A Neuro-Symbolic Framework for Geometric Problem Solving

Using Multimodal LLMs and Knowledge Graphs

Anupreksha Jain Nikhil Rayaprolu Sajid Ansari

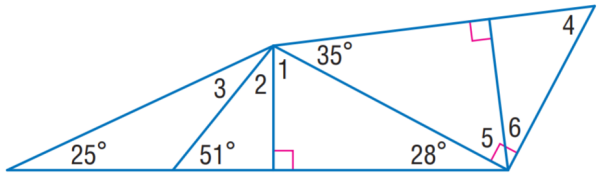
IIIT Hyderabad

Final Project Evaluation

Outline

Why is Geometry Hard for AI?

Geometry requires a unique blend of perception and logic. MLLMs often fail due to:



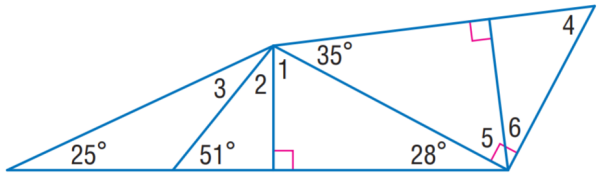
Problem: Find the measure of angle 4.

Choices: A) 30, B) 45, C) 60, D) 90

Why is Geometry Hard for AI?

Geometry requires a unique blend of perception and logic. MLLMs often fail due to:

Spatial Misinterpretation: Missing subtle visual cues like right-angle markers or congruence ticks.



Problem: Find the measure of angle 4.

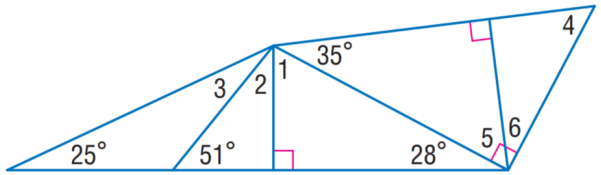
Choices: A) 30, B) 45, C) 60, D) 90

Why is Geometry Hard for AI?

Geometry requires a unique blend of perception and logic. MLLMs often fail due to:

Spatial Misinterpretation: Missing subtle visual cues like right-angle markers or congruence ticks.

Logical Gaps: Lacking or misapplying the specific geometric theorems needed for a proof.



Problem: Find the measure of angle 4.

Choices: A) 30, B) 45, C) 60, D) 90

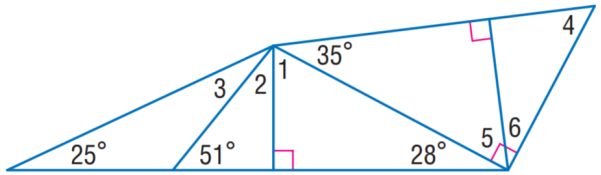
Why is Geometry Hard for AI?

Geometry requires a unique blend of perception and logic. MLLMs often fail due to:

Spatial Misinterpretation: Missing subtle visual cues like right-angle markers or congruence ticks.

Logical Gaps: Lacking or misapplying the specific geometric theorems needed for a proof.

Lack of Grounding: Reasoning is not grounded in an explicit, verifiable knowledge structure.



Problem: Find the measure of angle 4.

Choices: A) 30, B) 45, C) 60, D) 90

Why is Geometry Hard for AI?

Our Goal

To build a system that decouples perception from reasoning for more robust and interpretable problem-solving.

Our Journey: Two Architectures

Our project evolved through two main stages

Approach 1: Baseline

Our Journey: Two Architectures

Our project evolved through two main stages

Approach 1: Baseline

Symbolic Theorem Retriever

Used an MLLM to describe the problem in text.

Searched a vector database (FAISS) for relevant theorems.

Fed unstructured text (description + theorems) to a solver.

Result: 85.4% Accuracy

Approach 2: Final System

Our Journey: Two Architectures

Our project evolved through two main stages

Approach 1: Baseline

Symbolic Theorem Retriever

Used an MLLM to describe the problem in text.

Searched a vector database (FAISS) for relevant theorems.

Fed unstructured text (description + theorems) to a solver.

Result: 85.4% Accuracy

Approach 2: Final System

Knowledge Graph Framework

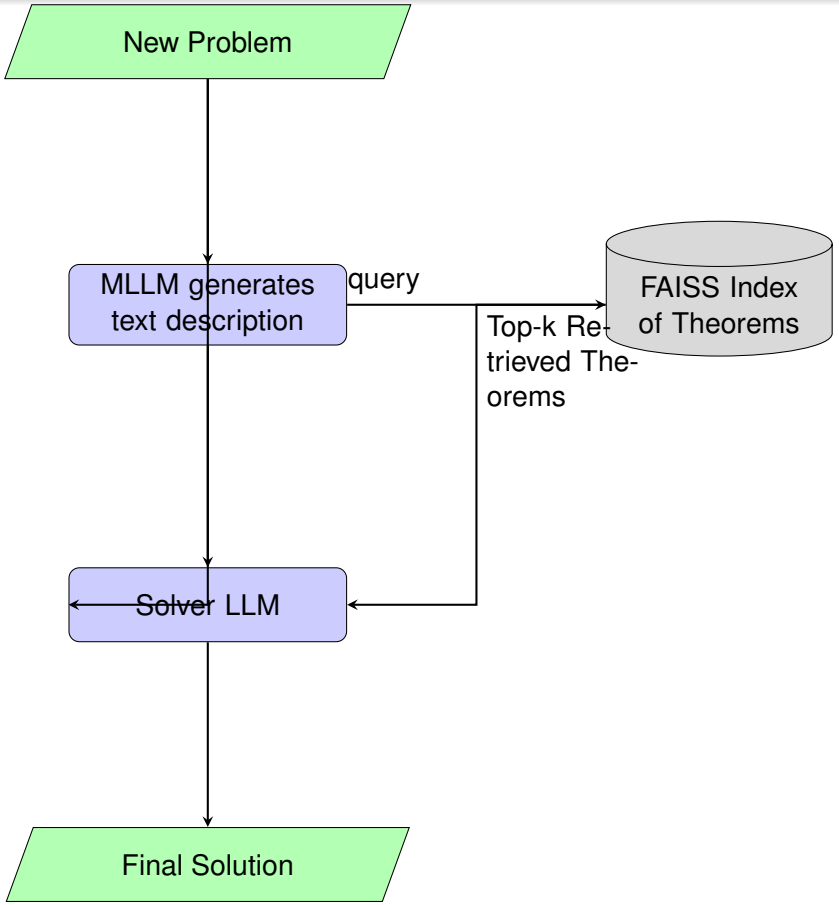
Uses an MLLM to parse the image into a **structured graph**.

Reasons over this formal, symbolic representation.

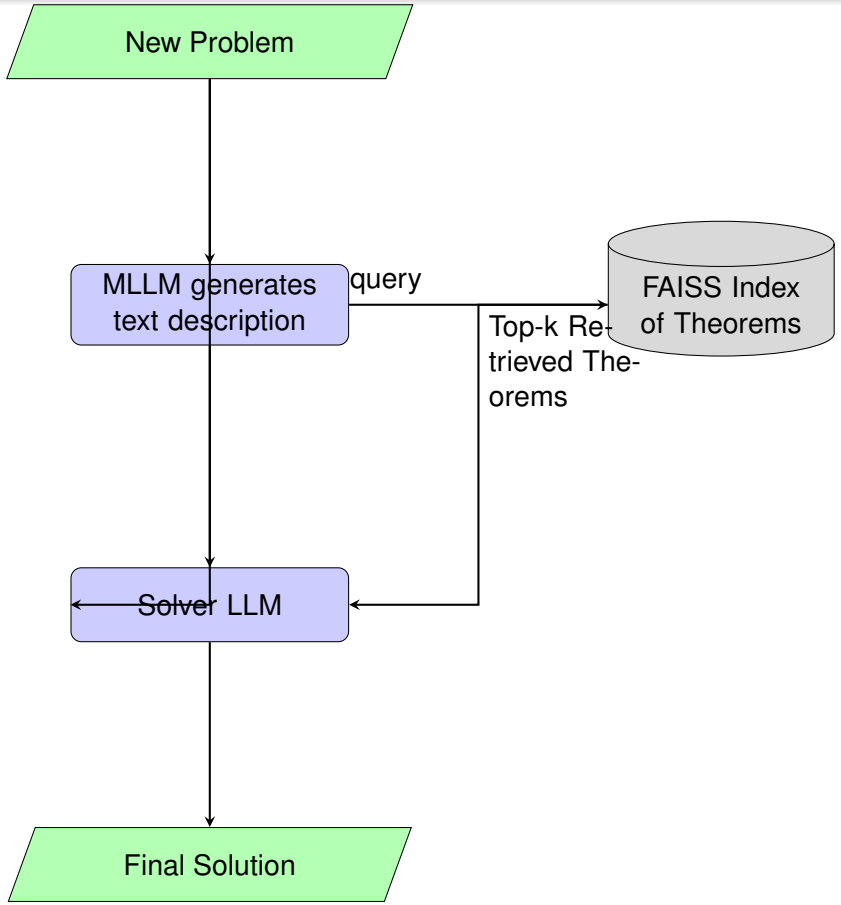
Provides explicit, verifiable context to the solver.

Result: 96.0% Accuracy

How our Initial System Worked



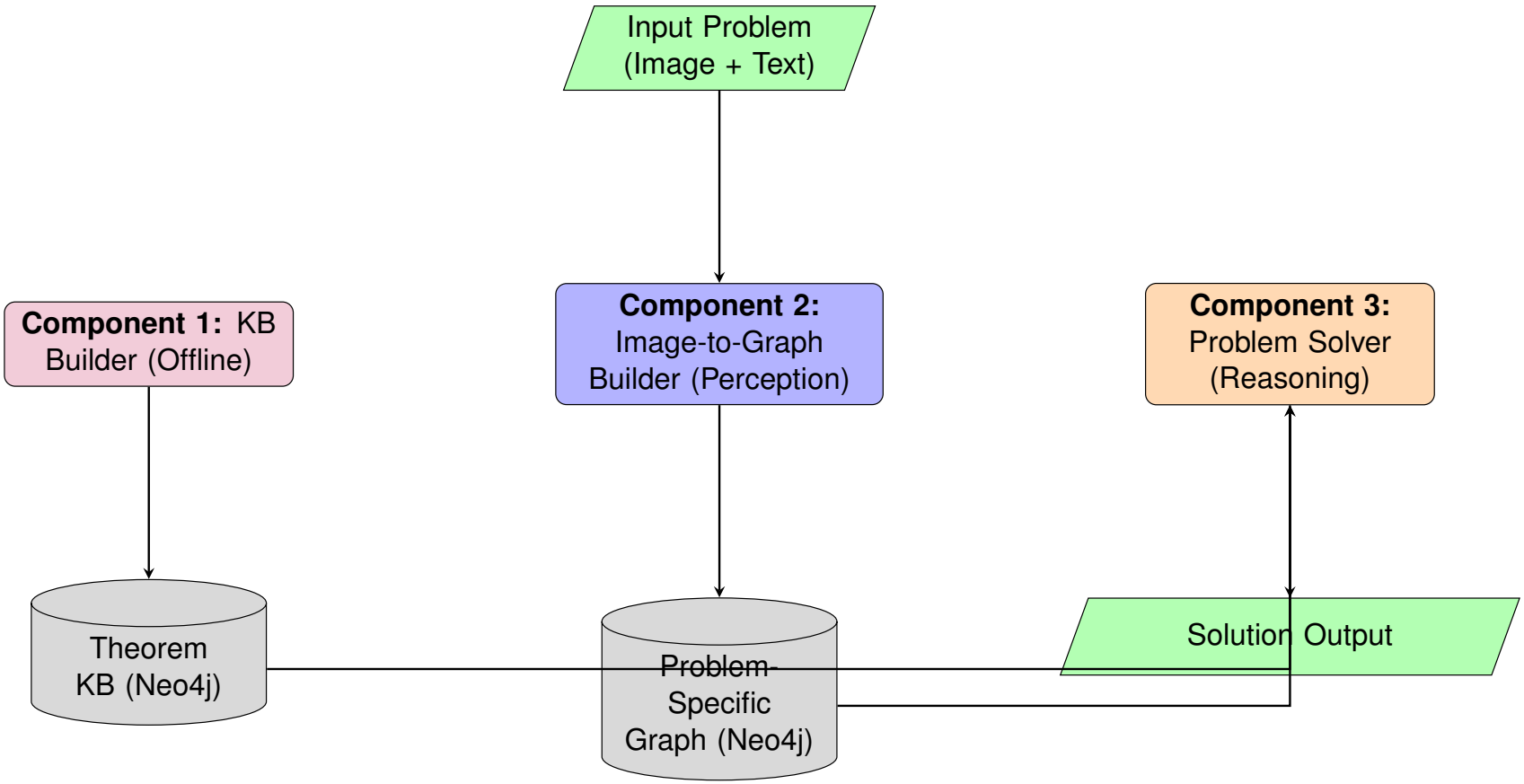
How our Initial System Worked



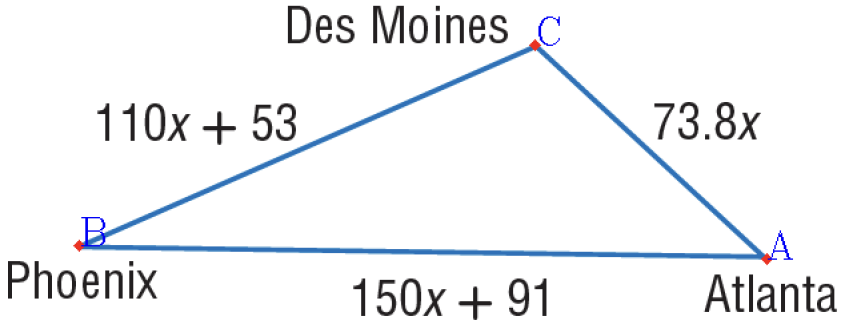
Key Limitation

Reasoning over unstructured text is imprecise. Semantic similarity from FAISS doesn't guarantee logical applicability, leading to errors.

Final Architecture: A Neuro-Symbolic Pipeline



Problem Statement



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."*

The system perceives the image and text, then builds a structured graph.

The MLLM extracts points, shapes, and parametric expressions into a JSON format.

```
{
  "shapes_by_type": {
    "triangle": [{
        "shape_id": "109_triangle_PDA",
        "properties": {
          "side_lengths": {
            "PD": "110*x + 53",
            "DA": "73.8*x",
            "PA": "150*x + 91"
          }
        }
      }]
  },
  "parameters": [{"parameter_name": "x"}]
}
```

Listing 1: Key excerpts from the generated JSON.

The generated graph is formatted into a rich prompt for the solver.

This provides all necessary context, grounding the LLM's reasoning process.

```
You are an expert geometry solver.
PROBLEM TO SOLVE:
Find the distance from Des Moines to Phoenix if the
    total trip was 3482 miles.
...
COMPLETE KNOWLEDGE GRAPH:
PARAMETERS:
  - Parameter 'x': parameter representing variable
    length
GEOMETRIC SHAPES:
  - TRIANGLE (ID: 109_triangle_PDA)
    Properties: side_lengths={'PD': '110*x + 53', 'DA':
    '73.8*x', 'PA': '150*x + 91'}
...
AVAILABLE THEOREMS AND PROPERTIES:
- Triangle perimeter: P = sum of all sides
...
```

The solver uses the prompt to generate a step-by-step solution.

Step 1: Set up Perimeter Equation

The model identifies "total trip" as the triangle's perimeter.

The solver uses the prompt to generate a step-by-step solution.

Step 1: Set up Perimeter Equation

The model identifies "total trip" as the triangle's perimeter.

Work: Total Trip = PD + PA + DA

Result: '3482 = (110*x + 53) + (150*x + 91) + (73.8*x)'

The solver uses the prompt to generate a step-by-step solution.

Step 1: Set up Perimeter Equation

The model identifies "total trip" as the triangle's perimeter.

Work: Total Trip = PD + PA + DA

Result: '3482 = (110*x + 53) + (150*x + 91) + (73.8*x)'

Step 2: Solve for Parameter 'x'

It simplifies the equation to find the value of 'x'.

The solver uses the prompt to generate a step-by-step solution.

Step 1: Set up Perimeter Equation

The model identifies "total trip" as the triangle's perimeter.

Work: Total Trip = PD + PA + DA

Result: '3482 = (110*x + 53) + (150*x + 91) + (73.8*x)'

Step 2: Solve for Parameter 'x'

It simplifies the equation to find the value of 'x'.

Work: '3482 = 333.8*x + 144 $\implies$ $3338 = 333.8 * x$'

Result: 'x = 10'

The reasoning chain culminates in the final answer.

Step 3: Calculate Target Distance

The value of ‘x‘ is substituted back into the expression for the distance from Des Moines to Phoenix (PD).

Inputs: ‘x = 10‘, Expression for PD = ‘110*x + 53‘.

Work: ‘PD = 110*(10) + 53 $\implies$ $PD = 1100 + 53$‘

Result: ‘PD = 1153‘

The reasoning chain culminates in the final answer.

Step 3: Calculate Target Distance

The value of ‘x‘ is substituted back into the expression for the distance from Des Moines to Phoenix (PD).

Inputs: ‘x = 10‘, Expression for PD = ‘110*x + 53‘.

Work: ‘PD = 110*(10) + 53 $\implies$ $PD = 1100 + 53$‘

Result: ‘PD = 1153‘

Final Answer: **1153** miles (Choice D)

Experimental Setup

Dataset: A random sample of 50 problems from the **Geo3K** test set.

Primary Model: Google **Gemini 2.5 Flash**.

Metrics:

Success Rate: Percentage of problems solved without errors.

Accuracy: Percentage of correct final answers among successful solves.

Configurations Tested:

Baseline System (Unstructured Theorem Retrieval)

Final KG System (Graph Data Only)

Final KG System (Graph Data + Original Image)

Performance Leap: From Retrieval to KG-based Reasoning

Table: Final Performance Comparison on 50 Geo3K Problems

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

Performance Leap: From Retrieval to KG-based Reasoning

Table: Final Performance Comparison on 50 Geo3K Problems

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

A **10.6 percentage point improvement** in accuracy validates our hypothesis: reasoning over structured graphs is superior to reasoning over unstructured text.

Case Study: Solving a Parallelogram Problem (Part 1)

Problem: Find the length of RS in parallelogram PQRS, given PQ = 5.

1. Graph Generation (Perception): The MLLM parses the image and creates a graph:

Node: ‘shape_PQRS‘ (type: parallelogram)
Node: ‘edge_PQ‘ (length: 5)
Node: ‘edge_RS‘ (length: unknown)
Relationship: ‘edge_PQ‘ -[:IS_OPPOSITE_TO]-> ‘edge$_R S$‘

Case Study: Solving a Parallelogram Problem (Part 1)

Problem: Find the length of RS in parallelogram PQRS, given PQ = 5.

1. Graph Generation (Perception): The MLLM parses the image and creates a graph:

Node: ‘shape_PQRS‘ (type: parallelogram)
Node: ‘edge_PQ‘ (length: 5)
Node: ‘edge_RS‘ (length: unknown)
Relationship: ‘edge_PQ‘ -[:IS_OPPOSITE_TO]-> ‘edge$_R S$‘

2. Theorem Attachment: The system queries the Theorem KB and attaches a relevant rule:

"Opposite sides of a parallelogram are equal."

Case Study: Solving a Parallelogram Problem (Part 2)

Problem (continued): Use the attached theorem to find the length of RS.

3. Logical Deduction (Reasoning): The solver LLM executes a verifiable reasoning chain:

Goal: Find length of ‘edge_RS‘.
Query Graph: Find what is opposite to ‘RS‘. Result: ‘PQ‘.
Apply Theorem: Therefore, ‘length(RS) = length(PQ)‘.
Lookup in Graph: ‘length(PQ)‘ is 5.
Conclusion: The length of RS is **5**.

Where Does the System Fail?

The Bottleneck is Perception

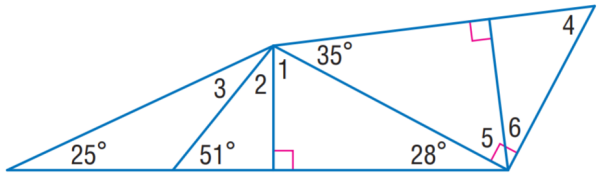
Nearly all failures originate in the **Image-to-Graph generation** stage.

Example failure:

A complex diagram with overlapping triangles.

The MLLM failed to correctly identify all vertices of a smaller, nested triangle.

It generated an incomplete and incorrect knowledge graph.



(Illustrative diagram with overlapping shapes)

Conclusion

If the initial symbolic grounding is flawed, all subsequent reasoning will be incorrect. Improving perception is the key to higher accuracy.

Conclusion

We successfully developed a neuro-symbolic framework that significantly improves upon baseline methods for geometric problem solving.

Key Achievements

Achieved **96% accuracy** on the Geo3K dataset, a **10.6%** improvement over our retrieval-based baseline.

Conclusion

We successfully developed a neuro-symbolic framework that significantly improves upon baseline methods for geometric problem solving.

Key Achievements

Achieved **96% accuracy** on the Geo3K dataset, a **10.6%** improvement over our retrieval-based baseline.

Validated that grounding LLM reasoning in an explicit **Knowledge Graph** leads to more reliable and interpretable results.

Conclusion

We successfully developed a neuro-symbolic framework that significantly improves upon baseline methods for geometric problem solving.

Key Achievements

Achieved **96% accuracy** on the Geo3K dataset, a **10.6%** improvement over our retrieval-based baseline.

Validated that grounding LLM reasoning in an explicit **Knowledge Graph** leads to more reliable and interpretable results.

Showcased that modern MLLMs can act as powerful **symbol grounders**, converting visual data into structured representations.

Conclusion

We successfully developed a neuro-symbolic framework that significantly improves upon baseline methods for geometric problem solving.

Key Achievements

- Achieved **96% accuracy** on the Geo3K dataset, a **10.6%** improvement over our retrieval-based baseline.
- Validated that grounding LLM reasoning in an explicit **Knowledge Graph** leads to more reliable and interpretable results.
- Showcased that modern MLLMs can act as powerful **symbol grounders**, converting visual data into structured representations.
- Demonstrated the synergy of providing both the structured graph and the raw image, which yielded the best performance.

Future Work

Improving Perception

Hybrid Vision Models: Combine the MLLM with specialized detectors for geometric primitives (e.g., right-angle markers).

Interactive Correction: Allow a human-in-the-loop to fix errors in the generated graph.

Enhancing Reasoning

Formal Proof Generation: Translate the LLM's reasoning steps into a machine-verifiable language like Coq or Lean.

Dynamic Theorem Derivation: Enable the system to derive new geometric truths from first principles.

Thank You

Questions?

Project Code: `https://github.com/anukvma/geometric_problem_solver`