A Neuro-Symbolic Framework for Geometric Problem Solving using Multimodal LLMs and Knowledge Graphs

Anupreksha Jain Nikhil Rayaprolu Sajid Ansari
IIIT Hyderabad
{anupreksha.j, nikhil.r, sajid.ansari}@research.iiit.ac.in

Abstract

This paper presents a novel neuro-symbolic framework for solving complex geometry problems by integrating the perceptual capabilities of Multimodal Large Language Models (MLLMs) with the structured reasoning of knowledge graphs (KGs). State-of-the-art MLLMs often struggle with the precise spatial and logical reasoning required for geometry. Our system addresses this by decoupling perception from reasoning. It employs a multi-stage pipeline where an MLLM (Google Gemini) first parses a geometric diagram and its text to generate a detailed, problem-specific knowledge graph. This graph is used alongside a persistent, global KG of geometric theorems. A reasoning module then queries these graphs to provide a rich, structured context to a solver LLM. Evaluated on the Geo3K dataset, our framework achieves an accuracy of 96% when combining the generated graph with the original image. This result significantly surpasses our own baseline system, which used unstructured semantic theorem retrieval and achieved 85.4% accuracy. This demonstrates that grounding LLM reasoning in an explicit, symbolic knowledge structure is superior to unstructured retrieval and dramatically enhances performance, interpretability, and reliability in complex domains like geometry.

Project related Code available at: https://github.com/anukvma/geometric_problem_solver

1 Introduction

The ability to integrate visual perception with abstract, logical reasoning is a cornerstone of human intelligence, yet it remains a significant challenge for artificial intelligence. The domain of geometry provides a clear benchmark for this capability, requiring an agent to understand a visual diagram, comprehend a textual problem statement, and execute a sequence of logical deductions based on a body of axiomatic knowledge.

While modern Multimodal Large Language Models (MLLMs) have demonstrated remarkable abilities, their performance on geometry problems is often inconsistent. Failures can stem from several sources:

- **Spatial Misinterpretation:** An MLLM might correctly identify a shape but fail to recognize crucial properties, such as a right angle denoted by a small square or congruent sides indicated by tick marks.
- **Logical Gaps:** The model may lack grounded knowledge of specific theorems required for a multi-step proof or misapply a known theorem.
- **Brittleness:** End-to-end models can be sensitive to minor variations in an image or its labeling, revealing a lack of robust, invariant understanding.

Our project confronts this semantic gap by evolving from a promising retrieval-based system to a more robust neuro-symbolic architecture. Initially, we developed a baseline system that augmented an LLM with a **Symbolic Theorem Retriever**. This approach used an MLLM to generate a textual description of a problem, which then served as a query to a FAISS index to retrieve relevant theorems. While achieving a respectable 85.4% accuracy, this method's reliance on unstructured text for both the query and the retrieved context limited its precision and reliability.

To overcome these limitations, we propose a novel framework to solve geometric problems using a formal graph representation. Our core contributions include: (1) The creation of a persistent and extensible **Theorem Knowledge Base**; (2) A **theorem retrieval system** that surfaces relevant axioms based on the problem's geometric entities; and (3) A complete neuro-symbolic **framework to solve geometric problem using graph representation** that first translates unstructured visual and textual data into a structured KG. This graph serves as a robust, explicit foundation for the reasoning stage, offering superior interpretability, modularity, and reliability. Our final results show a significant performance leap, with the KG-based system achieving **96% accuracy**, validating our core hypothesis that reasoning over formal symbolic structures is superior to reasoning over unstructured retrieved text.

2 Related Work

Our work is situated at the intersection of multimodal learning, knowledge representation, and automated reasoning.

End-to-End Multimodal Reasoning The dominant paradigm involves fine-tuning MLLMs on domain-specific datasets. **G-LLaVA** (Gao et al., 2023) exemplifies this, demonstrating strong performance on the Geo170K dataset. While effective, this end-to-end approach creates an opaque, entangled mapping from input to output, making it difficult to analyze the model's reasoning process or diagnose specific failure modes. Our framework explicitly decouples perception and reasoning for greater transparency.

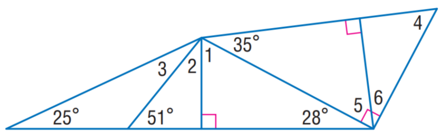
Vision-Centric Enhancements Recognizing that visual perception is a bottleneck in MLLMs, some research focuses on improving this initial step. The work of Zhang et al. (2024) argues for using specialized vision models to provide richer visual features to an LLM before reasoning. We extend this principle by not just providing latent features, but by prompting a state-of-the-art MLLM (Gemini) to externalize its perception into a structured, symbolic, and human-readable KG.

Knowledge Retrieval for Complex Reasoning Augmenting LLMs with retrieved knowledge is a standard technique for grounding and reducing hallucination. Our baseline system adapts this paradigm to geometry, where the knowledge base is a curated collection of formal theorems. This is similar to systems that retrieve passages from textbooks for science questions, but tailored to the axiomatic nature of mathematics.

Knowledge Graphs for Neuro-Symbolic AI The neuro-symbolic paradigm aims to combine the pattern recognition strengths of neural networks with the rigorous logic of symbolic systems. Formal KGs are a key component of this vision. Work on symbolic reasoners for text-based math problems (Wang et al., 2025) highlights the precision offered by structured knowledge. The primary challenge in applying such systems to multimodal problems is the **symbol grounding problem**: automatically and accurately creating the symbolic KG from unstructured visual data. Our project directly confronts this challenge, using a powerful MLLM as the "neuro" component for symbol grounding, which then enables a more "symbolic" reasoning process.

3 Dataset Details

We use the **Geo3K** dataset for all development and evaluation. It is a representative subset of the larger **Geo170K** benchmark (Gao et al., 2023) and contains approximately 3,000 problems, each presented as a triplet of (Image, Text, Answer Choices).



Text: Find the measure of angle 4.
Choices: A) 30, B) 45, C) 60, D) 90

Figure 1: An example data point from the Geo3K dataset.

Dataset Characteristics The problems cover a wide range of 2D Euclidean geometry topics, including properties of triangles, quadrilaterals, circles, parallel lines, similarity, and congruence.

Challenges The dataset is challenging because success often requires:

- **Implicit Information:** Not all properties are stated in the text. Right-angle symbols, congruence tick marks, and parallelism arrows in the diagram are crucial visual cues that must be correctly perceived.
- **Multi-Step Reasoning:** Most problems cannot be solved in a single step and require a chain of logical deductions.

- **Focus:** Diagrams can be complex, and the model must identify the relevant shapes and relationships while ignoring distractors.

4 System Architectures

This section details the design of both our initial baseline system and our final, more advanced framework, highlighting the evolution of our approach from unstructured retrieval to structured reasoning.

4.1 Baseline: Symbolic Theorem Retriever

Our initial system was a pragmatic pipeline designed to validate the core hypothesis that explicit knowledge grounding improves reasoning. It separated perception from reasoning, using unstructured text as the intermediate representation.

Phase 1: Knowledge Indexing (Offline)
Geo3K Problems $\rightarrow$ [MLLM] $\rightarrow$ Extract Theorems
Theorem Corpus $\rightarrow$ [Encoder] $\rightarrow$ FAISS Index

Phase 2: Augmented Inference (Online)
New Problem $\rightarrow$ [MLLM] $\rightarrow$ Text Description
Description $\rightarrow$ [FAISS Search] $\rightarrow$ Top-k Theorems
(Problem, Theorems) $\rightarrow$ [Solver LLM] $\rightarrow$ Solution

Figure 2: Architecture of the baseline theorem retrieval system.

Phase 1: Knowledge Extraction and Indexing This offline process created a searchable knowledge base. We used an MLLM to iterate through the Geo3K training set and extract natural language statements of the theorems required for each solution. This corpus of theorems was then encoded into 384-dimensional vectors using the ‘all-MiniLM-L6-v2‘ SentenceTransformer model. Finally, these vectors were indexed using FAISS’s ‘IndexFlatL2‘ to enable fast semantic similarity search.

Phase 2: Theorem-Augmented Inference At inference time, a new problem was fed to an MLLM to generate a rich, contextual ‘description‘ of the geometric setup. This description was then embedded and used as a query vector to search the FAISS index, retrieving the top-5 most relevant theorems. These retrieved theorem strings, along with the original image and problem text, were compiled into a prompt for a solver LLM, which then generated the final step-by-step answer.

Limitations While effective, this system’s main limitation was its reliance on unstructured text. Semantic similarity does not guarantee logical applicability, and reasoning over prose is less precise than reasoning over a formal graph.

4.2 Final System: Knowledge Graph Framework

To address the limitations of the baseline, our final system replaces the unstructured text intermediate with a formal, symbolic knowledge graph. The architecture is composed of three modular components.

Component 1: Knowledge Base Builder This offline component creates a centralized, reusable knowledge base of geometric theorems and shape properties. It processes a corpus of solved problems and uses an LLM to extract theorems into a structured format. This knowledge is stored in a Neo4j graph database. Furthermore, the framework includes a feedback loop to enrich this knowledge base; when a problem is solved using a novel application of a theorem or a new property, this information can be flagged for review and potential inclusion, allowing the system’s knowledge to grow over time.

- **Schema:** The KB consists of `:Theorem` nodes (with properties such as `name`, `description`, and `mathematical_form`) and `:Shape` nodes. An `:APPLICABLE_TO` relationship connects shapes to the theorems that can be applied to them.

```
// Nodes
(t:Theorem {
  name: string,
  description: string,
  conditions: list,
  mathematical_form: string
})
(s:Shape { name: string })

// Relationships
(s:Shape)-[:APPLICABLE_TO]->(t:Theorem)
```

Listing 1: Neo4j Schema for the Theorem Knowledge Base.

Component 2: Image-to-Graph Builder This is the core perception module, responsible for symbol grounding. Given a new problem, it uses the Gemini Vision API to parse the image and generate a detailed, problem-specific KG.

- **Extraction Prompt:** A carefully engineered prompt instructs the MLLM to act as a graph extractor and return a comprehensive JSON object

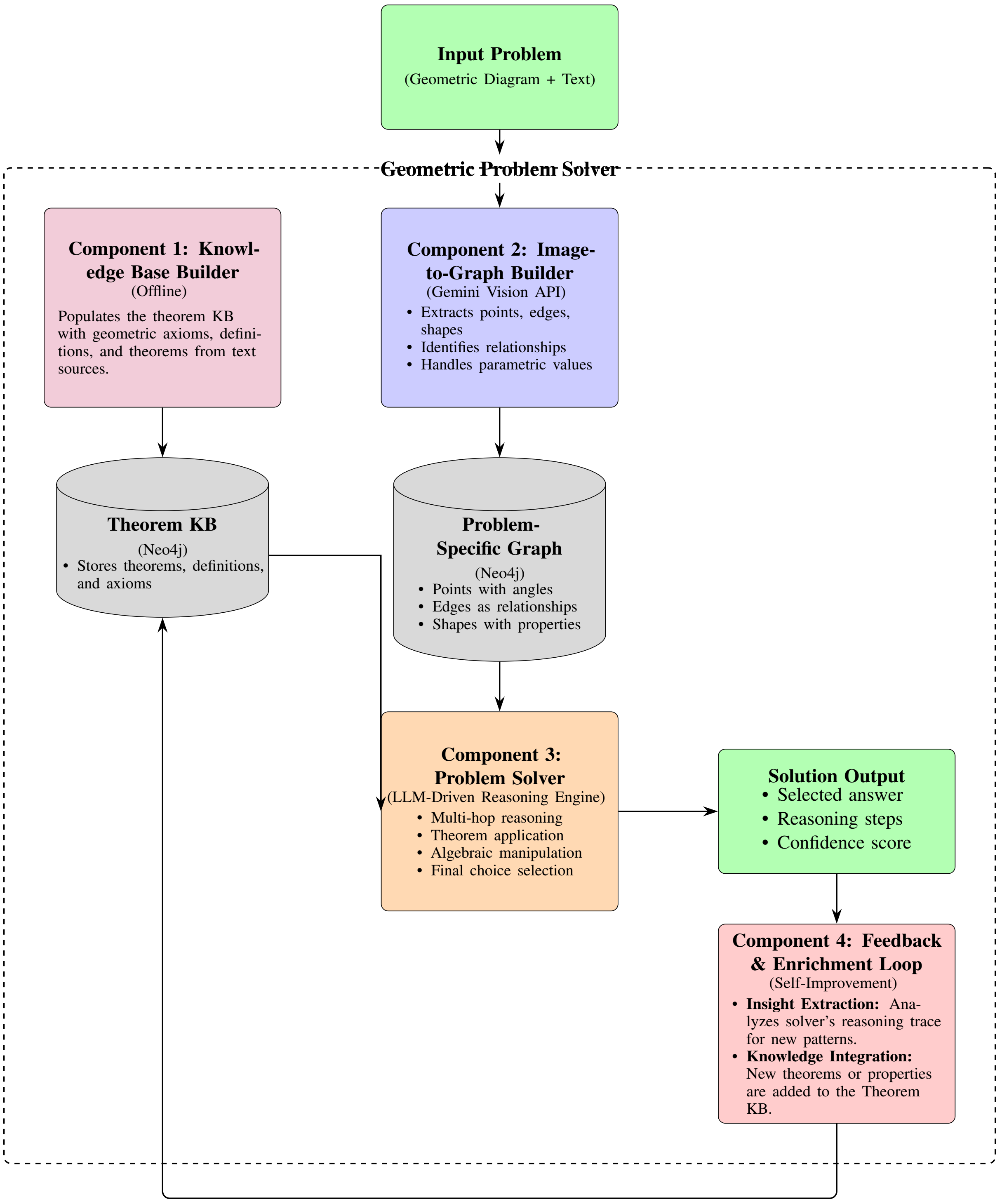

Input Problem
(Geometric Diagram + Text)
Geometric Problem Solver
Component 1: Knowledge Base Builder
(Offline)
Populates the theorem KB with geometric axioms, definitions, and theorems from text sources.
Component 2: Image-to-Graph Builder
(Gemini Vision API)
• Extracts points, edges, shapes
• Identifies relationships
• Handles parametric values
Theorem KB
(Neo4j)
• Stores theorems, definitions, and axioms
Problem-Specific Graph
(Neo4j)
• Points with angles
• Edges as relationships
• Shapes with properties
Component 3: Problem Solver
(LLM-Driven Reasoning Engine)
• Multi-hop reasoning
• Theorem application
• Algebraic manipulation
• Final choice selection
Solution Output
• Selected answer
• Reasoning steps
• Confidence score
Component 4: Feedback & Enrichment Loop
(Self-Improvement)
• Insight Extraction: Analyzes solver's reasoning trace for new patterns.
• Knowledge Integration: New theorems or properties are added to the Theorem KB.

Figure 3: Detailed architecture of the Geometric Problem Solver. The system has four main components. An offline component (1) builds a reusable theorem knowledge base. For each problem, an online component (2) uses the Gemini Vision API to convert the input into a problem-specific graph. The LLM-driven reasoning engine (3) utilizes both the theorem KB and the problem graph to derive the solution. Finally, a feedback and enrichment component (4) analyzes the solution trace to identify new insights, which are used to incrementally update the theorem KB, enabling the system to learn and improve over time.

detailing all geometric elements, their properties, and their relationships. This includes support for parametric expressions (e.g., ‘2x+5‘).

```
Analyze this geometric diagram and
    extract its
complete graph structure. Identify ALL
    points,
edges, shapes, angles, and their
    relationships.
```

```
Handle parametric values. Return a
    single, valid JSON.

{
  "points": [{"point_id": "A", ...}],
  "edges": [{"edge_id": "AB", "length":
    5.0, ...}],
  "shapes": [{"shape_id": "triangle_ABC
    ", ...}],
  "parameters": [{"parameter_name": "x",
     ...}]
}
```

Listing 2: Snippet of the MLLM prompt for graph extraction.

- **Graph Construction:** The extracted JSON is used to populate a new, isolated graph in Neo4j. All entities are namespaced with a unique `graph_id` (e.g., `problem_001_A`) to prevent conflicts. The schema represents points and shapes as nodes, while edges are modeled as `[:EDGE]` relationships between point nodes.

Component 3: KG-Augmented Problem Solver This component orchestrates the final reasoning process. It retrieves the problem-specific graph and queries the central Theorem KB for relevant theorems based on the shapes identified.

- **Comprehensive Context Prompt:** All of this structured information is compiled into a single prompt for the solver LLM. The prompt provides the problem text, the complete graph data serialized into a human-readable format, and the list of applicable theorems.

```
You are an expert geometry problem
    solver.

PROBLEM: {problem_text}
CHOICES: {choices}

COMPLETE KNOWLEDGE GRAPH:
SHAPES:
 - TRIANGLE (ID: prob_001_ABC)
   Properties: type=right
POINTS:
 - Point A: Angles at vertex: BAC=90
EDGES:
 - Edge AB: length=3.0

AVAILABLE THEOREMS:
1. Pythagorean Theorem: a^2 + b^2 = c^2

TASK:
1. Analyze the knowledge graph.
2. Use multi-hop reasoning and theorems
    to solve.
3. Return a JSON with your reasoning
    chain and
   the final answer.
```

Listing 3: High-level structure of the solver prompt.

This structured approach forces the LLM to ground its reasoning in the provided facts, enabling a transparent, multi-hop deductive process from knowns to unknowns.

Component 4: Continuous Feedback and Knowledge Enrichment Loop This component ensures that the system's intelligence evolves over time through self-improvement. After each problem-solving session, the solver's reasoning trace and final solution are analyzed to identify potential new insights—such as a novel theorem application, a newly inferred geometric property, or a corrected misconception.

- **Insight Extraction:** When the solver produces an explanation or derivation, an LLM-based analyzer reviews it to detect patterns not explicitly covered in the existing knowledge base. For example, if a previously unseen relationship between diagonals and angles emerges, it is flagged as a potential new theorem or corollary.

- **Incremental Knowledge Integration:** The new information is automatically converted into structured Neo4j entries — typically as new `:Theorem` nodes, updated `:APPLICABLE_TO` links, or additional properties on existing nodes. The schema version is incremented to ensure backward compatibility.

5 Experiments and Evaluation

5.1 Experimental Setup

We evaluated our systems on a random sample of 50 problems from the Geo3K test set. Our primary model was Google's Gemini 2.5 Flash, with Gemma 3-27b used for comparison.

Evaluated Configurations:

1. **Baseline (Theorem Retrieval):** Our mid-semester system using FAISS-based retrieval.

2. **Final KG System (Graph Only):** The final system where the solver LLM receives only the generated knowledge graph and theorems.

3. **Final KG System (Graph + Image):** The final system where the solver also receives the original problem image.

Metrics: We measure performance using success rate (percentage of error-free solves) and accuracy (percentage of correct answers among successful solves).

5.2 Quantitative Analysis

The results show a clear progression in performance, with the final KG-based system significantly outperforming the baseline. Table 1 details the extensive ablation study conducted on our baseline system, which established the importance of both visual context and theorem retrieval. Table 2 compares the best baseline configuration against our final KG-based system.

Baseline Ablation Configuration	Acc. (%)
Full Pipeline (Image + Text + Thm)	**85.42**
Image + Text (No Theorems)	81.25
Text + Theorems (No Image)	42.00
Text Only (No Image/Theorems)	40.00

Table 1: Ablation study results for the baseline theorem retrieval system on 50 Geo3K problems, demonstrating the critical role of both the image and retrieved theorems.

System Configuration	Model	Success (%)	Acc. (%)
Baseline (Thm Retrieval)	Gemini	98	85.4
KG System (G-Only)	Gemini	94	90.0
KG System (G+I)	**Gemini**	**98**	**96.0**

Table 2: Final performance comparison on 50 Geo3K problems. The final KG-based system ("G+I") significantly outperforms the unstructured theorem retrieval baseline.

6 Analysis of Results

6.1 Insights from the Baseline System

The ablation study on our initial retrieval-based system (Table 1) provided crucial insights. The full pipeline achieved 85.42% accuracy. Removing the retrieved theorems caused a 4.17 point drop, confirming that explicit knowledge grounding is beneficial. The most dramatic result was the performance collapse to 40% when the image was removed. This underscored that geometry is fundamentally a multimodal domain where visual grounding is non-negotiable. Textual descriptions, no matter how detailed, cannot fully replace the rich spatial information of a diagram.

6.2 Superiority of the KG Framework

The most critical finding is the performance leap from our baseline to the final system (Table 2). The KG-based approach with Gemini (Graph + Image) achieved **96% accuracy**, a significant **10.6 percentage point improvement** over the 85.4% accuracy of the theorem retrieval baseline. This result strongly validates our hypothesis: reasoning over a structured, explicit knowledge graph is far more effective than reasoning over a semantically retrieved but unstructured set of text snippets. The graph provides a precise, unambiguous representation of the problem's entities and relationships, eliminating the ambiguity inherent in natural language and leading to more reliable deductions.

The graph's superiority stems from its ability to facilitate structured, multi-step reasoning. An LLM agent cannot iteratively query a static image for new information based on intermediate conclusions. It performs a single perception pass to extract features, and its subsequent reasoning is confined to that initial extraction. In contrast, the KG is a dynamic reasoning space. The LLM can treat a complex problem as a sequence of smaller sub-problems, performing targeted queries against the graph at each step to retrieve the precise information needed to advance the solution. This process of dividing the problem and methodically querying for relevant facts is analogous to human logical deduction and is not possible when reasoning over a flat, unstructured representation like an image or its textual description.

6.3 Synergy of Graph and Image

Within the final KG framework, the highest accuracy (96%) was achieved in the "Graph + Image" mode. This configuration outperforms the "Graph Only" mode (90% accuracy), suggesting that while the generated graph captures the vast majority of necessary information, the raw image provides subtle but important contextual cues that help the LLM resolve ambiguities or ground its reasoning more effectively.

6.4 Qualitative Case Study: KG-based Reasoning

To illustrate the precision of the KG system, consider a problem involving a parallelogram PQRS with PQ=5. The goal is to find the length of RS.

1. **Image-to-Graph Generation:** The perception module generates a KG containing nodes for the shape (‘id: PQRS, type: parallelogram‘), its edges (‘id: PQ, length: 5‘; ‘id: RS, length: unknown‘), and their relationships (‘PQ -[:IS_OPPOSITE_TO]-> RS‘).

2. **Theorem Attachment:** The system queries the theorem KB and attaches relevant theorems, including ‘T1: "Opposite sides of a parallelogram are equal."‘ to the ‘PQRS‘ node.

3. **KG-Augmented Reasoning:** The solver LLM receives this graph. Its reasoning becomes a series of deterministic queries:

    - **Goal:** Find ‘length‘ of ‘RS‘.
    - **Context:** ‘RS‘ is an edge of ‘parallelogram PQRS‘.
    - **Logic:** Apply theorem ‘T1‘ to ‘PQRS‘.
    - **Execution:** Theorem ‘T1‘ implies ‘length(RS) = length(side$_{o}pposite_{t}o_{R}S$)‘.
    - **Query:** Find node connected to ‘RS‘ by ‘IS_OPPOSITE_TO‘. Result: ‘PQ‘.
    - **Lookup:** Get ‘length‘ of ‘PQ‘. Result: 5.
    - **Conclusion:** The length of ‘RS‘ is 5.

This transparent, verifiable process, grounded in the KG, is a key advantage over both end-to-end models and our unstructured retrieval baseline.

6.5 Failure Analysis

Most failures across all systems originate from the perception stage. In one failed case involving overlapping triangles, the MLLM failed to correctly identify all vertices of the smaller, nested triangle in the generated graph. This incorrect KG made downstream reasoning impossible, as the solver was operating on a flawed representation of the world. This highlights that the quality of the symbolic grounding is the primary bottleneck of the system.

7 Limitations and Future Work

Limitations

1. **Perception as a Bottleneck:** The system's accuracy is fundamentally capped by the MLLM's ability to correctly parse the image into a graph. Complex, cluttered, or ambiguous diagrams remain a significant challenge.

2. **Static Theorem Base:** The theorem knowledge base is built offline. While it can learn new theorems from solved examples, it cannot derive new geometric truths from first principles during an unsolved problem.

3. **Implicit Algebraic Reasoning:** The system relies on the LLM's intrinsic ability to perform algebraic manipulation and apply logical rules. This reasoning is not fully formalized within the graph structure itself.

Future Work

1. **Hybrid Vision Models:** To improve perception, we could combine the general-purpose MLLM with specialized computer vision models fine-tuned to detect specific geometric primitives (e.g., right-angle markers, parallel line indicators), creating a more robust graph generation pipeline by using the specialized detectors to verify or correct the MLLM's output.

2. **Formal Proof Generation:** The current system provides a reasoning chain but not a formal, machine-verifiable proof. Future work could focus on translating the LLM's reasoning into a formal language like Coq or Lean, enabling automated verification of the solution's correctness.

3. **Interactive Reasoning:** An interactive system could allow for human-in-the-loop correction. If the initial graph generation is flawed, a user could correct it, allowing the reasoning module to proceed with accurate information. This would be invaluable for both error correction and data collection for fine-tuning the perception model.

8 Conclusion

We have presented a neuro-symbolic framework that effectively solves geometry problems by creating and reasoning over knowledge graphs. By evolving from a promising but limited unstructured retrieval baseline to a formal KG-based system, we demonstrated a significant increase in accuracy from 85.4% to 96%. This journey validates our core hypothesis: grounding LLM reasoning in explicit, symbolic structures is superior to relying on unstructured semantic similarity. Our key findings are threefold: (1) Modern MLLMs can act as powerful "symbol grounders," translating visual information into structured KGs. (2) The precision of a formal graph enables more reliable and accurate multi-hop reasoning. (3) The synergy of providing both the structured graph and the original image as context yields the best results, highlighting the complementary strengths of symbolic and perceptual data. This work represents a promising step towards building more robust, reliable, and interpretable AI systems for complex, multi-modal reasoning domains.

References

Jiahui Gao, Renjie Pi, Jipeng Zhang, Jiacheng Ye, Wanjun Zhong, Yufei Wang, Lanqing Hong, Jianhua Han, Hang Xu, Zhenguo Li, and Lingpeng Kong. 2023. G-llava: Solving geometric problem with multi-modal large language model. *Preprint*, arXiv:2312.11370.

Ying Wang, Wei Zhou, Yongsheng Rao, and Hao Guan. 2025. A knowledge and semantic fusion method for automatic geometry problem understanding. *Applied Sciences*, 15(7):3857.

Shan Zhang, Aotian Chen, Yanpeng Sun, Jindong Gu, Piotr Koniusz, Kai Zou, Anton van den Hengel, Yi-Yu Zheng, and Yuan Xue. 2024. Open eyes, then reason: Fine-grained visual mathematical understanding in mllms. *Preprint*, arXiv:2501.06430.

A Example Problem and Solution

This appendix provides a detailed walkthrough of how the system solves a parametric problem from the Geo3K dataset (ID: 109).

A.1 Problem Statement

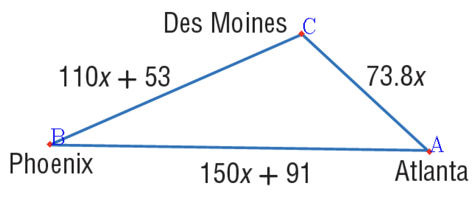


Figure 4: Diagram for Problem ID 109.

Question: "A plane travels from Des Moines to Phoenix, on to Atlanta, and back to Des Moines, as shown below. Find the distance in miles from Des Moines to Phoenix if the total trip was 3482 miles."

A.2 Extracted Knowledge Graph

The Image-to-Graph Builder processes the image and text, producing the following JSON representation, which is then loaded into the problem-specific Neo4j graph.

```
{
  "shapes_by_type": {
    "triangle": [
      {
        "shape_id": "109_triangle_PDA",
        "shape_type": "triangle",
        "properties": {
          "triangle_type": "scalene",
          "side_lengths": {
            "PD": "110*x + 53",
            "DA": "73.8*x",
            "PA": "150*x + 91"
          }
        }
      }
    ]
  },
  "points": [
    { "point_id": "109_A", "label": "
    Atlanta" },
    { "point_id": "109_D", "label": "Des
     Moines" },
    { "point_id": "109_P", "label": "
    Phoenix" }
  ],
  "edges": [
    {
      "edge_id": "109_DA",
      "length_expression": "73.8*x",
      "start_point": "109_D",
      "end_point": "109_A"
    },
    {
      "edge_id": "109_PA",
      "length_expression": "150*x + 91",
      "start_point": "109_P",
      "end_point": "109_A"
    },
    {
      "edge_id": "109_PD",
      "length_expression": "110*x + 53",
      "start_point": "109_P",
      "end_point": "109_D"
    }
  ],
  "has_parameters": true,
  "parameters": [{"parameter_name": "x
    "}]
}
```

Listing 4: JSON output from the perception module.

A.3 Prompt to Solver LLM

The extracted graph data and relevant theorems are formatted into a comprehensive prompt for the reasoning engine.

```
You are an expert geometry problem
    solver. You have been given a
    complete knowledge graph of a
    geometric figure and all applicable
    theorems. Use multi-hop reasoning to
     solve the problem.
PROBLEM TO SOLVE:
A plane travels from Des Moines to
    Phoenix, on to Atlanta, and back to
    Des Moines, as shown below. Find the
     distance in miles from Des Moines
    to Phoenix if the total trip was
    3482 miles.
ANSWER CHOICES:
A. 73.8
B. 91
C. 110
D. 1153
**IMPORTANT: After solving, you MUST
    select the correct answer choice
    from the options above.**
COMPLETE KNOWLEDGE GRAPH:
PARAMETERS:
  - Parameter 'x': parameter
    representing variable length
    Appears in: PD, DA, PA
    Constraints: x > 0
GEOMETRIC SHAPES:
  - TRIANGLE (ID: 109_triangle_PDA,
    Label:   PDA  )
    Properties: triangle_type=scalene,
    side_lengths={'PD': '110*x + 53', '
    DA': '73.8*x', 'PA': '150*x + 91'},
    angles={}, special_properties=[]
    Confidence: 0.85
POINTS:
  - Point 109_A: coordinates (0.85,
    0.75) - Label: Atlanta
  - Point 109_D: coordinates (0.50,
    0.25) - Label: Des Moines
  - Point 109_P: coordinates (0.15,
    0.75) - Label: Phoenix
EDGES:
  - Edge DA (109_D -> 109_A): length
    =73.8*x (parametric), type=segment
  - Edge PA (109_P -> 109_A): length
    =150*x + 91 (parametric), type=
    segment
  - Edge PD (109_P -> 109_D): length
    =110*x + 53 (parametric), type=
    segment
AVAILABLE THEOREMS AND PROPERTIES:
1. Definition of Isosceles Triangle
    Sides
   ID: 7b29a2129110
   Description: An isosceles triangle is
     a triangle that has at least two
    sides of equal length...
... (other theorems omitted for brevity)
     ...
BASIC GEOMETRIC PROPERTIES:
- Triangle perimeter: P = sum of all
    sides
... (other properties omitted for
    brevity) ...
TASK:
1. Analyze the knowledge graph to
    understand the geometric figure
2. Identify what information is given (
    numeric or parametric) and what
    needs to be found
3. Use multi-hop reasoning to connect
    known values to unknown values
... (other instructions omitted for
    brevity) ...
```

Listing 5: Full prompt provided to the solver LLM.

A.4 Generated Reasoning Chain

The solver LLM returns a structured JSON output containing the following step-by-step reasoning.

1. **Step 1: Set up Perimeter Equation.** The model identifies that the "total trip" corresponds to the perimeter of the triangle PDA. It applies the "Triangle perimeter" property.

   - **Property Used:** Triangle perimeter: P = sum of all sides.
   - **Inputs:** Total trip = 3482, PD = ‘110*x + 53‘, PA = ‘150*x + 91‘, DA = ‘73.8*x‘.
   - **Work:** Total Trip = PD + PA + DA.
   - **Result:** ‘3482 = (110*x + 53) + (150*x + 91) + (73.8*x)‘

2. **Step 2: Solve for Parameter ’x’.** The model simplifies the algebraic equation from Step 1 to find the value of x.

   - **Property Used:** Solving a linear equation.
   - **Inputs:** The equation from Step 1.
   - **Work:**

     ```
     3482 = (110 + 150 + 73.8)*x + (53 + 91)
     3482 = 333.8*x + 144
     3338 = 333.8*x
     x = 3338 / 333.8
     ```

   - **Result:** ‘x = 10‘

3. **Step 3: Calculate Target Distance.** The model substitutes the value of ‘x‘ back into the expression for the distance from Des Moines to Phoenix (PD).

   - **Property Used:** Parameter substitution.
   - **Inputs:** ‘x = 10‘, Expression for PD = ‘110*x + 53‘.
   - **Work:**

     ```
     PD = 110*(10) + 53
     PD = 1100 + 53
     PD = 1153
     ```

   - **Result:** ‘PD = 1153‘

A.5 Final Answer Selection

The calculated distance for PD is 1153 miles. The model matches this result with the provided answer choices and selects the correct option.

- **Final Answer:** 1153 miles
- **Selected Choice:** D