MEMOGRAPH: AUGMENTING LLMS WITH EXPLICIT EPISODIC MEMORY FOR MULTI-STEP MATHEMATICAL REASONING

Yutong Li, Yitian Zhou, Guo Chen, Xudong Wang, Chenghao Li & Chaoning Zhang*
University of Electronic Science and Technology of China

ABSTRACT

Large Language Models (LLMs) fundamentally struggle with complex mathematical reasoning due to the volatility of their implicit parametric memory, which leads to context drift and hallucination. Existing paradigms, relying on linear generation or static retrieval, fail to maintain a precise, persistent record of the evolving proof state. To address this, we propose MemoGraph, a neuro-symbolic framework that augments LLMs with an explicit episodic memory layer. We formulate reasoning as the dynamic maintenance of a heterogeneous graph, enabling state-aware reading that utilizes graph-structural encoding to retrieve relevant principles from a verified semantic memory. Furthermore, we introduce a write-gating verification module to intercept invalid deductions before they are consolidated into the reasoning context. Empirical evaluations across multiple benchmarks demonstrate that MemoGraph significantly outperforms strong baselines in both accuracy and robustness by ensuring memory integrity, establishing a scalable paradigm for trustworthy reasoning agents.

1 INTRODUCTION

Large Language Models (LLMs) have achieved remarkable milestones in mathematical reasoning, demonstrating capabilities that range from elementary arithmetic to complex competition-level challenges (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2023). Despite these advances, their reliability diminishes sharply as reasoning chains grow in length and complexity. A fundamental limitation is contextual drift: as derivation steps accumulate, the model often loses track of the precise logical prerequisites established in earlier steps, leading to error accumulation where a single hallucinatory step propagates downstream (Zhang et al., 2025; Dziri et al., 2023).

We attribute this fragility to the inherent limitations of LLMs' implicit parametric memory. While autoregressive generation captures local dependencies, it lacks a mechanism to maintain a precise, persistent episodic memory of the evolving reasoning state. Mathematical reasoning effectively involves a dynamic traversal of a knowledge graph (Lightman et al., 2024), where the applicability of a specific theorem is conditioned on intermediate derived states rather than solely on the initial problem description. For example, in geometry, the Pythagorean Theorem becomes relevant only after a right-angled triangle is explicitly constructed or identified. Current approaches fail to model this dynamic dependency adequately. Chain-of-Thought (CoT) (Wei et al., 2022; Yao et al., 2023) relies entirely on internal parametric knowledge, effectively overburdening the model's limited working memory with accumulating context, which inevitably leads to hallucination and context drift (Pan et al., 2023). Recent studies also highlight the structural limitations of linear reasoning chains through topological data analysis (Li et al., 2025a), prompting the need for more complex reasoning paradigms such as syzygy-based minimal free resolution (Li et al., 2025b). Similarly, standard Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Trivedi et al., 2023) generally performs retrieval using the original question context. This static retrieval strategy fails to capture the shifting information needs of the evolving memory state, often leading to the retrieval of irrelevant or insufficient knowledge as the derivation progresses (Mallen et al., 2023).

*Corresponding author.

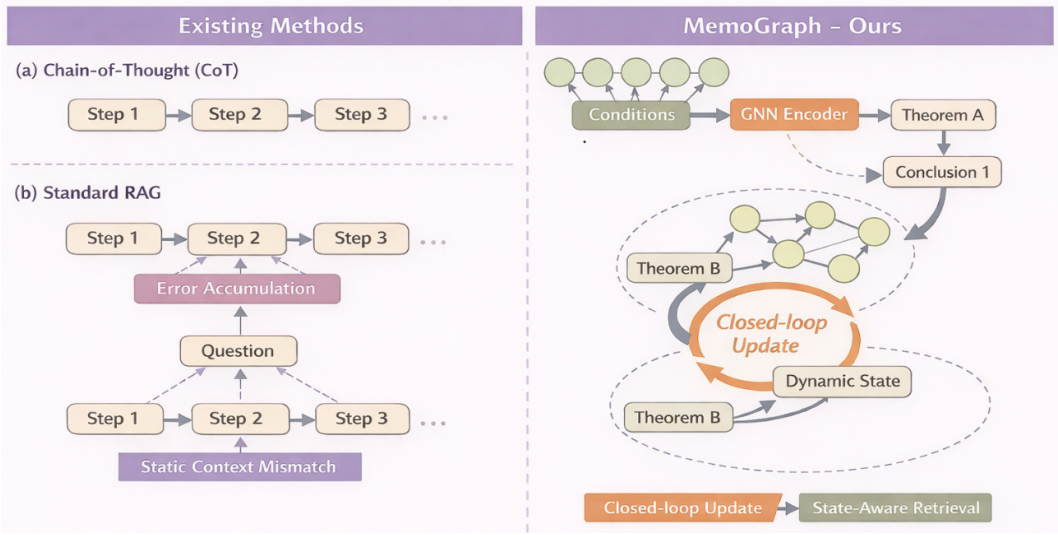

Figure 1: **Comparison of Reasoning Paradigms.** Unlike (a) CoT and (b) Standard RAG, (c) **MemoGraph** utilizes a GNN to encode evolving states for dynamic theorem retrieval.

To bridge this gap, we propose **MemoGraph**, a graph-based explicit memory layer designed for reasoning agents. Instead of treating reasoning as a transient sequence of tokens, MemoGraph models it as the iterative maintenance of a heterogeneous episodic memory graph. In this structure, nodes represent atomic conditions, formal theorems, and derived conclusions, explicitly encoding the logical topology of the problem. At each reasoning step, a relational GNN encodes the current graph structure into a dense state vector. Crucially, this vector serves as a state-aware query to read the most appropriate principle from the semantic memory, which comprises our verified theorem library. This enables the model to perform state-conditioned retrieval, adapting knowledge selection based on the accumulated coherent context, effectively mimicking a mathematician's deliberate memory recall. To further ensure memory integrity, we employ a consistency verification strategy to filter out invalid deductions before they are consolidated into the graph.

To align with the cognitive architecture of agentic systems, we explicitly map our components to memory primitives. We instantiate the verified theorem library as semantic memory to store long-term static knowledge, while the evolving reasoning graph functions as episodic memory by preserving the structured history of the current problem. Furthermore, our topological linearization acts as a working memory management policy that compresses the graph into the context window.

Our contributions are summarized as follows:

- We propose **MemoGraph**, which introduces a structured episodic memory layer to the reasoning process. This explicit memory structure overcomes the volatility of parametric memory, allowing the agent to maintain a precise state across long reasoning horizons.
- We develop a GNN-guided memory read mechanism. By encoding the topological semantics of the episodic state, our method achieves state-conditioned retrieval from semantic memory, significantly surpassing static, query-based RAG approaches.
- Extensive experiments on multiple datasets demonstrate that MemoGraph achieves superior performance over strong baselines. Notably, our framework is VRAM-efficient compared to topology-search methods while remaining deployable on commodity hardware.

2 Related Work

2.1 LLMs for Mathematical Reasoning

Recent progress in mathematical reasoning has evolved from prompting strategies to process-oriented supervision. Foundational methods like Chain-of-Thought (CoT) (Wei et al., 2022) and its extensions, such as Self-Consistency (Wang et al., 2023) and Tree of Thoughts (ToT) (Yao et al., 2023), encourage LLMs to decompose complex problems into intermediate steps. While effective,

these approaches operate purely within the model's parametric memory, leaving them vulnerable to hallucinations when internal knowledge is imprecise or outdated (Ji et al., 2023). To further optimize model elicitation, recent studies have also explored structural evolution in soft prompt tuning (Huang et al., 2026b), indicating that embedding structural awareness directly into the tuning process can significantly enhance performance. A parallel line of research focuses on fine-tuning specialized models, such as LLaMA-2-Math (Touvron et al., 2023) and DeepSeek-Math (Shao et al., 2024). More recently, Process Supervision (Lightman et al., 2024) has emerged as a promising direction, training reward models to verify each reasoning step. However, these methods typically require expensive human-annotated process labels or separate reward models. In contrast, MemoGraph grounds reasoning in an external semantic memory constructed from a verified theorem library. By treating theorems as executable rules, we achieve rigorous step-level verification without the need for massive process-supervision datasets. This architecture effectively offloads the burden from the model's fragile parametric memory to a reliable explicit storage system, bridging the gap between parametric intuition and symbolic rigor.

2.2 Retrieval-Augmented Generation (RAG)

RAG (Lewis et al., 2020) addresses the knowledge gap by retrieving relevant documents from external corpora. While transformative for open-domain QA (Guu et al., 2020), standard RAG struggles with the sequential dependency of mathematical deduction. Most approaches employ static retrieval based on the initial problem statement, which fails to capture the evolving information needs of multi-step proofs (Jiang et al., 2023). Iterative RAG methods (Trivedi et al., 2023) attempt to mitigate this by retrieving context at each step. However, they predominantly rely on text-based similarity of the immediate predecessor sentence. This shallow interaction often misses deep logical dependencies—for instance, realizing that a geometric property implies a specific algebraic theorem. MemoGraph advances this paradigm by encoding the full reasoning topology into a graph state. Our GNN-based mechanism acts as a state-conditioned memory reader, capturing the structural semantics of the problem to retrieve principles that match the current episodic state, a capability that static text-based heuristics cannot match.

2.3 Neuro-Symbolic and Graph-Based Reasoning

Graph-based LLM approaches generally bifurcate into knowledge-centric methods (Yasunaga et al., 2021; Pan et al., 2024), which leverage static KGs for factual recall but lack procedural modeling, and structure-centric methods like Graph of Thoughts (GoT) (Besta et al., 2024) that explore non-linear solution paths. However, GoT focuses on topology search over unstructured text nodes without semantic typing, and its extensive sampling requirements often prohibit real-time deployment (Ning et al., 2024). MemoGraph unifies these paradigms by employing a heterogeneous graph with distinct node types (Condition, Theorem, Conclusion). The importance of maintaining a global structural view has also been echoed in recent advances, such as leveraging global hypothesis spaces to enhance synergistic reasoning chains (Zhang et al., 2026b) and utilizing persistent homology for structure-aware text summarization (Zhang et al., 2026a). Furthermore, while recent Graph-based RAG methods like Think-on-Graph (Sun et al., 2024) and GraphRAG (Edge et al., 2024) utilize structured representations for knowledge retrieval, they primarily focus on factual question-answering rather than multi-step mathematical deduction. Similarly, programmatic solvers such as MathCoder (Wang et al., 2024) and PAL (Gao et al., 2023) excel at computational correctness through code integration but lack an explicit episodic memory to maintain evolving logical states. MemoGraph bridges these paradigms by combining structured state maintenance with logical deduction.

3 Method

We propose MemoGraph, a closed-loop reasoning framework that combines graph neural networks and large language models for multi-step mathematical problem solving. At a high level, MemoGraph builds an explicit episodic memory graph over conditions, theorems, and conclusions, and uses a graph encoder to drive theorem retrieval and step-wise verification. This section first introduces the construction of the symbolic theorem library that serves as semantic memory, and then

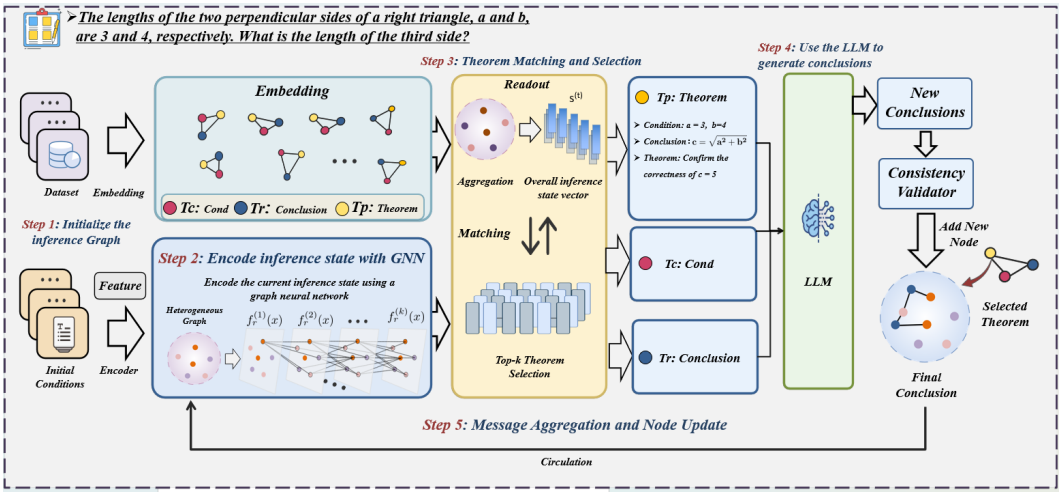

Figure 2: **MemoGraph** as a closed-loop memory layer for reasoning agents. Starting from initial conditions, MemoGraph first initializes an evolving inference graph that functions as episodic memory (Step 1). It then encodes the current graph state with a GNN (Step 2) and performs a state-aware read operation from the semantic memory (theorem library) (Step 3). The topological linearization manages working memory to prompt the LLM (Step 4). Finally, a write gating mechanism verifies and consolidates valid conclusions back into the graph (Step 5).

describes the graph-based reasoning loop and training objectives. The overall architecture is illustrated in Figure 2.

3.1 Automated Construction of Semantic Memory

To ground reasoning in rigorous rules rather than raw text, we establish a symbolic theorem library $\mathcal{T}$ that serves as the agent's semantic memory containing abstract, reusable axioms. The construction pipeline begins by prompting an LLM to distill generalized axioms from training solutions, such as the GSM8K dataset. In this phase, specific entities are converted into variables, and relations are formalized into symbolic expressions. For example, the phrase "5 apples" is abstracted to "N objects", while geometric properties are standardized into formal definitions like "Pythagorean Theorem: $a^2 + b^2 = c^2$". To eliminate redundancy among these extracted entries, the rules are encoded into dense embeddings $\mathbf{e}_T$ via Sentence-BERT (Reimers & Gurevych, 2019) and subsequently grouped using agglomerative clustering with a threshold of $\tau = 0.85$. The centroid of each resulting cluster is then designated as the canonical theorem T_j.

3.2 Episodic Memory Graph

The core of MemoGraph is the explicit modeling of the episodic memory as a dynamic graph evolution. Unlike linear chains, our graph captures the non-linear dependencies between facts and rules, maintaining a structured history of the problem state.

3.2.1 Graph Formalization

We formalize the episodic state at step t as a directed heterogeneous graph $\mathcal{G}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$, where the node set $\mathcal{V}^{(t)}$ comprises three distinct semantic types essential to mathematical proofs. Specifically, Conditions (C) represent atomic facts derived from the problem statement, such as initial quantities; Theorems (T) correspond to abstract rules retrieved from semantic memory $\mathcal{T}$; and Conclusions (Z) denote intermediate results generated by applying a theorem to a set of conditions.

To capture the logical derivation paths, the edge set $\mathcal{E}^{(t)}$ utilizes a relation set $\mathcal{R} = \{\texttt{Support}, \texttt{Apply}, \texttt{Derive}\}$ to model specific dependencies. A relation $(c, T, \texttt{Support})$ ex-

plicitly indicates that a condition c satisfies a premise of theorem T, while $(T, z, \texttt{Derive})$ signifies that theorem T logically produces a conclusion z. This explicit typing scheme enables the downstream GNN to semantically distinguish between "input data" and "reasoning rules" during the inference process.

3.2.2 INITIALIZATION AND FEATURE ENCODING

A critical challenge is bootstrapping this structured memory from an unstructured problem text Q, particularly when ground-truth logical forms are unavailable at test time. We address this via a two-step automated initialization process:

1. Atomic Decomposition: We employ a lightweight LLM as a zero-shot parser. By providing a few-shot prompt that demonstrates how to segment math problems into discrete facts, we instruct the model to decompose Q into a list of atomic conditions $\{c_1, \ldots, c_m\}$. We enforce a strict JSON output format to ensure parsing stability. These parsed facts form the initial node set $\mathcal{V}^{(0)}$.

2. Semantic Vectorization: To enable neural processing, we initialize the feature vector $\mathbf{h}_i^{(0)} \in \mathbb{R}^d$ for each node v_i. We utilize a pre-trained Sentence-BERT (Reimers & Gurevych, 2019) encoder to map the textual content of each node into a dense semantic embedding. For example, a node containing the text "John has 5 apples" is converted into its corresponding vector representation:

$$\mathbf{h}_i^{(0)} = \text{SBERT}(\text{text}(v_i)). \tag{1}$$

This initialization ensures that the graph carries both the structural topology (via nodes) and the semantic content (via embeddings) necessary for the subsequent GNN reasoning loop.

3.3 THE MEMOGRAPH FRAMEWORK

We adopt a closed-loop design for structured reasoning. At each step t, the system operates on the episodic graph $\mathcal{G}^{(t)}$ and executes five core modules.

3.3.1 GRAPH ENCODING

To capture the evolving logical state, we employ a Relational Graph Convolutional Network (RGCN) (Schlichtkrull et al., 2018). Unlike standard GCNs that treat all edges uniformly, RGCN explicitly differentiates between dependency types, such as `Support` and `Derive`. The node embedding $\mathbf{h}_i^{(k)}$ at layer k is updated by aggregating messages from neighbors $\mathcal{N}_i^r$ under each relation $r \in \mathcal{R}$:

$$\mathbf{h}_i^{(k+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}\mathbf{W}_r^{(k)}\mathbf{h}_j^{(k)} + \mathbf{W}_0^{(k)}\mathbf{h}_i^{(k)}\right), \tag{2}$$

where $c_{i,r}$ serves as a normalization constant and $\mathbf{W}_r^{(k)}$ denotes relation-specific weight matrices. After K layers, we obtain a graph-level state vector $\mathbf{s}^{(t)}$ via an attention-based readout mechanism. This operation prioritizes nodes that represent the reasoning frontier, specifically focusing on the most recently generated conclusions:

$$\mathbf{s}^{(t)} = \sum_{v_i\in\mathcal{V}^{(t)}} \alpha_i \mathbf{h}_i^{(K)}, \quad \alpha_i = \text{Softmax}(\mathbf{q}^T\mathbf{h}_i^{(K)}), \tag{3}$$

where $\mathbf{q}$ is a learnable query vector. This resulting $\mathbf{s}^{(t)}$ functions as the semantic fingerprint of the current proof progress.

3.3.2 STATE-CONDITIONED MEMORY RETRIEVAL

Unlike static RAG, our retrieval mechanism functions as a dynamic memory read operation conditioned on the evolving graph state. We compute the relevance score between the current state vector $\mathbf{s}^{(t)}$ and each pre-encoded theorem embedding $\mathbf{t}_j$ stored in the semantic memory:

$$\text{score}(T_j|\mathcal{G}^{(t)}) = \frac{\mathbf{s}^{(t)}\cdot\mathbf{t}_j}{\|\mathbf{s}^{(t)}\|\|\mathbf{t}_j\|}. \tag{4}$$

Based on these scores, we select the top-k candidate theorems. In our primary implementation, we utilize the highest-ranked theorem T^* to guide the next reasoning step. This mechanism acts as a neural compass, pivoting the search direction based on intermediate derivations to effectively solve the contextual drift problem.

3.3.3 CONCLUSION GENERATION AND CONTEXT MANAGEMENT

To apply the selected theorem T^*, we formulate a structured prompt for the LLM. Simply flattening the entire graph $\mathcal{G}^{(t)}$ into text may introduce noise and exceed context limits. Instead, we employ a **Topological Linearization** strategy as a working memory management policy: we extract the subgraph $\mathcal{G}_{sub}^{(t)}$ containing the ancestors of the current frontier nodes and serialize them in topological order. The LLM then generates a candidate conclusion $\hat{z}^{(t)}$:

$$\hat{z}^{(t)} = \text{LLM}(\texttt{Prompt}(\mathcal{G}_{sub}^{(t)}) \oplus \texttt{Def}(T^*)). \tag{5}$$

This ensures the model attends to the specific dependencies required by the theorem while maintaining the logical derivation sequence.

3.3.4 CONSISTENCY VERIFICATION AND WRITE GATING

To intercept hallucinations, we employ a hybrid consistency validator that acts as a write-gating mechanism for the episodic memory. This module operates automatically:

- **Type Check (S_{type}):** A rule-based parser verifies if the output format aligns with T^*'s signature (e.g., ensuring a generated "velocity" is a vector, not a scalar).
- **Premise Check (S_{prem}):** We perform a symbolic lookup against the current graph node set $\mathcal{V}^{(t)}$. This ensures that any entity referenced in the generated conclusion (e.g., variables, geometric shapes) has been previously defined or derived, effectively filtering out hallucinations of non-existent objects.
- **Semantic Check (S_{sem}):** To detect subtle logical contradictions, we utilize an off-the-shelf NLI model (DeBERTa-v3-large) to compute the entailment score between $\hat{z}^{(t)}$ and the existing facts in $\mathcal{G}^{(t)}$.

The conclusion is accepted only if the validity $V = S_{type} \wedge S_{prem} \wedge (S_{sem} > \delta)$ holds. We empirically set $\delta = 0.8$. If rejected, the system automatically discards $\hat{z}^{(t)}$ and backtracks to attempt the next-best theorem from the retrieval step.

3.3.5 MEMORY CONSOLIDATION

Upon validation, we update the graph to $\mathcal{G}^{(t+1)}$, effectively consolidating the new state into episodic memory. This involves adding the new conclusion node $z^{(t)}$, the theorem node T^*, and the directed edges that encode their provenance. The node set update is defined as:

$$\mathcal{V}^{(t+1)} \leftarrow \mathcal{V}^{(t)} \cup \{T^*, z^{(t)}\}. \tag{6}$$

The edge set is updated by connecting premises to the theorem (Support), and the theorem to the conclusion (Derive):

$$\begin{aligned} \mathcal{E}^{(t+1)} \leftarrow \mathcal{E}^{(t)} \ &\cup \ \{(c, T^*, \texttt{Support}) \mid c \in \text{Premises}\} \\ &\cup \ \{(T^*, z^{(t)}, \texttt{Derive})\}. \end{aligned} \tag{7}$$

This closed-loop update ensures that the GNN encoder in the next iteration $(t+1)$ captures the newly derived state.

3.4 TRAINING OBJECTIVES

To enable accurate memory retrieval, we train the framework to align the episodic state vector $\mathbf{s}^{(t)}$ with the embedding of the ground-truth theorem T^+, denoted as $\mathbf{t}^+$. We first define the similarity score for the positive pair at step t:

$$s_t^+ = \text{sim}(\mathbf{s}^{(t)}, \mathbf{t}^+), \tag{8}$$

where the function $\mathrm{sim}(\mathbf{u}, \mathbf{v})$, defined as $\mathbf{u}^T\mathbf{v}/(\|\mathbf{u}\|\|\mathbf{v}\|)$, represents the cosine similarity metric between the two state vectors.

We optimize the alignment using the InfoNCE loss (Oord et al., 2018), which maximizes the probability of selecting the correct theorem while suppressing irrelevant ones. The loss function for step t is formulated as:

$$\mathcal{L}_{\text{match}}^{(t)} = -\log\left(\frac{\exp(s_t^+/\tau)}{\exp(s_t^+/\tau) + \sum_{T_j \in \mathcal{N}_t}\exp(s_{j,t}/\tau)}\right), \tag{9}$$

where $s_{j,t} = \mathrm{sim}(\mathbf{s}^{(t)}, \mathbf{t}_j)$ represents the similarity score for a negative sample T_j, and τ is a temperature hyperparameter. To construct the negative set $\mathcal{N}_t$, we sample a fixed number of incorrect theorems from the library such that $\mathcal{N}_t \subset \mathcal{T} \setminus \{T^+\}$. For computational efficiency, we employ in-batch negative sampling, treating theorems from other samples in the same batch as negatives. The total loss is minimized over all reasoning steps to encourage the graph encoder to develop discriminative, context-aware representations.

4 Experiments

4.1 Experimental Setup

We evaluate MemoGraph on three benchmarks of increasing difficulty: **GSM8K** (Cobbe et al., 2021), **MATH** (Hendrycks et al., 2021), and **AIME** (He et al., 2024), covering the spectrum from grade school arithmetic to Olympiad-level problems. To verify our method's effectiveness, we compare it against a diverse set of baselines spanning four paradigms: intrinsic prompting strategies, including Standard CoT (Wei et al., 2022) and CoT-SC (Wang et al., 2023); structure-centric methods such as Graph of Thoughts (Besta et al., 2024); static retrieval baselines like Standard RAG (Lewis et al., 2020); and supervised neuro-symbolic approaches, including LLM-FT (Hu et al., 2022) and LLM-ARC (Kalyanpur et al., 2024). This selection allows us to isolate the specific contributions of our dynamic memory mechanism against static context, topological search, and direct fine-tuning.

For implementation, we consistently employ Qwen2.5-Math-7B-Instruct as the backbone model across all experiments to ensure fair comparability. In the RAG baseline, we utilize Sentence-BERT (Reimers & Gurevych, 2019) for top-k ($k = 3$) retrieval based on query similarity. For the fine-tuning baseline, we apply Low-Rank Adaptation (LoRA) (Hu et al., 2022) to match the learnable capacity of our framework. All evaluations are conducted with a temperature of 0 to guarantee reproducibility. To mitigate training stochasticity, we report the average performance over three independent runs. Comprehensive hyperparameters and training details are provided in the Appendix.

4.2 Main Results

Table 1 reports accuracy across three benchmarks. MemoGraph consistently outperforms all baselines, achieving the best performance with both the math-specialized Qwen2.5 backbone and general-purpose backbones such as LLaMA-3 and Mistral. This robustness indicates that our framework provides transferable structural guidance rather than relying on backbone-specific heuristics.

Crucially, MemoGraph yields significant gains over static retrieval methods such as RAG-Theorem (Lewis et al., 2020), particularly on the challenging MATH and AIME datasets. This result confirms that conditioning memory read operations on an explicit, evolving episodic state is far more effective for multi-step deduction than static question-level retrieval. Furthermore, our method surpasses both full fine-tuning (LLM-FT) and neuro-symbolic critics like LLM-ARC (Kalyanpur et al., 2024). Since all three approaches leverage training data, this comparison isolates the specific advantage of MemoGraph's graph-conditioned theorem selection and step-wise verification, demonstrating that structured memory management contributes more to reasoning reliability than mere data exposure or unstructured critique.

4.3 Ablation Study

To dissect the contribution of each component, we conduct an ablation study on the MATH dataset (Table 2).

Table 1: Comparative analysis of **MemoGraph** against standard reasoning paradigms on three benchmarks. We evaluate the effectiveness of our explicit Episodic Memory mechanism compared to implicit memory baselines (e.g., CoT, Self-Consistency) and static retrieval methods (e.g., RAG-Theorem).

Method	GSM8K	MATH	AIME	Avg
Qwen2.5-Math-7B-Instruct				
Standard CoT (Wei et al., 2022)	95.6	83.6	26.7	68.6
CoT-SC (Wang et al., 2023)	96.1	84.8	30.0	70.3
Graph of Thoughts (Besta et al., 2024)	95.8	84.5	31.2	70.5
RAG-Theorem (Lewis et al., 2020)	95.9	84.2	29.5	69.9
LLM-FT (Full Fine-tuning)	96.0	85.1	28.5	69.9
LLM-ARC (Kalyanpur et al., 2024)	96.2	85.5	32.1	71.3
MemoGraph (Ours)	**96.8**	**86.9**	**36.4**	**73.4**
LLaMA-3-8B-Instruct				
Standard CoT (Wei et al., 2022)	79.6	30.0	4.5	38.0
CoT-SC (Wang et al., 2023)	81.8	32.5	5.8	40.0
Graph of Thoughts (Besta et al., 2024)	81.2	31.8	6.2	39.7
RAG-Theorem (Lewis et al., 2020)	83.5	36.2	8.4	42.7
LLM-FT (Full Fine-tuning)	85.2	39.5	9.5	44.7
LLM-ARC (Kalyanpur et al., 2024)	85.8	40.2	10.1	45.4
MemoGraph (Ours)	**88.4**	**44.5**	**14.2**	**49.0**
Mistral-7B-Instruct-v0.3				
Standard CoT (Wei et al., 2022)	76.2	28.4	3.2	35.9
CoT-SC (Wang et al., 2023)	78.5	30.1	4.5	37.7
Graph of Thoughts (Besta et al., 2024)	78.0	29.8	4.8	37.5
RAG-Theorem (Lewis et al., 2020)	80.1	33.5	6.5	40.0
LLM-FT (Full Fine-tuning)	82.4	36.8	7.8	42.3
LLM-ARC (Kalyanpur et al., 2024)	83.1	37.5	8.2	42.9
MemoGraph (Ours)	**85.6**	**41.2**	**11.8**	**46.2**

Table 2: Ablation study on the MATH dataset (Backbone: Qwen2.5-Math-7B-Instruct). We remove each core component individually to evaluate its contribution. Δ denotes the performance drop relative to the full model.

Variant	Accuracy (%)	Δ
MemoGraph (Full Model)	**86.9**	-
w/o Verification (Write Gating)	86.1	-0.8
w/o Memory Consolidation	85.3	-1.6
w/o Episodic State (Static RAG)	84.2	-2.7
w/o Semantic Memory (Pure CoT)	83.6	-3.3

Impact of Consistency Verification. Removing the verification module leads to a 0.8% drop. While numerically modest, this margin is critical for high-precision reasoning, as a single hallucinated step can invalidate an entire chain. Our validator acts as a gatekeeper, filtering out subtle fallacies that advanced models overlook.

Effectiveness of Graph-Conditioned Retrieval. Replacing GNN-based encoding with static retrieval (w/o Episodic State) causes a significant 2.7% degradation, dropping performance to near-RAG levels. This highlights that static queries fail to capture the evolving proof state. Our graph encoding ensures retrieved theorems align with the current reasoning frontier rather than the initial context.

Necessity of Memory Consolidation. Disabling graph updates (w/o Memory Consolidation) results in a 1.6% decline. Without consolidating new deductions into the graph, the model cannot leverage intermediate conclusions for subsequent retrieval. This confirms that MemoGraph's power lies in

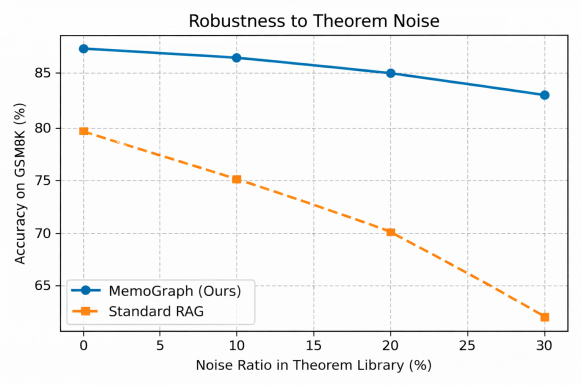

Figure 3: Robustness analysis on GSM8K with varying noise ratios. MemoGraph maintains stability even at 20% noise.

Table 3: Efficiency comparison on GSM8K (Backbone: Qwen2.5-Math-7B-Instruct). MemoGraph achieves the best accuracy-efficiency balance. Measured on a single NVIDIA RTX 4090 (24GB).

Method	**Acc (%)**	**Time (s)**	**Mem (GB)**
Standard CoT	95.6	**3.8**	**15.2**
RAG-Theorem	95.9	4.5	16.5
Graph of Thoughts	95.8	18.2	23.4
MemoGraph	**96.8**	6.5	19.1

its closed-loop design, where new knowledge actively triggers the retrieval of advanced theorems initially irrelevant to the start state.

4.4 Robustness Analysis

Since the theorem library is constructed via an automated pipeline (Section 3.1), it may inevitably contain noise. To evaluate the resilience of MemoGraph against such imperfections, we conducted a robustness analysis by intentionally injecting noise into the theorem library. Specifically, we replaced a certain proportion of valid theorems (ranging from 0% to 30%) with irrelevant or erroneous entries and measured the performance on the GSM8K dataset.

As illustrated in Figure 3, the performance of standard RAG degrades rapidly as the noise ratio increases. This decline occurs because RAG relies solely on semantic similarity, a metric that becomes unreliable when the retrieval corpus is polluted. In contrast, MemoGraph exhibits strong robustness, suffering only a marginal performance drop even at a 20% noise level. This stability is primarily attributed to the synergy between our graph-conditioned retrieval, which leverages structural context to filter out irrelevant noisy theorems, and the consistency verification module, which actively rejects incorrect applications of noisy rules during inference.

4.5 Computational Efficiency

We evaluate the inference efficiency of MemoGraph using the Qwen2.5-Math-7B-Instruct backbone on a single NVIDIA RTX 4090 (24GB), with results summarized in Table 3.

When compared to lightweight baselines such as Standard CoT (3.8s), MemoGraph incurs a moderate latency cost (6.5s) attributable to the overhead of GNN encoding and verification. However, given the complexity of multi-step reasoning tasks, this 1.7$\times$ latency increase is a justifiable trade-off for the substantial gains in reasoning stability and accuracy. More importantly, against structured reasoning baselines like Graph of Thoughts (GoT), MemoGraph demonstrates superior efficiency. It is nearly $\mathbf{3}\times$ faster (6.5s compared to 18.2s) and significantly more memory-efficient (19.1GB compared to 23.4GB). While GoT's complex topology search pushes the VRAM limits of consumer GPUs, MemoGraph maintains a practical balance between rigorous structural reasoning and deployability, rendering it suitable for real-world applications.

5 CONCLUSION

In this work, we proposed **MemoGraph**, a neuro-symbolic framework that augments LLMs with an explicit episodic memory graph. Unlike linear generation paradigms, MemoGraph treats mathematical reasoning as the dynamic maintenance of a structured state, enabling a GNN-guided read mechanism to perform precise, context-aware theorem retrieval from semantic memory. By integrating a rigorous write-gating verification module, our system effectively filters hallucinations and ensures memory integrity. Empirical results across three benchmarks demonstrate that MemoGraph significantly outperforms static retrieval and prompting baselines in both accuracy and robustness, while maintaining computational efficiency. We believe this paradigm of structured memory evolution offers a robust foundation for next-generation reasoning agents, with promising potential for extension to code generation and formal verification.

6 ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grant No. 62572104).

REFERENCES

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. *Advances in neural information processing systems*, 36:70293–70332, 2023.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Zhenzhen Huang, Haoyu Bian, Jiaquan Zhang, Yibei Liu, Kuien Liu, Caiyan Qin, Guoqing Wang, Yang Yang, and Chaoning Zhang. Rethinking input domains in physics-informed neural networks via geometric compactification mappings. *arXiv preprint arXiv:2602.16193*, 2026a.

Zhenzhen Huang, Chaoning Zhang, Haoyu Bian, Songbo Zhang, Chi-lok Andy Tai, Jiaquan Zhang, Caiyan Qin, Jingjing Qu, Yalan Ye, Yang Yang, and Heng Tao Shen. Optimizing soft prompt tuning via structural evolution. *arXiv preprint arXiv:2602.16500*, 2026b.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Aditya Kalyanpur, Kailash Karthik Saravanakumar, Victor Barres, Jennifer Chu-Carroll, David Melville, and David Ferrucci. Llm-arc: Enhancing llms with an automated reasoning critic. *arXiv preprint arXiv:2406.17663*, 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

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

Chenghao Li, Chaoning Zhang, Yi Lu, Shuxu Chen, Xudong Wang, Jiaquan Zhang, Zhicheng Wang, Zhengxun Jin, Kuien Liu, Sung-Ho Bae, Guoqing Wang, Yang Yang, and Heng Tao Shen. Understanding chain-of-thought in large language models via topological data analysis. *arXiv preprint arXiv:2512.19135*, 2025a.

Chenghao Li, Chaoning Zhang, Yi Lu, Jiaquan Zhang, Qigan Sun, Xudong Wang, Jiwei Wei, Guoqing Wang, Yang Yang, and Heng Tao Shen. Syzygy of thoughts: Improving llm cot with the minimal free resolution. *arXiv preprint arXiv:2504.09566*, 2025b.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2024.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 9802–9822, 2023.

Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. Skeleton-of-thought: Prompting llms for efficient parallel generation. In *The Twelfth International Conference on Learning Representations*, 2024.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, 2023.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599, 2024.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607. Springer, 2018.

Zhihong Shao, Peiyi Wang, Qiuju Zhu, Runxin Xu, Junxiao Song, Mingchuan Xiao, YK Li, Y Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*, 2024.

Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. In *The Twelfth International Conference on Learning Representations*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 535–546, 2021.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Yibei Liu, Chenghao Li, Qigan Sun, Shuai Yuan, Fachrina Dewi Puspitasari, Dongshen Han, Guoqing Wang, Sung-Ho Bae, and Yang Yang. Text summarization via global structure awareness. In *The Fourteenth International Conference on Learning Representations*, 2026a.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Xudong Wang, Zhenzhen Huang, Pengcheng Zheng, Shuai Yuan, Sheng Zheng, Qigan Sun, Jie Zou, LIK-HANG LEE, and Yang Yang. Learning global hypothesis space for enhancing synergistic reasoning chain. In *The Fourteenth International Conference on Learning Representations*, 2026b.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew C Yao. Cumulative reasoning with large language models. *Transactions on Machine Learning Research*, 2025.

Pengcheng Zheng, Chaoning Zhang, Jiarong Mo, GuoHui Li, Jiaquan Zhang, Jiahao Zhang, Sihan Cao, Sheng Zheng, Caiyan Qin, Guoqing Wang, and Yang Yang. Llava-fa: Learning fourier approximation for compressing large multimodal models. In *The Fourteenth International Conference on Learning Representations*, 2026.

A Appendix A: Semantic Memory Construction Details

The construction of our semantic memory $\mathcal{T}$ (formerly referred to as Theorem Library) is automated to ensure scalability while maintaining high precision for deductive reasoning.

A.1 Knowledge Extraction Pipeline

As described in Section 3, we prompt a teacher LLM to extract abstract, reusable theorems from training solutions. Specific entities are replaced with variables, and relations are formalized.

Extraction Prompt. We instruct the teacher LLM with the following prompt: "Given a mathematical solution step, identify the underlying theorem or formula. Abstract specific numbers into variables and output the formal definition." For example, given the step "$c^2 = 3^2 + 4^2$ in a right triangle", the model outputs the Pythagorean Theorem $a^2 + b^2 = c^2$.

A.2 Vectorization and Canonicalization

To eliminate redundancy, we encode extracted rules using Sentence-BERT and apply agglomerative clustering with a threshold of $\tau = 0.85$. The centroid of each cluster is selected as the canonical entry in the semantic memory.

Illustrative Example.

- **Raw Extractions (Cluster):**
  - "For any right triangle, $a^2 + b^2 = c^2$."
  - "The hypotenuse squared equals the sum of the legs squared."
  - "In a 90-degree triangle, $c = \sqrt{a^2 + b^2}$."
- **Canonical Entry (Centroid):**
  - **Pythagorean Theorem**: $a^2 + b^2 = c^2$.

B Appendix B: Implementation Details

We formally present the closed-loop inference process of MemoGraph in Algorithm 1, followed by specific configuration parameters.

Algorithm 1 MemoGraph: Closed-Loop Reasoning with Episodic Memory

Require: Problem Q, semantic memory $\mathcal{T}$
Ensure: Final Answer A
Initialize Episodic Memory:
  $C \leftarrow \text{AtomicDecomposition}(Q)$
  $\mathcal{G}^{(0)} \leftarrow (C, \emptyset)$; $t \leftarrow 0$
while not `Terminate` **do**
  {1. State-Aware Memory Read}
  $\mathbf{s}^{(t)} \leftarrow \text{GNN\_Encoder}(\mathcal{G}^{(t)})$
  $\{T_1, \ldots, T_k\} \leftarrow \text{Retrieve}(\mathbf{s}^{(t)}, \mathcal{T})$
  for T^* in $\{T_1, \ldots, T_k\}$ **do**
    {2. Working Memory Processing}
    $\hat{z} \leftarrow \text{LLM}(\text{Linearize}(\mathcal{G}^{(t)}), T^*)$
    {3. Write-Gating Verification}
    if $\text{Verify}(\hat{z}, \mathcal{G}^{(t)}) ==$ VALID **then**
      {4. Consolidate to Episodic Memory}
      $\mathcal{G}^{(t+1)} \leftarrow \text{UpdateGraph}(\mathcal{G}^{(t)}, T^*, \hat{z})$
      $t \leftarrow t + 1$
      break
    end if
    {Backtrack: Try next candidate theorem}
  end for
end while
return $\text{ExtractAnswer}(\mathcal{G}^{(Final)})$

B.1 Model Architecture

Episodic Memory Encoder (RGCN). Consistent with the methodology, our graph encoder parameters are:

- **Input Dimension (d_{in}):** 768. This corresponds to the embedding size of the pre-trained Sentence-BERT model (`all-mpnet-base-v2`) used to initialize node features.
- **Hidden Dimension (d_{hidden}):** 256.
- **Number of Layers (K):** 3. We employ a 3-layer Relational Graph Convolutional Network (RGCN) to capture multi-hop dependencies.
- **Readout Mechanism:** Attention-based pooling with a learnable query vector of dimension 256.

Heterogeneous Graph Schema. To explicitly model logical dependencies and memory provenance, we define a set of relation types $\mathcal{R} = \{\texttt{Support}, \texttt{Apply}, \texttt{Derive}\}$. `(Condition, Theorem, Support)` indicates that a condition satisfies a premise of a theorem. `(Theorem, Conclusion, Derive)` signifies that a theorem logically produces a conclusion. `(Conclusion, Theorem, Apply)` indicates that a derived conclusion is used as a premise for a subsequent theorem application.

B.2 Training and Inference Configuration

Retrieval Training. We train the retrieval module using Contrastive Learning (InfoNCE loss) on the training splits of GSM8K and MATH. The model is optimized using AdamW ($\beta_1 = 0.9, \beta_2 = 0.999$) with a batch size of 32. We set the initial learning rate to 1×10^{-4} for the RGCN encoder, employing a linear warmup for the first 10% of steps followed by cosine decay. For the loss function, we use a temperature $\tau = 0.07$ and augment the training with negative sampling, utilizing both in-batch negatives and 3 hard negatives retrieved via BM25 per sample.

Generation Config. For the Conclusion Generation module, we employ three backbone models: Qwen2.5-Math-7B-Instruct, LLaMA-3-8B-Instruct, and Mistral-7B-Instruct. To ensure repro-

ducibility and minimize hallucination variance, we strictly use greedy decoding (Temperature $= 0$) for all reasoning steps. We set the maximum generation length to 512 tokens per step and apply a repetition penalty of 1.05 to prevent loop degradation. To ensure reproducibility across all experiments, we fix the random seeds to $42, 123, 2024$ and report the average performance over three runs.

C Appendix C: Prompt Templates

In this section, we provide the specific prompts used in the Atomic Decomposition, Memory-Augmented Solver, and Verification modules.

C.1 Atomic Decomposition Prompt

To ensure high-quality decomposition, our system prompt enforces a rigorous protocol where each extracted condition must be atomic, containing exactly one semantic fact, and fully self-contained so that it remains understandable without external context. Furthermore, we mandate a strict JSON output format to guarantee parsing stability and seamless integration with the downstream graph initialization module. We initialize the Episodic Memory by decomposing the problem statement Q into atomic conditions.

Instruction: You are an expert at breaking down math problems. Given a math problem, decompose it into a list of atomic conditions (facts). Each condition must be a self-contained sentence. Output strictly in JSON format with a single key condition.

Input Problem: A triangle ABC has sides $AB = 3$ and $AC = 4$, and the angle A is $90°$. Find the length of BC.

Response:

```
{
  "conditions": [
    "Shape is a triangle ABC.",
    "Side length AB is 3.",
    "Side length AC is 4.",
    "Angle A is 90 degrees."
  ]
}
```

C.2 Memory-Augmented Solver Prompt

Once a theorem T^* is retrieved from semantic memory, we construct a prompt that combines the current reasoning state (linearized subgraph) and the retrieved definition.

System: You are a rigorous mathematical reasoning agent. You have access to a verified semantic memory. Use the provided Retrieved Theorem to deduce the next logical step based on the "Current Context".

Current Context: 1. Shape is a triangle ABC. 2. Side length AB is 3. 3. Side length AC is 4. 4. Angle A is 90 degrees. 5. (Derived) Triangle ABC is a right-angled triangle.

Retrieved Theorem: pythagorean theorem: In a right-angled triangle with legs a, b and hypotenuse c, $a^2 + b^2 = c^2$.

Task: Apply the theorem to the context to generate a new conclusion. Output the conclusion in a single sentence.

Output: Applying the Pythagorean Theorem to sides AB and AC, the length of BC is $\sqrt{3^2 + 4^2} = 5$.

C.3 Consistency Verification Prompt

As part of our Write Gating mechanism, we verify conclusions before memory consolidation.

Instruction: Verify if the "Candidate Conclusion" logically follows from the "Premises". Return "VALID" only if the conclusion is a strict logical consequence. Otherwise, return "INVALID".

Premises: - Triangle ABC is right-angled at A. - AB=3, AC=4. - Pythagorean Theorem: $a^2 + b^2 = c^2$.

Candidate Conclusion: The area of triangle ABC is 12.

Output: INVALID (Reasoning: Area is $0.5 \times base \times height = 0.5 \times 3 \times 4 = 6$, not 12.)

D Appendix D: Datasets

We evaluate MemoGraph on three widely used mathematical reasoning benchmarks. Below, we provide detailed descriptions and our specific processing protocols for each dataset.

D.1 Benchmarks Description

- **GSM8K (Grade School Math) Cobbe et al. (2021):** A dataset of 8.5k high-quality grade school math word problems. It consists of 7,473 training samples and 1,319 test samples. The problems require multi-step arithmetic reasoning but rely on relatively simple mathematical concepts.
- **MATH (Mathematics Aptitude Test of Heuristics) Hendrycks et al. (2021):** A more challenging dataset containing 12,500 competition-level problems from AMC 10, AMC 12, and AIME. It covers diverse domains such as algebra, geometry, and number theory. We utilize the standard split of 7,500 training and 5,000 test samples. The answers often involve complex LaTeX formatting (e.g., fractions, matrices).
- **AIME (American Invitational Mathematics Examination) He et al. (2024):** To evaluate out-of-distribution (OOD) robustness, we construct an AIME test set using problems from the OlympiadBench collection (2024 and recent years). These problems represent a significant leap in difficulty, requiring the creative retrieval of advanced semantic knowledge and rigorous logical deduction.

D.2 Data Processing and Evaluation Protocol

Unlike standard prompting baselines that consume raw text, MemoGraph requires structured inputs and rigorous answer verification. Our protocol is designed as follows:

Semantic Knowledge Preprocessing. Since MemoGraph relies on an explicit semantic memory, we do not simply feed the raw question Q into the model. Instead, for the training sets of GSM8K and MATH, we perform Automated Knowledge Extraction:

1. We use a teacher LLM to analyze the ground-truth solution and extract the implicit mathematical rules (e.g., "Triangle Inequality") used in each step.
2. These extracted rules are canonicalized to form our **Semantic Memory** $\mathcal{T}$ (details in Appendix B).

Note: For AIME (OOD setting), we do not train on it. The model must generalize using the semantic memory learned from MATH/GSM8K, testing its transferability.

Answer Extraction and Matching. Different datasets require distinct extraction strategies to ensure fair evaluation:

- **For GSM8K:** We use a rule-based parser to locate the numeric value after the `"####"` separator. Correctness is determined by strict string matching of the integer value.
- **For MATH:** Ground-truth answers are enclosed in `\boxed{}`. We employ the official normalization script provided by Hendrycks et al. Hendrycks et al. (2021) to handle LaTeX variations (e.g., treating $\frac{1}{2}$ and 0.5 as equivalent) before comparison.

- **For AIME (Strict Integer Constraint):** AIME problems strictly require an integer answer between 000 and 999. Given the complexity of the reasoning chains generated by MemoGraph, standard regex often fails. We adopt a hybrid extraction strategy:
  1. First, we attempt to extract the answer using the pattern "`The final answer is \boxed{NUM}`".
  2. If this fails or yields a non-integer, we employ a lightweight LLM parser to identify the final integer result from the generated reasoning graph.

E Appendix E: Baselines and Backbone Details

In this section, we provide the justification for our backbone selection and detailed configurations for all baseline methods.

E.1 Backbone Models Selection

We select three diverse LLMs to evaluate the universality of MemoGraph across different capabilities. **Qwen2.5-Math-7B-Instruct** is chosen as the representative of state-of-the-art math-specialized models, allowing us to test if MemoGraph can improve even highly optimized backbones. **LLaMA-3-8B-Instruct** is included as the current general-purpose SOTA open-source model, while **Mistral-7B-Instruct** serves as a widely-adopted open-source standard to verify architectural transferability.

E.2 Baseline Implementations

All baselines are implemented following their official configurations to ensure a rigorous comparison.

Standard Prompting (CoT & CoT-SC). We utilize the standard 8-shot Chain-of-Thought prompts with greedy decoding (Temperature $= 0$). For Self-Consistency (CoT-SC), to ensure path diversity, we sample $K = 10$ reasoning paths with a temperature of $T = 0.7$ and select the final answer via majority voting.

Retrieval-Augmented Generation (RAG-Standard). We employ a hybrid retriever combining sparse BM25 and dense Sentence-BERT. For each problem, we retrieve the top-$k = 3$ most relevant theorems based on the initial question text. This baseline serves as a direct control to demonstrate the superiority of our state-aware memory reading over static retrieval strategies.

Graph of Thoughts (GoT). Following the official implementation by Besta et al. (2024), we utilize the aggregation and refine operations. To ensure a fair comparison of inference budget, we restrict GoT to a maximum of 10 thought nodes per problem.

Supervised and Neuro-Symbolic Baselines. For the **LLM-FT** baseline, we perform parameter-efficient fine-tuning using LoRA (Rank $r = 64$, Alpha $\alpha = 16$, Dropout $p = 0.1$) for 3 epochs with a learning rate of $2e^{-5}$ using DeepSpeed ZeRO-2. Finally, for **LLM-ARC** Kalyanpur et al. (2024), we implement the critic module as a lightweight Python executor that verifies arithmetic correctness at each reasoning step.

F Appendix F: Limitations and Future Work

While MemoGraph establishes a reliable explicit episodic memory for mathematical reasoning, it currently has a few limitations. First, the greedy single-trajectory inference nature of our framework lacks a backtracking mechanism. Once an intermediate step is written into the episodic graph, it cannot be revised. This greedy approach limits performance on creative problem-solving tasks like AIME, where heuristic "tricks" and trial-and-error exploration are often required. Future work could integrate exploration algorithms, such as Beam Search or Monte Carlo Tree Search (MCTS), over the episodic graph to enable multi-path exploration.

Second, while the write-gating stage guarantees logical consistency and structural format, it does not currently guarantee absolute computational correctness. Arithmetic or calculus mistakes made by the LLM might bypass the semantic checks and be permanently stored in the graph. Augmenting the write-gating mechanism with an external symbolic engine (e.g., SymPy) or Python code-execution would effectively catch computational errors.

Finally, our current approach focuses on text-based axiomatic deduction in mathematics. Two promising directions emerge for future extensions. On one hand, many mathematical problems (e.g., Olympiad geometry) rely heavily on visual diagrams. Extending MemoGraph to incorporate visual chain-of-thought reasoning and employing efficient multimodal models (Zheng et al., 2026) could enable sophisticated multimodal deductions. On the other hand, the dual-memory architecture could be naturally extended to broader STEM domains like Physics. In such contexts, structured reasoning over scientific laws can synergize seamlessly with recent geometric and physics-informed neural architectures (Huang et al., 2026a).