requiring multi-turn reasoning and feedback integration.

C.2 Choice of Benchmark

Publicly available multi-turn reasoning benchmarks with verifiable turn-level ground truth are extremely limited. We use TurnBench-MS because it requires models to integrate feedback across many turns, provides verifiable feedback at each turn as well as an automatically checkable final answer, and offers controlled difficulty levels. These properties make it well suited for isolating the effect of memory on multi-turn reasoning.

CogMem is not specifically designed for TurnBench-MS. Its memory layers operate on generic dialogue turns, summaries, and notes, without relying on benchmark-specific structures or annotations. The framework also exposes an OpenAI-SDK-compatible interface, allowing it to be integrated into other multi-turn pipelines without task-specific changes to the memory architecture. Our experiments focus on TurnBench-MS; evaluating CogMem on additional benchmarks and task types is left for future work.

C.3 Model Configuration

- **Reasoning Agent.** We use Gemini 2.5 Flash as the main reasoning agent, with decoding parameters matching those reported in the TurnBench-MS paper to ensure a fair comparison. For the additional experiments in Appendix D.1, we use Qwen3-14B as the reasoning agent.
- **Memory Agent.** Gemini 2.5 Flash Lite is used for summarization, retrieval, and note maintenance, with its thinking mode disabled to reduce latency and cost.
- **Vector Storage.** Long-term memories are stored in Milvus (Wang et al., 2021) and retrieved via semantic similarity search.
- **Prompting.** All configurations, including the baseline, use chain-of-thought prompting with identical prompts and decoding parameters. The only difference between configurations is which memory components (FoA, DA, LTM) are enabled; the baseline uses none of them.

C.4 Evaluation Metrics

Accuracy is computed using scikit-learn (Pedregosa et al., 2011), comparing model outputs against ground-truth rules in each TurnBench–MS scenario. Scores are reported per difficulty level and evaluation mode.

D Additional Results

D.1 Results with Qwen3-14B

To examine whether the gains of CogMem transfer across base models, we repeat the classic-mode ablation with Qwen3-14B as the reasoning agent (Table 2). The trend closely mirrors that of Gemini 2.5 Flash: total accuracy increases as each memory layer is added, and the full system achieves the best performance at every difficulty level.

Models	Average Accuracy			
	Total	Easy	Medium	Hard
gemini-2.5-flash	0.76	0.87	0.93	0.47
+ FoA	0.78	0.93	0.87	0.53
+ FoA + DA	0.84	0.93	0.93	0.67
+ FoA + DA + LTM	**0.93**	**1.00**	**1.00**	**0.80**
qwen-3-14b	0.60	0.73	0.53	0.53
+ FoA	0.67	0.80	0.80	0.40
+ FoA + DA	0.73	0.87	0.80	0.53
+ FoA + DA + LTM	**0.89**	**1.00**	**0.93**	**0.73**
Random Guess	0.0085	0.0079	0.0098	0.0077

Table 2: Ablation results of CogMem with Gemini 2.5 Flash and Qwen3-14B under the TurnBench-MS classic mode settings. A random-guessing baseline is included for comparison. All results are obtained using the chain-of-thought (CoT) prompting strategy. The **bold** text represents the best performance.

D.2 Nightmare Mode Results

We further evaluate CogMem on the nightmare mode of TurnBench-MS, which introduces substantially higher noise and perturbations designed to destabilize multi-turn reasoning. Because this mode is considerably more expensive to run, we evaluate only the full CogMem configuration (Baseline + FoA + DA + LTM) and compare it with the reported Gemini 2.5 Flash results. Results are presented in Table 3.

Although absolute accuracy remains low, as is typical for this mode, CogMem improves overall accuracy from 0.18 to 0.24, with gains on the easy (0.13 to 0.27) and hard (0.13 to 0.27) setups. Performance on the medium setups decreases slightly (0.27 to 0.20), which corresponds to a difference of a single setup given 15 setups per difficulty level. Overall, these results suggest that the benefits of CogMem largely carry over to this more adversarial setting, although the per-level results are less stable than in the classic mode.

We do not report component-wise ablations (FoA, DA, LTM) in nightmare mode. Our primary focus is the classic mode, where evaluation is more controlled and reproducible, and where the ablations on both base models (Tables 1 and 2) show that each memory layer generally contributes to performance, with the full system achieving the best total accuracy. In addition, the high noise level of nightmare mode would make component-wise differences harder to interpret, as they could conflate robustness to perturbations with the contribution of individual memory layers.

Models	Average Accuracy			
	Total	Easy	Medium	Hard
Baseline - gemini-2.5-flash	0.18	0.13	0.27	0.13
Baseline + FoA + DA + LTM	0.24	0.27	0.20	0.27
Random Guess	0.0076	0.0074	0.0079	0.0075

Table 3: Performance comparison between the CogMem framework and the baseline model under the TurnBench–MS nightmare mode settings. A random-guessing baseline is included for comparison. All results are obtained using the chain-of-thought (CoT) prompting strategy.

E Qualitative Analysis

Qualitative evidence helps explain how CogMem improves reasoning, not only how much. We therefore present a case study, based on the same evaluation setup as the main experiments, that illustrates how CogMem changes the model's reasoning dynamics and prevents concrete failure modes of a vanilla LLM.

> **Sample case:** In this TurnBench–MS setup, the ground-truth code is 122.

Baseline (No CogMem). LLM eventually predicts 222. This is not a random failure: the full reasoning trace shows several systematic issues that are typical for long-chain LLM reasoning:

- **Premature commitment from a single observation**. After seeing only one <FAIL> from Verifier 1, the model immediately "locks in" a very strong claim: "Verifier 1 (<FAIL> with 143): The active criterion is 'no numbers in ascending order' ($B \geq Y$ and $Y \geq P$)." This is logically unjustified (many rules are consistent with a single FAIL) but is later treated as a hard constraint to prune the search space.

- **Hallucinating unobserved feedback**. Later, the model writes:"241 got V1 <PASS> while 143 got V1 <FAIL>." In the actual interaction, **241 was never tested against V1**. This is a clear hallucination of past feedback, caused by long, entangled context; that hallucinated fact is then used to justify a new interpretation of V1.
- **Silently redefining verifier semantics**. To reconcile the hallucinated "241 PASS / 143 FAIL" pattern, the model abandons the original semantics of V1 (about ascending sequences) and replaces it with an ad-hoc content rule: "The most consistent interpretation ... is that V1's active criterion is 'contains a 2'." This is outside the allowed rule family specified in the task description. Once this incorrect rule is written into the context, subsequent reasoning treats it as ground truth.

Without explicit control over memory, a long reasoning chain can drift, overwrite earlier assumptions, and hallucinate intermediate results. The failure is not random noise; it is structurally linked to how the model manages (or fails to manage) its own history.

In CogMem. On the same setup, CogMem correctly infers the true code 122. It avoids the above failure modes via three concrete design choices:

- **Notes extracted from feedback and conclusions.** After each interaction, a separate "memory" model summarizes the most important information into 'note', e.g.: "The active criterion for Verifier 2, when testing code 123, is PURPLE < 3." and "The true secret code must have PURPLE in 1, 2 based on Verifier 2." These notes are always traceable back to explicit <PASS>/<FAIL> signals and local logic. As a result, in the CogMem trace there is no analogue of the hallucinated "241 got V1 <PASS> while 143 got V1 <FAIL>" statement.
- **Two-stage context: summary first, targeted expansion on demand.** Instead of feeding the full raw history to the model, CogMem provides: a compact per-round summary (including only notes + a short description of what was tested), and the ability to request the full text of specific past rounds by ID if the summary is judged insufficient. In this puzzle, the model repeatedly reasons over the set of candidate rules, e.g.: "$Active V3 \in \{BLUE < YELLOW, BLUE < PURPLE\}$ based on V3(123)=PASS and V3(111)=FAIL." and "After testing 112 on V3: This PASS strongly suggests the active criterion for V3 is BLUE < PURPLE." Earlier "trial hypotheses" do not linger as raw text to be mis-remembered; only the vetted set-valued constraints are carried forward. This prevents early over-confident statements from hard-coding themselves into later reasoning, which is what happened in the baseline.
- **Separation of reasoning vs. memory**. Hypotheses stay local, constraints stay global. In CogMem, rich free-form reasoning stays in the current turn, while only distilled constraints (e.g., "$P \in \{1, 2\}$", "$V5 \in \{B = P, Y = P\}$") are written into notes. The subsequent turns consistently operate on these constraints, systematically intersecting them: From V2 and V4 notes: $digits \in \{1, 2, 4, 5\}$ and $P \in \{1, 2\}$; From V3: B < P; From V5 after 111 PASS and 112 FAIL: $V5 \in \{B = P, Y = P\}$, and final reasoning shows V5 must be Y=P; Combining these: P=2, B=1, Y=2, unique candidate 122.

TurnBench–MS also shows that multi-turn reasoning is easily affected by problems like memory decay and hallucination. CogMem is designed to reduce these issues: its Notes are updated continuously, and its reorganized context lets the reasoning agent pull in needed past information without being distracted by the full raw history. Together, these features help the model keep logical consistency across turns and avoid treating guesses or hallucinations as facts. This leads to more stable and accurate reasoning.

F Efficiency Analysis

F.1 Token Usage

Figure 2 compares token consumption across dialogue turns between CogMem and the model-only baseline. The baseline shows a near-linear increase in token usage as turn count rises, resulting from the continual accumulation of prior dialogue history in the prompt. In contrast, CogMem maintains a bounded and much flatter growth curve. Its focus-of-attention mechanism reconstructs a concise reasoning context at each turn, and its direct-access memory allows the model to reuse structured notes instead of reloading full historical text. This leads to a substantial reduction in token count—on average, CogMem uses less than half the tokens of the baseline after 15 turns, with the gap widening

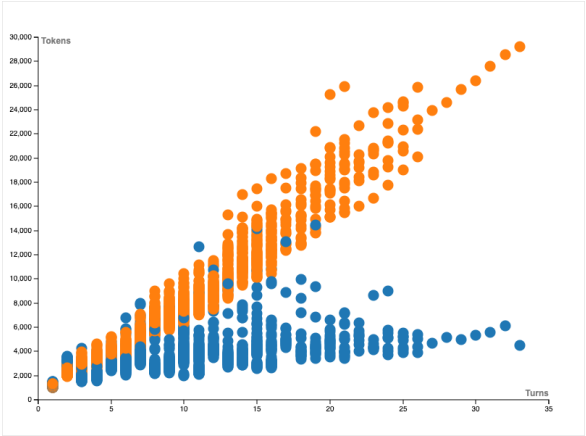


Figure 2: Token consumption per dialogue turn for CogMem (blue) and the model-only baseline (orange) on the TurnBench-MS classic mode. The x-axis represents dialogue turns, while the y-axis shows the number of tokens processed at each turn. CogMem maintains a stable token usage as conversations progress, whereas the model-only baseline exhibits nearly linear growth, reflecting the accumulation of full dialogue context.

as dialogues extend. The result confirms that CogMem's layered memory architecture not only improves reasoning accuracy but also enforces computational efficiency, making multi-turn reasoning more scalable for long or iterative interactions.

F.2 Latency and Computational Cost

CogMem was designed with efficiency as a core constraint, and we explicitly instrumented the framework with latency tracing to measure contributions from each component (FoA, DA, and LTM). Below we summarize the net overhead relative to a baseline model-only system.

- **Focus of Attention (FoA)**. FoA operations (turn hashing, cache lookup, and context reassembly) involve only lightweight string manipulation. These steps incur 0–1 ms of latency and no additional model inference. Thus, the FoA contributes negligible overhead.
- **Long-Term Memory (LTM) Retrieval**. LTM is queried only when a new session begins, not per turn. This step includes: one short summarization by the Memory Agent (<100 tokens in most case), one embedding computation using a lightweight local embedding model, and one vector search in a cloud-hosted Milvus/Zilliz database. The total latency for this sequence is typically <1 second, dominated by the network round-trip. Because this occurs once per session, the amortized per-turn cost is negligible.
- **Two-step Reasoning Workflow**. The system prompts the Reasoning Agent twice per turn: First pass: returns only a small JSON dict containing IDs of missing turns, which leads to very low token count, minimal latency; Second pass: produces the actual answer, same cost as a normal LLM inference. Thus, the additional cost versus baseline is one small, low-token LLM call per turn.
- **Asynchronous Memory Agent Operations**. All expensive Memory Agent tasks (turn summarization, note updating, long-term distillation) run asynchronously after the final answer is returned. These operations do not affect user-perceived latency and use a much cheaper model (Gemini 2.5 Flash Lite, 1/6 the cost of Gemini 2.5 Flash), with short generations (in most cases <100 tokens), making their cost marginal.
- **CogMem Reduces Long-horizon Cost**. A key advantage of CogMem is that its two-step reasoning workflow prevents context from growing linearly over turns, unlike the baseline model-only approach. As shown in Appendix F.1, baseline token usage increases nearly linearly with turn count, whereas CogMem's FoA + DA mechanism keeps the prompt size bounded and stable.

CogMem introduces minimal synchronous overhead while significantly reducing token consumption in long interactions. In practice, end-to-end latency remains dominated by the main Reasoning Agent's generation time, and the total computational cost becomes lower than the baseline as the number of turns increases.

G Code Availability

Please check our code at: https://github.com/grantzyr/CogMem