

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

Anonymous ACL submission

## 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 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 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,

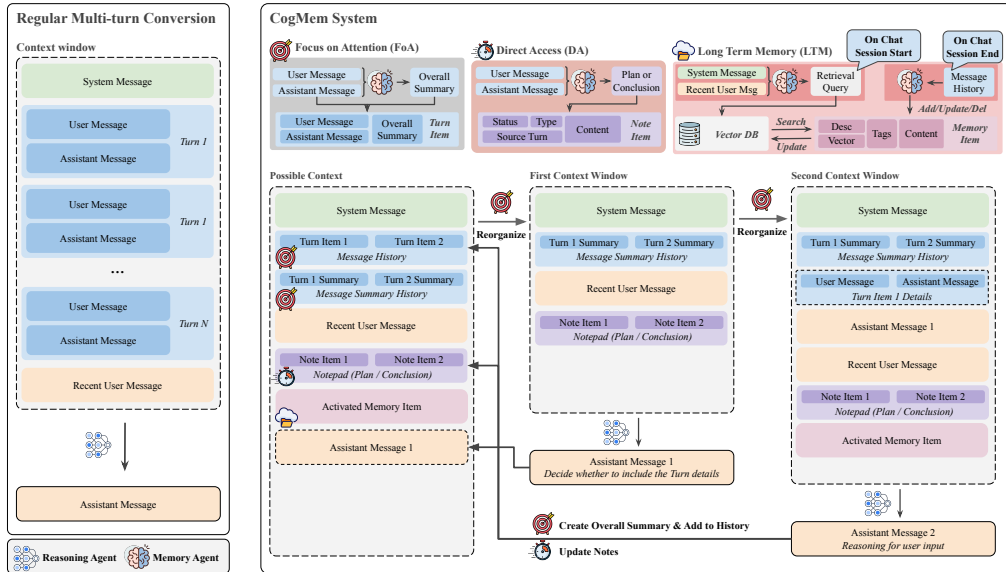


Figure 1: Overview of the CogMem framework.

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 C. 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

148	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.	197
149		198
150		199
151		200
152		201
153		202
154	Reasoning begins with constructing the first FoA, the <i>First Context Window</i> (Figure 1b), 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 <i>Second Context Window</i> . 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.	203
155		204
156		205
157		206
158		207
159		208
160		209
161		210
162		
163		
164		
165	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.	211
166		212
167		213
168		214
169		215
170		216
171		217
172		218
173	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.	219
174		220
175		221
176		222
177		223
178		224
179		225
180		226
181		227
182		228
183		229
184		230
185		231
186		232
187		233
188		234
189		235
190		236
191		237
192		
193		
194		
195		
196		
	<b>2.3 Interaction among Memory Layers</b>	
	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.	
	<b>2.4 Implementation and Efficiency</b>	
	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.	
	<b>2.5 Design Rationale and Error Mitigation</b>	
	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 revisiting 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.	
	<b>3 Experiment Setup and Results</b>	
	<b>3.1 Experimental Settings</b>	
	We evaluate our framework on the TurnBenchMS benchmark (Zhang et al., 2025), which is designed to assess iterative reasoning and adaptation in multi-turn interactions. TurnBench simulates rule-discovery games where models infer hidden patterns through sequential queries and feed-	

Models	Average Accuracy			
	Total	Easy	Medium	Hard
Baseline - gemini-2.5-flash	0.76	0.87	0.93	0.47
Baseline + FoA	0.76	0.93	0.84	0.53
Baseline + FoA + DA	0.84	0.93	0.93	0.66
Baseline + FoA + DA + LTM	<b>0.93</b>	<b>1.00</b>	<b>1.00</b>	<b>0.80</b>
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 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.

back. Following 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 standards (Pedregosa et al., 2011). Full experimental and implementation details are provided in Appendix A.

### 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, prompts, and memory configurations are in Appendix A.

### 3.3 Results and Analysis

Table 1 summarizes the performance for the classic mode of TurnBench-MS (see Appendix D for additional experiments). The baseline uses the official Gemini 2.5 Flash results, achieving an overall accuracy of 0.76. Introducing the FoA alone brings a small but consistent improvement on harder setups

in the classic environment, suggesting that dynamic context reconstruction helps reduce redundant reasoning and maintain focus, though its impact remains limited under high-noise conditions.

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 D.3.

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 E.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.

## 333 Limitations

334 While CogMem demonstrates clear improvements  
335 in reasoning stability and coherence, its current  
336 evaluation remains limited in scope. All exper-  
337 iments were conducted on the TurnBench-MS  
338 benchmark with a single base model, which con-  
339 strains the generalizability of the findings. Future  
340 work will extend the evaluation to a wider range of  
341 reasoning datasets, task types, and language mod-  
342 els to further validate CogMem’s adaptability and  
343 robustness across diverse multi-turn reasoning sce-  
344 narios.

## 345 Ethical

346 This work focuses on improving the reasoning  
347 stability and efficiency of large language models  
348 through structured memory design. All exper-  
349 iments were conducted using publicly available  
350 benchmark data (TurnBench-MS) and proprietary  
351 language models accessed through standard APIs.  
352 No personally identifiable or sensitive information  
353 was used in any stage of development or evaluation.  
354 The goal of CogMem is to enhance the reliabil-  
355 ity and interpretability of multi-turn reasoning sys-  
356 tems, which may reduce unintended hallucinations  
357 and overconfident outputs in deployed AI appli-  
358 cations. Nonetheless, improved long-term reason-  
359 ing capabilities also raise considerations regarding  
360 information persistence and user privacy. Future  
361 work will further examine safeguards for memory  
362 management, data retention, and user consent to  
363 ensure that cognitively inspired memory systems  
364 remain transparent, controllable, and aligned with  
365 ethical standards for responsible AI development.

## 366 References

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

373 Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie  
374 Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind  
375 Neelakantan, Pranav Shyam, Girish Sastry, Amanda  
376 Askell, Sandhini Agarwal, Ariel Herbert-Voss,  
377 Gretchen Krueger, Tom Henighan, Rewon Child,  
378 Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu,  
379 Clemens Winter, and 12 others. 2020. Language  
380 models are few-shot learners. In *Proceedings of the*  
381 *34th International Conference on Neural Information*  
382 *Processing Systems, NIPS ’20*, Red Hook, NY, USA.  
383 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, Mar-  
cel 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*. Ox-  
ford University Press.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio  
Petroni, Vladimir Karpukhin, Naman Goyal, Hein-  
rich Küttler, Mike Lewis, Wen-tau Yih, Tim Rock-  
täschel, and 1 others. 2020. Retrieval-augmented gen-  
eration 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 genera-  
tion (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 work-  
ing memory: Exploring the focus of attention. *Journal*  
*of Experimental Psychology: Learning, Memory,*  
*and Cognition*, 28(3):411–421.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gram-  
fort, Vincent Michel, Bertrand Thirion, Olivier Grisel,  
Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vin-  
cent Dubourg, and 1 others. 2011. Scikit-learn: Ma-  
chine 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 sys-  
tem. In *Proceedings of the 2021 international con-*  
*ference 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.

- 441 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten  
442 Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou,  
443 and 1 others. 2022. Chain-of-thought prompting elicits  
444 reasoning in large language models. *Advances*  
445 *in neural information processing systems*, 35:24824–  
446 24837.
- 447 Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zu-  
448 jie Liang, and Yongfeng Zhang. 2025. A-mem:  
449 Agentic memory for llm agents. *arXiv preprint*  
450 *arXiv:2502.12110*.
- 451 Yiran Zhang, Mo Wang, Xiaoyang Li, Kaixuan  
452 Ren, Chencheng Zhu, and Usman Naseem. 2025.  
453 [TurnBench-MS: A benchmark for evaluating multi-](#)  
454 [turn, multi-step reasoning in large language models.](#)  
455 In *Findings of the Association for Computational Lin-*  
456 *guistics: EMNLP 2025*, pages 19892–19924, Suzhou,  
457 China. Association for Computational Linguistics.
- 458 Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and  
459 Yanlin Wang. 2024. Memorybank: Enhancing large  
460 language models with long-term memory. In *Pro-*  
461 *ceedings of the AAAI Conference on Artificial Intelli-*  
462 *gence*, volume 38, pages 19724–19731.
- 463 Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan  
464 Kim, Alok Prakash, Daniela Rus, Jinhua Zhao,  
465 Bryan Kian Hsiang Low, and Paul Pu Liang. 2025.  
466 Mem1: Learning to synergize memory and reason-  
467 ing for efficient long-horizon agents. *arXiv preprint*  
468 *arXiv:2506.15841*.

## A Experimental Details

### A.1 Benchmark Setup

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 iterative feedback integration.

### A.2 Model Configuration

- **Reasoning Agent** Gemini 2.5 Flash is used with standard chain-of-thought prompts. Parameters match those reported in the TurnBench paper to ensure fair comparison.
- **Memory Agent** Gemini 2.5 Flash Lite is used for summarization, retrieval, and note maintenance. Its generative reasoning is disabled to improve speed and reduce cost.
- **Vector Storage** All memory retrievals use Milvus (Wang et al., 2021). Session and long-term memories are stored as vectors and retrieved using semantic similarity queries.

### A.3 Prompting and CoT Setup

- All models use chain-of-thought prompting.
- Prompts are consistent across all configurations.
- Memory components are only active in FoA, DA, and LTM settings, not in the baseline.

### A.4 Baselines and Ablations

- **Baseline:** Gemini 2.5 Flash with CoT, no memory.
- **Baseline + FoA:** Adds focus-of-attention for bounded context reconstruction.
- **Baseline + FoA + DA:** Adds session-level direct-access memory.
- **Baseline + FoA + DA + LTM:** Full memory hierarchy for cross-session knowledge accumulation.
- **Random-guessing:** Lower-bound reference.

### A.5 Evaluation Metrics

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

### A.6 Implementation Notes

- Short-term notes, turns, and session metadata are stored in RAM with optional external caching (Redis).

- LTM is stored in a vectorized database supporting selective updates.
- Reasoning and memory agents run asynchronously; the memory agent updates DA after reasoning steps.
- Session manager handles identification, caching, expiration, and garbage collection.

All hyperparameters, session management protocols, and retrieval configurations are fully documented in the internal code repository (link omitted for anonymization in submission).

### A.7 Addtional Experiments

Models	Average Accuracy			
	Total	Easy	Medium	Hard
gemini-2.5-flash	0.76	0.87	0.93	0.47
+ FoA	0.76	0.93	0.84	0.53
+ FoA + DA	0.84	0.93	0.93	0.66
+ FoA + DA + LTM	<b>0.93</b>	<b>1.00</b>	<b>1.00</b>	<b>0.80</b>
qwen-3-14b	0.6	0.73	0.53	0.53
+ FoA	0.66	0.8	0.8	0.4
+ FoA + DA	0.74	0.9	0.8	0.53
+ FoA + DA + LTM	0.87	1.0	0.93	0.73
Random Guess	0.0085	0.0079	0.0098	0.0077

Table 2: Performance comparison between the CogMem framework and the baseline model under the TurnBench 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.

## B 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

548	models to articulate intermediate steps, improv-	long-term refinement, thereby enabling reasoning	599
549	ing single-instance reasoning but remaining insuf-	continuity and governance beyond what previous	600
550	ficient for multi-turn or session-level reasoning,	systems support.	601
551	where memory continuity and selective recall are		
552	essential. Our work builds upon these insights, us-		
553	ing TurnBench as a testbed while introducing an		
554	explicit structured memory framework to improve		
555	reasoning persistence and efficiency.		
556	<b>Memory-Augmented LLMs and Long-Term In-</b>	<b>C Method Extension</b>	602
557	<b>teraction</b> Recent research increasingly views		
558	memory as a core component of scalable LLM	<b>C.1 Agent Definition</b>	603
559	systems. Early approaches such as Retrieval-		
560	Augmented Generation (RAG) (Lewis et al., 2020)	The <b>reasoning agent</b> is solely responsible for task-	604
561	expanded effective context length by incorporating	level inference and decision making, while the	605
562	external retrieval, but lacked persistent storage or	<b>memory agent</b> performs only lightweight mem-	606
563	lifecycle management. Subsequent systems formal-	ory operations, such as summarization, information	607
564	ized memory as a managed and evolving resource	extraction, and semantic retrieval, without partici-	608
565	rather than a static cache.	pating in reasoning.	609
566	Among these, MemBank (Zhong et al., 2024)		
567	introduced a unified external memory store for	<b>C.2 Implementation</b>	610
568	continual distillation and adaptive overwriting.		
569	Mem0 (Chhikara et al., 2025) and its graph-based	<b>The Memory Update Policy</b> is determined directly	611
570	variant extended this paradigm to conversational	by the LLM, during distillation, the Reasoning	612
571	agents, dynamically extracting, updating, and or-	Agent decides whether an incoming memory trace	613
572	ganizing salient facts from dialogue. Mem1 (Zhou	should update an existing entry or create a new one.	614
573	et al., 2025) and A-Mem (Xu et al., 2025) further	This choice was made intentionally to avoid fixed	615
574	explored agentic and self-organizing memories, en-	heuristics and leverage the model’s semantic under-	616
575	abling models to autonomously create and refine	standing. We will add a clear description of this	617
576	memory entries as they interact. MemOS (Li et al.,	mechanism to the appendix as well.	618
577	2025) conceptualizes memory as a first-class oper-		
578	ating resource, establishing unified governance and	<b>C.3 Case Study</b>	619
579	lifecycle management across parametric, activation,		
580	and external memories	Our framework includes a validation step in which	620
581	These developments collectively illustrate a transi-	the reasoning agent checks whether the summary	621
582	tion from context extension to structured, persist-	produced by the memory agent contains enough	622
583	ent, and self-adaptive memory architectures that	information for the current task. If the reasoning	623
584	support long-term reasoning and continuity across	agent detects missing or incomplete information,	624
585	sessions. But they generally treat reasoning and	it will request that specific turns be expanded to	625
586	memory as loosely coupled components. In con-	obtain more detailed content. This prevents errors	626
587	trast, our approach integrates these elements within	caused by possible information loss during summa-	627
588	a unified, cognitively grounded framework. The	ri- zation and ensures that reasoning quality is not	628
589	memory hierarchy is explicitly inspired by Ober-	affected (as illustrated in the example below).	629
590	auer (2002)’s model, mapping the relationships		
591	among focus, working, and long-term memory to	<b>Example:</b> Some parts are shortened as [...].	630
592	guide reasoning dynamics. Through tight interac-		
593	tion between the reasoning and memory agents, the		
594	system continuously refines and reconstructs con-		
595	text at each turn, rather than updating memory only		
596	after reasoning is complete. Furthermore, we intro-		
597	duce session-level lifecycle management that han-		
598	dles session inheritance, recycling, and selective		

```

{"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\":
↪ [\"593da13289755cd8\"]\n"}

{"role": "user", "content": "**detailed turn
↪ items:** ```\n**turn:**
↪ 593da13289755cd8\nuser_input: [...the
↪ detailed user
↪ input]\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"}

```

## D Experiment Extension

### D.1 TurnBench Nightmare Results

we conducted an additional evaluation on the nightmare mode of TurnBench-MS. Due to the significantly higher computational cost of nightmare mode, we evaluated the full CogMem architecture

(Baseline + FoA + DA + LTM). The Results are presented in Table 3.

While the absolute numbers remain low, as is typical for nightmare mode, the results show consistent improvements relative to the reported Gemini 2.5 Flash performance (18% overall accuracy), indicating that CogMem offers measurable gains even in adversarial reasoning environments.

We did not include full ablations (FoA, DA, LTM) in nightmare mode due to its intensive cost and because our primary focus remains on the classic mode, where controlled, reproducible evaluation is standard and where the ablation trend is already clean and monotonic (Table 1). Given that the nightmare mode setup introduces very high noise and intentional perturbations designed to destabilize reasoning, ablations there would be substantially harder to interpret and risk conflating robustness to adversarial perturbations with architectural contributions.

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 nightmare mode settings. A random-guessing baseline is included for comparison. All results are obtained using the chain-of-thought (CoT) prompting strategy.

### D.2 More Benchmarks

Current open multi-turn reasoning benchmarks with verifiable turn-level ground truth are extremely limited. After careful selection, we used TurnBench to evaluate the performance of our current framework.

At the same time, we would like to clarify that CogMem is designed as a task-agnostic architectural framework, rather than a dataset-specific solution. We also provide support for the OpenAI SDK, which makes using CogMem as simple as calling a standard model. This allows CogMem to be applied to any dataset with minimal effort.

### D.3 Qualitative Analysis

The qualitative evidence is important to understand how CogMem improves reasoning rather than only how much, we provide a qualitative case study, based on the same evaluation setup as the main experiments, that directly illustrates how CogMem

631  
632  
633  
634  
635  
636

637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
  
670  
671  
672  
673  
674  
675

676 changes the model’s reasoning dynamics and pre-  
677 vents concrete failure modes of a vanilla LLM.

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

678  
679 **Baseline (No CogMem).** LLM eventually pre-  
680 dict 222. This is not a random failure: the full  
681 reasoning trace shows several systematic issues  
682 that are typical for long-chain LLM reasoning:

- 683 • **Premature commitment from a single observa-**  
684 **tion.** After seeing only one <FAIL> from Verifier  
685 1, the model immediately “locks in” a very strong  
686 claim: “Verifier 1 (<FAIL> with 143): The ac-  
687 tive criterion is ‘no numbers in ascending order’  
688 (B Y and Y P).” This is logically unjustified  
689 (many rules are consistent with a single FAIL)  
690 but is later treated as a hard constraint to prune  
691 the search space.
- 692 • **Hallucinating unobserved feedback.** Later, the  
693 model writes: “241 got V1 <PASS> while 143  
694 got V1 <FAIL>.” In the actual interaction, **241**  
695 **was never tested against V1.** This is a clear  
696 hallucination of past feedback, caused by long,  
697 entangled context; that hallucinated fact is then  
698 used to justify a new interpretation of V1.
- 699 • **Silently redefining verifier semantics.** To rec-  
700 oncile the hallucinated “241 PASS / 143 FAIL”  
701 pattern, the model abandons the original seman-  
702 tics of V1 (about ascending sequences) and re-  
703 places it with an ad-hoc content rule: “The most  
704 consistent interpretation . . . is that V1’s active  
705 criterion is ‘contains a 2’.” This is outside the  
706 allowed rule family specified in the task descrip-  
707 tion. Once this incorrect rule is written into the  
708 context, subsequent reasoning treats it as ground  
709 truth.

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

716 **In CogMem.** On the same setup, CogMem cor-  
717 rectly infers the true code 122. It avoids the above  
718 failure modes via three concrete design choices:

- 719 • **Notes extracted from feedback and conclu-**  
720 **sions.** After each interaction, a separate “mem-  
721 ory” model summarizes the most important infor-  
722 mation into ‘note’, e.g.: “The active criterion for

723 Verifier 2, when testing code 123, is PURPLE <  
724 3.” and “The true secret code must have PURPLE  
725 in 1, 2 based on Verifier 2.” These notes are al-  
726 ways traceable back to explicit <PASS>/<FAIL>  
727 signals and local logic. As a result, in the Cog-  
728 Mem trace there is no analogue of the halluci-  
729 nated “241 got V1 <PASS> while 143 got V1  
730 <FAIL>” statement.

- 731 • **Two-stage context: summary first, targeted**  
732 **expansion on demand.** Instead of feeding the  
733 full raw history to the model, CogMem provides:  
734 a compact per-round summary (including only  
735 notes + a short description of what was tested),  
736 and the ability to request the full text of specific  
737 past rounds by ID if the summary is judged in-  
738 sufficient. In this puzzle, the model repeatedly  
739 reasons over the set of candidate rules, e.g.: “Ac-  
740 tive V3 BLUE < YELLOW, BLUE < PURPLE  
741 based on V3(123)=PASS and V3(111)=FAIL.”  
742 and “After testing 112 on V3: This PASS strongly  
743 suggests the active criterion for V3 is BLUE  
744 < PURPLE.” Earlier “trial hypotheses” do not  
745 linger as raw text to be mis-remembered; only  
746 the vetted set-valued constraints are carried for-  
747 ward. This prevents early over-confident state-  
748 ments from hard-coding themselves into later  
749 reasoning, which is what happened in the base-  
750 line.
- 751 • Separation of reasoning vs. memory: hypotheses  
752 stay local, constraints stay global. In CogMem,  
753 rich free-form reasoning stays in the current turn,  
754 while only distilled constraints (e.g., “P 1,2”,  
755 “V5 B=P, Y=P”) are written into notes. The sub-  
756 sequent turns consistently operate on these con-  
757 straints, systematically intersecting them: From  
758 V2 and V4 notes: digits 1,2,4,5 and P 1,2; From  
759 V3: B < P; From V5 after 111 PASS and 112  
760 FAIL: V5 B=P, Y=P, and final reasoning shows  
761 V5 must be Y=P; Combining these: P=2, B=1,  
762 Y=2, unique candidate 122.

763 TurnBench also shows that multi-turn reasoning  
764 is easily affected by problems like memory decay  
765 and hallucination. CogMem is designed to reduce  
766 these issues: its Notes are updated continuously,  
767 and its reorganized context lets the reasoning agent  
768 pull in needed past information without being dis-  
769 tracted by the full raw history. Together, these  
770 features help the model keep logical consistency  
771 across turns and avoid treating guesses or hallu-  
772 cinations as facts. This leads to more stable and  
773 accurate reasoning.

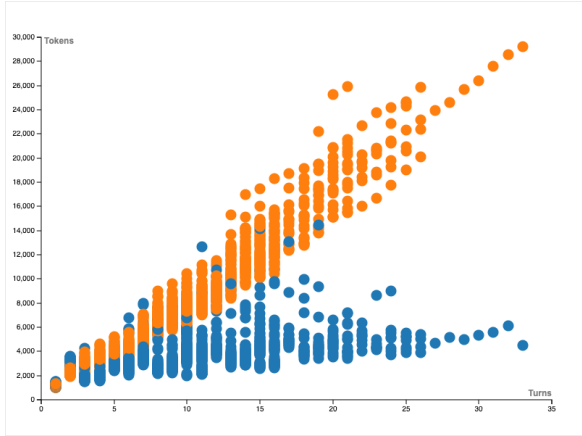


Figure 2: Token consumption per dialogue turn for CogMem (blue) and the model-only baseline (orange) on the TurnBench-MS classic mode. The x-axis represents dialogue turns, while the y-axis shows the number of tokens processed at each turn. CogMem maintains a stable token usage as conversations progress,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

## E System Overhead

### E.1 Efficiency and Token Usage Analysis

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 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.

### E.2 Latency and Total 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 the 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 lead 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 E.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.