Seeing and Solving: An Interpreter-Solver Framework for Geometric Reasoning with Large Vision and Language Models

Anonymous ACL submission

Abstract

Geometrical Problem Solving (GPS), which involves interpreting diagrams and text to solve problems using logical reasoning and mathematical principles, has gained significant attention with the advancement of Multimodal Large Language Models (MLLMs). However, solving these problems in a zero-shot setting has received comparatively little attention, despite the growing improvements in AI reasoning for visual mathematics understanding. In this study, we propose Interpreter-Solver, a two-stage pipeline that seamlessly integrates Vision Language Models (VLMs) and Large Language Models (LLMs) to address these issues. Our approach harnesses the VLM's visual understanding to extract formal textual descriptions of geometric relationships, which are then processed by the LLM for its outstanding reasoning capabilities. This entire process employs a zero-shot prompting strategy to resolve the previous challenges. Without any fine-tuning, it establishes itself as a new state-of-the-art by achieving accuracies of 83.19% on the Geometry3K dataset and 69.67% on the MathVerse dataset. It surpasses leading methods like InterGPS, GeoDRL, and AutoGPS while requiring 5× and 2.8× fewer parameters than the top models on these benchmarks. You can find all the codes, data, and reasoning files here `https://anonymous.4open.science/r/Interpreter-Solver/`.

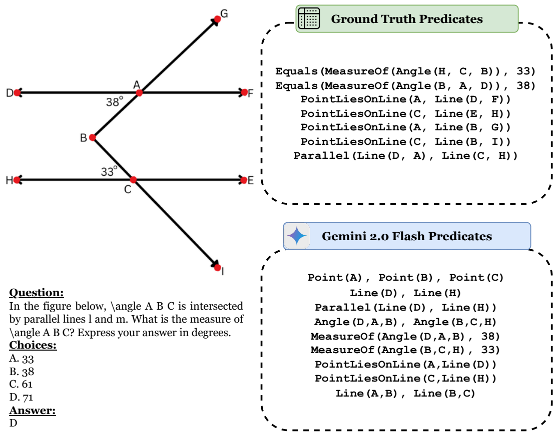

Figure 1: Data example of a geometry problem annotated with formal language predicates for both ground truth and Gemini 2.0 Flash-generated predicates.

1 Introduction

Solving geometric problems from diagrams and natural language text remains a challenging task at the intersection of visual understanding and symbolic reasoning. Previous studies have underscored the superiority of neuro-symbolic frameworks, wherein transformer-based models such as BART (Lewis et al., 2020), LLaVA (Liu et al., 2024), and Qwen (Bai et al., 2025) have demonstrated significant efficacy in learning shared embedding spaces to solve geometrical problems effectively. In a similar context, theorem-based geometry solver frameworks have also shown strong performances (Lu et al., 2021). Furthermore, architectural refinements, such as the enhanced joint alignment for improved image understanding proposed by Gao et al. (2023), have yielded favorable results.

However, domain-specific supervised fine-tuning is resource-intensive and does not always achieve optimal performance, even with extensive resource usage, as noted in similar work by Gao et al. (2023). Moreover, the challenge of reducing overestimation bias in theorem prediction is not addressed in the work of Peng et al. (2023). Another notable approach by Ping et al. (2025) has found impressive outcomes using ground-truth formal language for geometrical diagrams. A substantial drawback, however, is that its dependence on human expertise for generating this formal language requires extensive manual labor. This raises a key question: *Can modern VLMs and LLMs work together effectively to solve geometry problems in a zero-shot setting without task-specific training or* *manual formalization?* Consequently, this study aims to harness the capabilities of state-of-the-art VLMs for this purpose. We initiate an investigation into their unexplored potential for both forging formal language and solving geometrical problems directly in a zero-shot context, thereby addressing a key opening in the current research landscape.

In this study, we propose an multi-agent pipeline for geometry problem solving, named Interpreter-Solver. This pipeline leverages a uniquely tailored two-stage process, combining an Interpreter Agent and a Solver Agent to optimize the balance between solving complexity and computational efficiency. The contributions of this study are summarized below:

- An efficient two-stage framework called Interpreter-Solver is proposed, which splits the visual perception and reasoning tasks in geometric problem-solving, using VLMs for formal language generation and LLMs for problem-solving.
- Interpreter-Solver is benchmarked on the `Geometry3K` and `MathVerse` datasets, showcasing state-of-the-art performance in zero-shot settings. It even outperforms state-of-the-art models by a significant margin on the `MathVerse` dataset even with 4-bit quantized models. We thoroughly examined important research questions (RQs) and provided potent examples to firmly support our findings.
- The efficacy of using VLMs to automate the generation of formal language from diagrams in a zero-shot context is showcased, addressing the limitations of previous methods that required extensive manual labor.
- An ablation study analyzes the impact of predicates on the performance of vision-language models. It shows that while predicates benefit open-source models, they can slightly impede the performance of more advanced enterprise models such as `Gemini 2.0 Flash`.

2 Related Work

The task of solving geometrical problems has gained significant attention due to its complexity, requiring both an understanding of abstract concepts and symbolic reasoning along with multimodal comprehension, which leads to the emergence of novel insights within methods and datasets. Recent research has focused on developing robust models and datasets to address the challenges of geometrical problem-solving. Our comprehensive analysis has identified several modern approaches, including neural-based (Gao et al., 2023; Huang et al., 2025a; Cho et al., 2025), symbolic-based (Lu et al., 2021), and neuro-symbolic-based methods (Kazemi et al., 2023; Zhuang et al., 2025; Xu et al., 2024; Ping et al., 2025), that are devoted to geometrical tasks.

Among symbolic approaches, Lu et al. (2021) proposed a novel framework that takes problem text and diagrams as input and decodes them into formal language descriptions utilizing an automatic parser. The framework includes theorem knowledge as conditional rules and performs symbolic reasoning step by step, acquiring an accuracy of 57.5

The advent of MMLMs has led to the widespread adoption of transformer-based architectures, where Qwen (Bai et al., 2025), PaLI (Chen et al., 2023), LLaVA (Liu et al., 2023), Gemini (Comanici et al., 2025), and GPT (Achiam et al., 2023) have been utilized to combine vision and language. Zhuang et al. (2025) proposed Progressive Multimodal Alignment, a three-stage training framework designed to improve the mathematical reasoning capabilities of MMLMs. Pan et al. (2025) developed an automated pipeline for developing step-wise reasoning paths from geometry diagrams, while GeoLogic is introduced to specifically translate between natural language and formal geometric representations. Xu et al. (2024) sought a homogeneous strategy, integrating in-context learning. Zhang et al. (2025) introduced MATHVERSE, a comprehensive benchmark for assessing visual mathematical reasoning in MMLMs, and conducted ample experiments across popular model families. In parallel, Kazemi et al. (2023) took a similar approach.

Our study also reveals that neural models show encouraging advances in complex reasoning tasks. Leveraging contrastive learning (Radford et al., 2021), Gao et al. (2023) enhanced the alignment phase of state-of-the-art vision–language transformers to enhance image understanding for step-by-step answer generation. Similarly, Huang et al. (2025b) and Cho et al. (2025) introduced diverse image–text paired datasets tailored to geometric problem solving. In contrast, Cheng et al. (2025) proposed a unified multimodal geometry proficient model that incorporates problem solving, precise diagram generation, and automated problem design based on key knowledge points. Specifically, Wang et al. (2025) and Deng et al. (2025) developed reinforcement learning frameworks that train smaller large language models to effectively unify auxiliary construction with robust geometric reasoning. Finally, Li et al. (2025) presented a simple visual augmentation framework to enhance perceptual robustness in multimodal LLMs.

3 Methodology

3.1 Problem Formulation & Overview

Consider a geometric image denoted as $\text{IMG} = \{\text{img}_1, \text{img}_2, \ldots, \text{img}_n\}$, where each img_i illustrates a geometric problem. Each corresponding question is denoted by $Q = \{q_1, q_2, \ldots, q_n\}$, where q_i is a question related to the geometric diagram of img_i. The pipeline begins with a vision–language model $\texttt{VL}(\cdot)$, which takes a diagram image alongside an input of text sequence $X_{vl} = [Q, P_1]$, where P_1 acts as the zero-shot diagram parsing prompt. The image is encoded using an image tokenizer $T_{vl_{img}}(\cdot)$, while the text sequence X_{vl} is tokenized by a text tokenizer $T_{vl_{text}}(\cdot)$. Both modalities are projected into a shared embedding space $S_{\text{emb}} = [V(\text{img}_1, X^1_{vl}), V(\text{img}_2, X^2_{vl}), \ldots, V(\text{img}_n, X^n_{vl})]$, where $V(\cdot,\cdot)$ denotes the joint embedding. After passing through the subsequent layers, the model autoregressively generates geometric literals $L = \{l_1, l_2, \ldots, l_n\}$, which represent formalized illustrations of the diagrams in textual format by leveraging shared embeddings and prior generated tokens. Mathematically, the complete process can be depicted using this equation 1. The generated literals L, the actual question Q, and another prompt for zero-shot geometrical problem solving P_2, are fed into a language model $\texttt{LM}(\cdot)$. The input sequence $X_{lm} = [L, Q, P_2]$ is tokenized using a pre-trained tokenizer $T_{lm}(\cdot)$, producing a contextualized vector representation $V = [V_1, V_2, \ldots, V_d]$, where d is the dimentionality. The $\texttt{L}$ autoregressively generates a reasoning process over this representation and prior tokens to derive the final answer $\hat{Y}$. The entire procedure can mathematically be abbreviated as follows 2 and illustrated in Fig. 2:

$$\hat{Y}_{vl} = VL((T_{vl}[IMG], T_{vl}[X_{vl}]), VL^{t-1}_{out}), W^{vl^\star}) \quad (1)$$

$$\hat{Y} = LM((T_{lm}[\hat{Y}_{vl}, X_{lm}], LM^{t-1}_{out}), W^{lm^\star}) \quad (2)$$

Here, $W^{vl^\star}$ and $W^{lm^\star}$ denote the frozen parameters of the vision-language model and LLM, respectively

3.2 Interpreter Agent

Given a multimodal input pair consisting of a text sequence $T = \{t_1, t_2, \ldots, t_n\}$ and an image I, the pre-trained visual language model uses a distinct feature extraction strategy. The image I is first partitioned into a sequence of fixed-size patches $P = \{p_1, p_2, \ldots, p_m\}$, which are processed by a dedicated vision encoder. A subsequent resampling mechanism then distills these patch features into a fixed-size set of continuous visual embeddings E_V. Concurrently, each token t_i in the text sequence undergoes an embedding layer to convert discrete inputs into continuous vector representations, such that $E^T_i = \text{Embed}(t_i)$. The visual embeddings E_V and textual embeddings E_T are combined to create a unified input sequence. These combined embeddings are subsequently enriched with positional encodings PE to account for token order. This unified sequence is then fed into a decoder-only transformer, which consists of a stack of K identical layers. Each layer is composed of two main sub-layers: a multi-head causal self-attention mechanism and a position-wise feedforward network. The causal self-attention mechanism computes weighted representations for each token by attending to all preceding tokens in the interleaved sequence using non-learnable query Q, key K, and value V vectors. This attention mechanism is defined as follows (Vaswani et al., 2017):

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \quad (3)$$

The causal nature of the attention ensures that the prediction for a token at a given position depends only on the available outputs at prior positions encompassing the full set of input visual tokens. The position-wise feedforward networks introduce non-linearity, which additionally refines the representations by utilizing the non-linear activation function. The outputs of each layer are then propagated through K identical layers, resulting in refined representations that capture both local and global dependencies. These vector representations provide rich contextual portrayals of the input, leading to refined formalized text literals, $L = \{l_1, l_2, \ldots, l_n\}$, of the geometrical image.

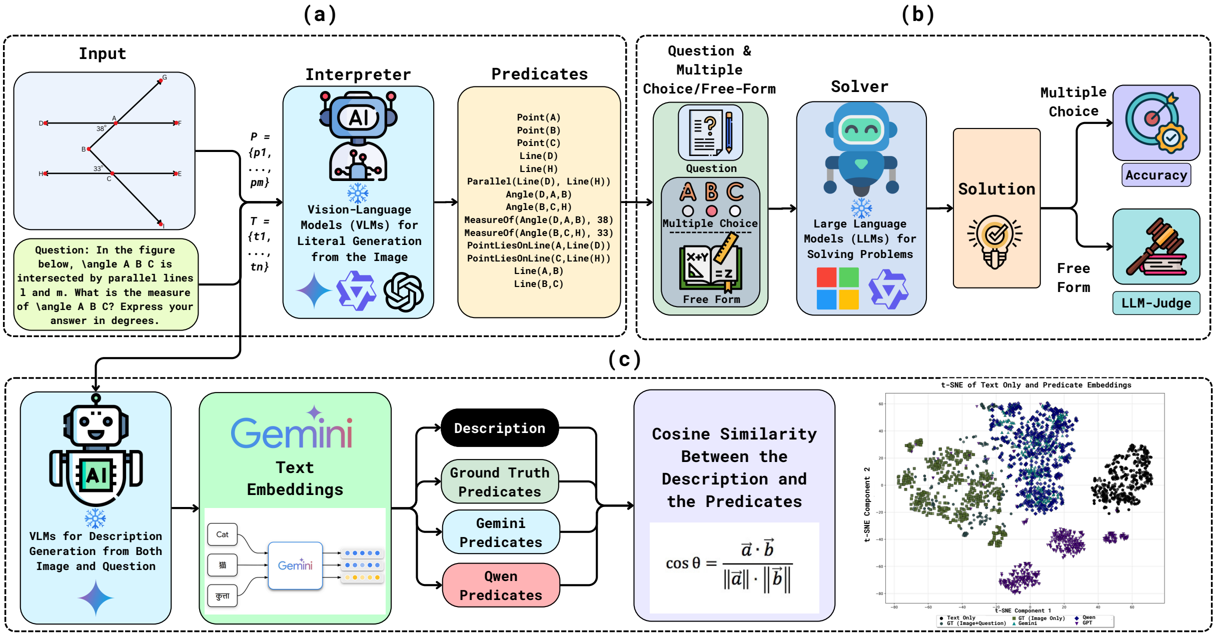

Figure 2: **(a)** An Interpreter Agent generates formal predicates from images and questions using VLMs. **(b)** A Solver Agent then solves the problem using these predicates as LLM input. **(c)** The 2D t-SNE plot visualizes the semantic similarity of generated description and predicate embeddings, indicating the Interpreter's comprehension of predicate generation.

3.3 Solver Agent

Firstly, the literals generated by Interpreter Agent, denoted as $L = \{l_1, l_2, \ldots, l_m\}$, where m is the number of literals, along with the question Q and the prompt P, are concatenated into a sequence $Y = [L, Q, P]$. This sequence is embedded into a latent space using non-learnable token embeddings $E(y_i) = Embed(y_i)$, followed by rotary position embeddings (RoPE) to encode positional information. The final input representations are computed as $Z_{y_i} = E(y_i) \oplus RoPE(y_i)$, where $\oplus$ denotes the positional rotations. These representations are passed through a stack of L transformer decoder layers. Each layer employs grouped-query self-attention (GQA), upholding robustness in capturing long-range dependencies. The masked multi-head self-attention operation (as given in Equation 3) is adapted to GQA, with queries projected into h heads and keys/values grouped into g smaller groups ($g < h$). This is followed by SwiGLU activation in the position-wise feedforward networks (FFN), which improves expressiveness by introducing multiplicative non-linearities. The decoder layers also integrate parallel attention and MLP pre-normalization using RMSNorm. Representations are iteratively refined through the stack of decoder layers, pinnacling in the final answer Y.

4 Experimental Analysis

In this section, we outline the experimental setup, which includes the model configuration, datasets, and evaluation metrics.

4.1 Datasets

To meticulously evaluate our proposed framework, we utilized two comprehensive benchmark datasets: Geometry3K and MATHVERSE.

- **Geometry3K (Lu et al., 2021)** `Geometry3K` is a large-scale benchmark dataset designed for solving geometry problems. It consists of 3,001 multiple-choice geometry problems accumulated from high-school textbooks, including both text and a diagram. We evaluated our framework on the official test set, which comprises 601 multiple-choice problems.

- **MATHVERSE (Zhang et al., 2025)** Additionally, we evaluated our framework on the `MATHVERSE` dataset, which comprises 2,612 high-quality visual math problems. For our evaluation, we sourced the question from the text-only version of the dataset and the affiliated image from the vision-dominant version. We present our evaluation on the test mini subset, which contains 788 problems. This subset consists of 436 multiple-choice problems and 352 free-form problems.

4.2 Models

We evaluated a range of state-of-the-art LLMs and VLMs to estimate their reasoning and diagram parsing abilities. Due to computational constraints, all open-source models were deployed using 4-bit precision using the unsloth library. We selected powerful open-source models like `Qwen-8B (LLM)` and `Qwen-2.5-7B/32B (VLMs)` for their well-documented zero-shot reasoning and advanced image understanding. We also incorporated `Phi-4`, an efficient Small Language Model (SLM) known for elegant performance despite its smaller number of parameters. To benchmark these against leading solutions, we included the high-performance proprietary model `Gemini 2.0 Flash` for both LLM and VLM tasks for comparing open-source evaluation results with an enterprise-grade model.

4.3 Evaluation Metrics

To facilitate an exhaustive performance comparison, our evaluation methodology employs a distinct accuracy metric for both multiple-choice and free-form questions. For multiple-choice tasks, a response is considered correct if the model's output either straight matches an option or, in the case of a numerical response, is most comparable in value to one of the given choices. For the more nuanced free-form tasks, answers are rigorously validated for numerical equivalence against ground-truth values. This validation is conducted by leveraging the concept of `LLM as a judge`, powered by `Gemini 2.0 Flash` (see Appendix 22, 23). The entire judgment process can be formally represented using the mathematical notation 4 provided below:

$$J : (A_{llm}, A_{gt}) \mapsto (R, [\|v(A_{llm}) - v(A_{gt})\| \leq \epsilon]) \tag{4}$$

Here, the judge function, J, takes Interpreter-Solver's answer (A_{llm}) and the actual ground truth (A_{gt}) as input. It uses a valuation function, v, to transform them to their true mathematical values and determines if the distance between them, calculated by the norm $\|\cdot\|$, is within a predefined tolerance, ϵ. The function returns a tuple containing thorough reasoning for the decision (R) and a binary score (1 for correct, 0 for incorrect). Any cases that cannot be resolved are conservatively categorized as incorrect. Performance for tasks involving MLLMs and LLMs is evaluated using the `Pass@3` metric, which represents success as achieving a solution in at least one of three independent attempts.

Method	#Params.	Accuracy
Geometry3K		
`Inter-GPS` (Lu et al., 2021)	406M	57.5%
`GeoDRL` (Peng et al., 2023)	44M	68.4%
`AutoGPS` (Ping et al., 2025)	≈200B	81.6%
`Interpreter-Solver-Phi-4 (Ours)`	14B-4bit	70.05%
`Interpreter-Solver-Qwen-3 (Ours)`	8B-4bit	79.53%
`Interpreter-Solver-Gemini-2.0-Flash (Ours)`	≈40B	83.19%
MathVerse		
`G-LLaVa` (Gao et al., 2023)	13B	16.6%
`MathVerse` (Zhang et al., 2025)	7B	25.9%
`OpenVLThinker` (Deng et al., 2025)	7B	47.9%
`Interpreter-Solver-Qwen-3 (Ours)`	8B-4bit	69.67%

Table 1: Comparison of `Interpreter-Solver` accuracy on the Geometry3K and MathVerse datasets.

5 Results and Analysis

5.1 Quantitative Reults

We scrutinize the performance of our proposed method, Interpreter-Solver, as shown in Table 1, which integrates our Thinker and Solver agents across two geometric benchmark datasets. On the `Geometry3k` dataset, Interpreter-Solver demonstrates state-of-the-art performance in zero-shot settings. When employing a 4-bit quantized, open-source model, our framework works with a parameter size that is 4.7× smaller than the current state-of-the-art, demonstrating only a ≈2% drop in performance. In contrast, when integrating an enterprise model `Gemini 2.0 Flash` that is 5× smaller than the state-of-the-art, this performance gap increases to ≈2%. Furthermore, the efficacy of our method is particularly enunciated on the `MathVerse` dataset, showcasing a substantial improvement of ≈44% in overall accuracy. Additionally, our `Interpreter-Solver`(Qwen) model achieved 79.03% accuracy on reannotated AutoGPS(Ping et al., 2025) images. All experiments were conducted in a zero-shot setting with controlled prompting (see Appendix 4 – 9) for both multiple-choice (see Appendix 10 – 12) and free-form (see Appendix 13, 14, 15) questions.

5.2 Predicate Alignment and Quality

To see how good the generated predicates are, we first generated a natural language description from both the image and the question using `Gemini-2.5-Flash` model. Then, we obtained text embeddings using the `gemini-embedding-001` model. Finally, we computed the cosine similarity between the description and the predicate embeddings to assess their semantic alignment, illustrated in Figure 2. Our results show that the average cosine similarity between the description and the predicates generated by the `Interpreter` (Gemini: 0.646, Qwen: 0.641, GPT: 0.637) is higher than that with the ground-truth predicates (Image+Question: 0.644, Image-only: 0.639) indicating their ability to produce high-quality predicates that are comparable to manual annotations, shown in Figure 3.

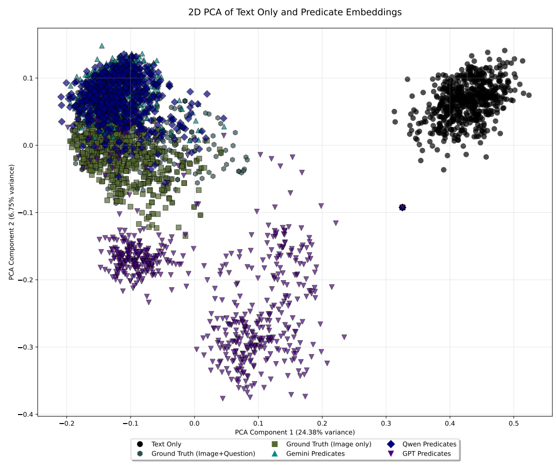

Figure 3: This 2D PCA plot visualizes predicate embeddings, showing text-only embeddings for Ground Truth (Image+Question), Ground Truth (Image only), and model-generated predicates (Gemini, Qwen, GPT).

5.3 Analysis of Interpreter-Solver's Reasoning

How does the presence of a list of possible answers influence the Solver's reasoning when faced with ambiguous problems? Providing predefined answer choices greatly enhances Solver's ability to tackle ambiguous problem statements. When faced with contradictions in free-form settings, the Solver frequently struggles to find a solution. However, multiple-choice options guide the Solver to explore different interpretations, often leading to explicit statements like, *"But since the problem provides the angle, we must use that,"* frequently leading to the correct answer (see Appendix 16). This also leads to more concise reasoning. Our empirical results on the `MathVerse` dataset support this: we achieved 81.88% accuracy on 436 multiple-choice questions versus 54.55% on 352 free-form questions.

The Conundrum of LLM Self-Doubt: When Internal Logic Meets External Constraints. Our detailed analysis showed that when an LLM consistently produces reasoning that contradicts a problem's external constraints, it can develop a state of recursive self-doubt. This type of behaviour is noticed where the predicate contains multiple color denoting internal relationships of a geometrical problem. Instead of discarding its reasoning, the model repeatedly re-examines its calculations and reinterprets key terms, often without adding new information (see Appendix 17). This looping behavior arises when the model detects a mismatch in inital attempts: its computed answer seems valid but doesn't fit the given options. This leads to cycles of verification and reinterpretation that, despite minor adjustments, result in the same unsatisfactory conclusion.

How can LLMs identify and disrupt self-reinforcing reasoning loops that are triggered by contradictory assumptions? Our analysis of Solver's (Phi-4) performance on geometric reasoning tasks reveals a vital susceptibility in current language model architectures. When initial assumptions possess fundamental errors (such as incorrect similarity ratios in triangle problems or contradictory angle measurements), models enter self-reinforcing reasoning loops instead of identifying and fixing foundational mistakes. Similarly, in some triangle problems, Phi-4 repeatedly miscalculated ratios while cycling through different variations without questioning the erroneous steps. Likewise, the model oscillated between angle calculations despite the problem's internal inconsistency, producing disconnected results that eventually settled on an incorrect answer (see Appendix 18). These patterns suggest that language models lack meta-cognitive monitoring capabilities to detect when their reasoning has become circular, instead treating each iteration as progressive despite identical logical foundations.

How does the "reassessment" process in the phi-4 model expose fundamental deficits in applying geometric theorems and logical validation? Our study indicates that the Solver's (phi-4) reassessment mechanism fails to correct initial errors in specific types of geometry problems due to a core insufficiency in applying and verifying foundational theorems, rather than serving as a genuine logical re-evaluation (see Appendix 18). The pattern appears most clearly in problems requiring a robust understanding of angle relationships and the ability to resolve contradictions. For example, the model misapplied inscribed angle theorems in circle geometry, repeatedly failed to correctly use angle properties of parallelogram diagonals,

Geometry3K							
Methods	#Params.	Interpreter-Solver (VLM)			Single Agent		
Qwen 2.5 VL	7B	60.07%			53.24%		
Qwen 2.5 VL	32B	72.05%			68.72%		
Gemini 2.0 Flash	≈40B	83.86%			85.19%		
MathVerse							
Methods	#Param	Interpreter-Solver (VLM)			Single Agent		
		Multiple Choice	Free Form	Overall Accuracy	Multiple Choice	Free Form	Overall Accuracy
Qwen 2.5 VL	7B	53.67%	36.93%	46.19%	58.94%	43.75%	52.16%
Qwen 2.5 VL	32B	78.44%	54.55%	67.77%	76.38%	54.55%	66.67%

Table 2: Predicate influence on zero-shot geometrical problem-solving with `Interpreter-Solver(VLM)`.

and ultimately mistrusted its correct calculation for a regular polygon. This suggests the model's re-assessment is not a process of re-reasoning from first principles but a simple check that falters when its internal logic conflicts with external constraints, exposing a vital failure in its capability for robust mathematical validation.

What are the key failure modes in Qwen 2.5-32B's reasoning that lead to its significant performance gap with Gemini on visual geometry tasks? Gemini outperforms Qwen 2.5-32B on geometry reasoning tasks due to superior OCR, logical consistency, and contradiction handling. Gemini accurately identifies key visual elements such as angles and lines, whereas Qwen often misreads or omits critical information, such as misidentifying vertical angles, leading to unsupported conclusions. In reasoning, Qwen frequently shifts between geometric rules without justification; for instance, it begins with an angle-sum around a point, then switches to a triangle sum rule mid-reasoning and incorrectly concludes, while ignoring essential relationships. In contrast, Gemini maintains logical coherence and selectively applies relevant geometric principles. Its ability to detect contradictions is also notable; for example, when a problem leads to an impossible result like negative angle measures, Gemini flags the inconsistency and re-evaluates the answer contextually. Gemini also avoids overcomplication, unlike Qwen, which prematurely applies superficial heuristics. However, Gemini is not without flaws: in one case involving a centroid, it claims "Since M is the centroid, $CM = 2 \times MR$," with given values $CM = 7$ and $MR = 4$ (implying $2 \times MR = 8 \neq 7$), recognizes the contradiction, but proceeds with the incorrect assumption. Still, Gemini's strong extraction and contradiction detection make it better for complex geometry than Qwen.

How does the quality of literals affect the performance of different LLMs on geometry tasks? We observe a notable divergence in model behavior when comparing performance on Gemini-extracted versus ground-truth literals: Phi-4 improves when using Gemini-extracted literals, while Qwen-8B sees a performance drop under the same conditions. This contrast arises from fundamental differences in how the models handle symbolic reasoning and input structure. Gemini-extracted literals, though occasionally noisy, repetitive, and biased toward simple function-like forms, provide detailed and explicit representations that seem to compensate for Phi-4's limitations in symbolic interpretation (see Appendix 20). Phi-4 struggles with symbolic correspondence in angle reasoning (see Appendix 21), but benefits from explicit, even imperfect, input. On the other hand, Qwen-8B performs best with ground-truth literals, which are concise, well-structured, and symbolically accurate. However, Qwen's performance declines with Gemini-extracted literals due to increased verbosity, redundant constraints, and symbolic inconsistencies. These introduce noise, obscure key logical relationships, and mislead Qwen's prioritization of predicates. For instance, overloaded expressions cause confusion in resolving dependencies, and minor naming inconsistencies reduce symbolic clarity. Thus, while Phi-4 benefits from explicitness, even if imperfect, Qwen is more sensitive to clutter and performs optimally with clean, structured symbolic input.

Why did Gemini perform worse with its own literals than without? Both Gemini (**I**mage + **G**emini-extracted **L**iterals + **Q**uestion + **C**hoices) and Gemini (**I**mage + **Q**uestion + **C**hoices) configurations performed well, achieving accuracies of 83.86% and 85.19% respectively, indicating that the model is generally effective at using images to reason about geometric problems. While the GL component was intended to provide structured information to aid reasoning, it often introduced errors such as inaccurate predicates, misinterpreted angle measures, or contradictions not reflected in the diagram. These flawed predicates frequently misled the model, causing it to abandon correct mathematical reasoning to match incorrect constraints or make arbitrary guesses when its conclusions didn't align with the provided answer choices. In contrast, the (I-Q-C) setup allowed the model to rely more directly on the visual content and the question structure, which allows to focus only on the relevant information—often extracting just what was necessary to answer the question. Thus, while both approaches effectively utilized image and language inputs, the cleaner and less constrained reasoning in (I-Q-C) enabled it to surpass the more error-prone (I-GL-Q-C) configuration. However, it is worth noting that in some cases where the image alone was ambiguous, the GL component provided valuable information that helped guide the reasoning more effectively.

5.4 Ablation Study

Table 2 elucidates the performance of open-source and enterprise `Interpreter-Solver` VLM models under two distinct circumstances: with and without the inclusion of predicates as additional information. For open-source models, the inclusion of predicates resulted in a notable performance increase. Particularly, on the `Geometry3K` dataset, the Interpreter-Solver(`Qwen 2.5-7B`)'s accuracy improved by ≈8%, while the Interpreter-Solver(`Qwen 2.5-32B`) showed an improvement of ≈3.5%. A similar pattern was observed for the Interpreter-Solver(`Qwen 2.5-32B`) on the `MathVerse` dataset. Besides, our experimental results confirm that models with a larger number of parameters typically outperform their smaller counterparts. In contrast, the enterprise model, Single Agent(`Gemini`), achieved ≈1.5% better when predicates were excluded from the input sequence. This suggests that for advanced multimodal models, additional text guidance can hinder, rather than help, image understanding, resulting in poorer outcomes. To identify the optimal Interpreter-Solver configuration, we evaluated various Interpreter-Solver pairs depicted in Table 3, uncovering that predicates from `GPT-4o mini`, while unsatisfactory to `Gemini`'s, still barely exceeded the 4-bit quantized `Qwen` model. Our comprehensive ablation demonstrates that our two-stage Interpreter-Solver framework overcomes the performance degradation typical in open-source Single Agent on visual mathematical reasoning tasks. Interpreter-Solver(`Gemini-Gemini`) setup achieves state-of-the-art results. It offers competitive performance with the quantized Interpreter-Solver(`Gemini-Qwen`) variant, and surpasses the AutoGPS (Ping et al., 2025) benchmark via our zero-shot strategy on a 4-bit quantized Single Agent`Qwen 2.5 VL`. Moreover, with ground-truth predicates, Interpreter-Solver(`Gemini`)'s reasoning (89.18%) proved exceptional, significantly surpassing Interpreter-Solver(`Qwen 2.5-7B`) (55.07%) and Interpreter-Solver(`Qwen 2.5-32B`) (46.26%). Conversely, both Interpreter-Solver(`Qwen`) models performed greatly better using predicates generated by `Gemini`.

Interpreter	#Params.	Solver	
		`Phi-4`	`Qwen 3`
`Qwen-2.5`	7B	35.77%	42.26%
`Qwen-2.5`	32B	56.74%	61.23%
`GPT-4o mini`	≈8B	58.24%	63.23%
`Gemini`	≈40B	70.05%	79.53%

Table 3: Comparison of the accuracy of different Interpreter-Solver settings on the Geometry3K and MathVerse datasets.

6 Conclusion

This study presents a memory-efficient, zero-shot baseline for the task at hand, proposing the Interpreter-Solver, a two-stage pipeline that leverages VLMs and LLMs, incorporating zero-shot prompting without relying on traditional strategies. Interpreter-Solver outperformed both neuro-symbolic and neural methods, solidifying its status as a new state-of-the-art approach. Notably, we also affirm that a 4-bit quantized model can achieve competitive results, with only a ≈2% performance decrease despite being 5× smaller in parameter count. Our open-source Interpreter-Solver (VLM) outperforms Single Agent VLMs in zero-shot, while the enterprise Interpreter-Solver(Gemini) achieves even better results. Accordingly, our work inquires the prevailing notion that Supervised Fine-Tuning might be the only adequate approach for geometric problem-solving, extending the relevance and performance of zero-shot, prompt-based methods to this complex domain.

Limitations

While Interpreter-Solver offers a robust pipeline for geometrical problem-solving, it has several limitations. First, our study exclusively employed zero-shot prompting, with the parameters of our Interpreter and Solver remaining frozen. This approach hinders the model's ability to capture global dependencies between images and predicates, which could improve with supervised fine-tuning of the Interpreter's cross-modality alignment. Additionally, fine-tuning the Solver may also enhance performance on the test set. Second, our methodology was restricted to a single-prompt approach. The adoption of iterative prompting techniques could progressively enhance the performance of Interpreter-Solver by allowing for sequential improvement of the reasoning process. Thirdly, incorporating a few-shot or one-shot examples would provide Interpreter-Solver with helpful prior knowledge, potentially improving its reasoning capabilities. Fourth, the advanced prompting strategies, such as Atom of Thoughts (AoT) (Teng et al., 2025) or Program of Thoughts (PoT) (Chen et al., 2022), which decompose problems into simpler, executable steps, could greatly facilitate the reasoning process. Fifth, incorporating LLM as a judging concept can enhance model reliability and significantly improve the accuracy of final outputs. Finally, our experiments were conducted under resource limitations, necessitating the use of 4-bit quantized models via the Unsloth library. Hence, our reliance on open-source Vision-Language Models (VLMs) and Large Language Models (LLMs) may not fully reflect the performance and behavior of larger, full-precision models or state-of-the-art proprietary systems like GPT-4.5, Gemini-2.5 Pro, or Claude-4.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish Thapliyal, James Bradbury, and 10 others. 2023. Pali: A jointly-scaled multilingual language-image model. *Preprint*, arXiv:2209.06794.

Jo-Ku Cheng, Zeren Zhang, Ran Chen, Jingyang Deng, Ziran Qin, and Jinwen Ma. 2025. Geouni: A unified model for generating geometry diagrams, problems and problem solutions. *arXiv preprint arXiv:2504.10146*.

Seunghyuk Cho, Zhenyue Qin, Yang Liu, Youngbin Choi, Seungbeom Lee, and Dongwoo Kim. 2025. Geodano: Geometric vlm with domain agnostic vision encoder. *arXiv preprint arXiv:2502.11360*.

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

Yihe Deng, Hritik Bansal, Fan Yin, Nanyun Peng, Wei Wang, and Kai-Wei Chang. 2025. Openvlthinker: An early exploration to complex vision-language reasoning via iterative self-improvement. *arXiv preprint arXiv:2503.17352*.

Jiahui Gao, Renjie Pi, Jipeng Zhang, Jiacheng Ye, Wanjun Zhong, Yufei Wang, Lanqing Hong, Jianhua Han, Hang Xu, Zhenguo Li, and 1 others. 2023. G-llava: Solving geometric problem with multi-modal large language model. *arXiv preprint arXiv:2312.11370*.

Zihan Huang, Tao Wu, Wang Lin, Shengyu Zhang, Jingyuan Chen, and Fei Wu. 2025a. Autogeo: Automating geometric image dataset creation for enhanced geometry understanding. *IEEE Transactions on Multimedia*.

Zihan Huang, Tao Wu, Wang Lin, Shengyu Zhang, Jingyuan Chen, and Fei Wu. 2025b. Autogeo: Automating geometric image dataset creation for enhanced geometry understanding. *IEEE Transactions on Multimedia*, 27:3105–3116.

Mehran Kazemi, Hamidreza Alvari, Ankit Anand, Jialin Wu, Xi Chen, and Radu Soricut. 2023. Geomverse: A systematic evaluation of large models for geometric reasoning. *arXiv preprint arXiv:2312.12241*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020.

BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yuting Li, Lai Wei, Kaipeng Zheng, Jingyuan Huang, Linghe Kong, Lichao Sun, and Weiran Huang. 2025. Vision matters: Simple visual perturbations can boost multimodal math reasoning. *arXiv preprint arXiv:2506.09736*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pages 34892–34916. Curran Associates, Inc.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 2021. Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6774–6786, Online. Association for Computational Linguistics.

Yicheng Pan, Zhenrong Zhang, Pengfei Hu, Jiefeng Ma, Jun Du, Jianshu Zhang, Quan Liu, Jianqing Gao, and Feng Ma. 2025. Enhancing the geometric problem-solving ability of multimodal llms via symbolic-neural integration. *arXiv preprint arXiv:2504.12773*.

Shuai Peng, Di Fu, Yijun Liang, Liangcai Gao, and Zhi Tang. 2023. GeoDRL: A self-learning framework for geometry problem solving using reinforcement learning in deductive reasoning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13468–13480, Toronto, Canada. Association for Computational Linguistics.

Bowen Ping, Minnan Luo, Zhuohang Dang, Chenxi Wang, and Chengyou Jia. 2025. Autogps: Automated geometry problem solving via multimodal formalization and deductive reasoning. *arXiv preprint arXiv:2505.23381*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR.

Fengwei Teng, Zhaoyang Yu, Quan Shi, Jiayi Zhang, Chenglin Wu, and Yuyu Luo. 2025. Atom of thoughts for markov llm test-time scaling. *arXiv preprint arXiv:2502.12018*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yikun Wang, Yibin Wang, Dianyi Wang, Zimian Peng, Qipeng Guo, Dacheng Tao, and Jiaqi Wang. 2025. Geometryzero: Improving geometry solving for llm with group contrastive policy optimization. *arXiv preprint arXiv:2506.07160*.

Shihao Xu, Yiyang Luo, and Wei Shi. 2024. Geo-llava: A large multi-modal model for solving geometry math problems with meta in-context learning. In *Proceedings of the 2nd Workshop on Large Generative Models Meet Multimodal Applications*, pages 11–15.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, Peng Gao, and Hongsheng Li. 2025. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *Computer Vision – ECCV 2024*, pages 169–186, Cham. Springer Nature Switzerland.

Wenwen Zhuang, Xin Huang, Xiantao Zhang, and Jin Zeng. 2025. Math-puma: Progressive upward multimodal alignment to enhance mathematical reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 26183–26191.

A Appendix

In this Appendix, we reproduce the exact prompts and guidelines referenced in the main text for ease of reference.

Prompt for Zero-Shot Geometry Predicates Generation

You are an expert AI mathematician specializing in geometry. Your task is to analyze the geometric figure in the provided image and generate accurate geometric predicates (literals) that represent ALL the relationships, measurements, and properties shown in the diagram.
GEOMETRY PROBLEM IMAGE: The image shows a geometric figure with various shapes, lines, angles, and measurements. Analyze this image carefully to understand all geometric relationships and constraints.
Question: [Given Question]

YOUR TASK:

1. First, provide step-by-step reasoning showing your analysis process
2. Then, generate geometric predicates based on your analysis using the Guidelines below

STEP-BY-STEP ANALYSIS (Required):
Please follow this format for your reasoning:

1. COMPREHENSIVE IMAGE ANALYSIS
   - Identify ALL geometric shapes (circles, triangles, quadrilaterals, etc.)
   - List ALL points, lines, and their labels/names
   - Note ALL visible measurements, angles, and numerical values
   - Identify ALL special markings (right angle symbols, parallel marks, congruent marks, equal marks, etc.)
   - Look for implied constructions (perpendiculars, bisectors, tangents, chords, radii, etc.)
2. CIRCLE-SPECIFIC ANALYSIS (If circles are present)
   - Identify the center and all points on the circle
   - Determine which lines are radii, chords, diameters, or tangents
   - Look for inscribed angles, central angles, and arc relationships
   - Check for perpendicular relationships involving radii and chords
   - Identify any equal radius relationships
3. ANGLE AND PERPENDICULARITY ANALYSIS
   - Examine ALL angles shown in the diagram, both marked and unmarked
   - Look for right-angle indicators or perpendicular relationships
   - Check for angle bisectors or special angle relationships
   - Identify complementary, supplementary, or vertical angles
   - Look for inscribed angles and their corresponding arcs
4. CONGRUENCE AND EQUALITY ANALYSIS
   - Identify ALL equal lengths, angles, or shapes (look for tick marks, identical measurements)
   - Check for congruent triangles or similar figures
   - Look for equal radii in circles
   - Identify parallel lines or equal distances
5. INTERSECTION AND POSITIONING ANALYSIS
   - Determine where lines intersect and at what points
   - Check if points lie on specific lines or circles
   - Identify midpoints, centroids, or other special points
   - Look for points that divide segments in specific ratios
6. CONSTRAINT AND RELATIONSHIP SYNTHESIS
   - Combine observations to identify implicit relationships
   - Look for theorem applications (Pythagorean, inscribed angle, etc.)
   - Identify geometric constructions that create specific relationships
   - Check for properties that follow from the given constraints

Table 4: Prompt for Zero-Shot Geometry Predicates Generation Part 1

Prompt for Zero-Shot Geometry Predicates Generation

1. QUESTION-DRIVEN COMPLETENESS CHECK
    - Ensure all information needed to solve the problem is captured
    - Verify that key relationships for the solution are represented
    - Double-check that no critical geometric properties are missed
    - Confirm that the predicates will provide sufficient information for problem-solving

CRITICAL ANALYSIS GUIDELINES:

- **LOOK FOR HIDDEN RELATIONSHIPS**: Many geometric problems have implicit perpendicular relationships, equal lengths, or special angle properties that aren't explicitly marked but are crucial for solving.
- **CIRCLE GEOMETRY FOCUS**: If the diagram contains circles, pay special attention to:
    - Which points lie on the circle vs. inside/outside
    - Perpendicular relationships between radii and chords
    - Equal radius lengths
    - Inscribed vs. central angles
    - Tangent-radius perpendicularity
- **CONSTRUCTION INDICATORS**: Look for:
    - Lines that appear to be perpendicular even without explicit markings
    - Points that appear to be midpoints or special positions
    - Equal lengths suggested by visual symmetry
    - Angle relationships implied by the construction

GUIDELINES:
***Follow these predicates to represent diagram literals.

GEOMETRIC SHAPES:

- Point: Point(A), Point()
- Line: Line(A,B), Line(m), Line()
- Angle: Angle(A,B,C), Angle(A), Angle(1), Angle()
- Triangle: Triangle(A,B,C), Triangle(), Triangle(1,2,3)
- Quadrilateral: Quadrilateral(A,B,C,D), Quadrilateral()
- Parallelogram: Parallelogram(A,B,C,D), Parallelogram(1), Parallelogram()
- Square: Square(A,B,C,D), Square(1), Square()
- Rectangle: Rectangle(A,B,C,D), Rectangle(1), Rectangle()
- Rhombus: Rhombus(A,B,C,D), Rhombus(1), Rhombus()
- Trapezoid: Trapezoid(A,B,C,D), Trapezoid(1), Trapezoid()
- Kite: Kite(A,B,C,D), Kite(1), Kite()
- Polygon: Polygon()
- Pentagon: Pentagon(A,B,C,D,E), Pentagon()
- Hexagon: Hexagon(A,B,C,D,E,F), Hexagon()
- Heptagon: Heptagon(A,B,C,D,E,F,G), Heptagon()
- Octagon: Octagon(A,B,C,D,E,F,G,H), Octagon()

Table 5: Prompt for Zero-Shot Geometry Predicates Generation Part-2

Prompt for Zero-Shot Geometry Predicates Generation Part-3

- Circle: Circle(A), Circle(1), Circle()
- Arc: Arc(A,B), Arc(A,B,C), Arc()
- Sector: Sector(O,A,B), Sector()
- Shape: Shape() // For unknown shapes or regions

UNARY GEOMETRIC ATTRIBUTES:

- RightAngle: RightAngle(Angle())
- Right: Right(Triangle()) // Right triangle
- Isosceles: Isosceles(Polygon()) // Isosceles polygon
- Equilateral: Equilateral(Polygon()) // Equilateral polygon
- Regular: Regular(Polygon())
- Red: Red(Shape())
- Blue: Blue(Shape())
- Green: Green(Shape())
- Shaded: Shaded(Shape())

GEOMETRIC ATTRIBUTES:

- AreaOf: AreaOf(A)
- PerimeterOf: PerimeterOf(A) // Perimeter of polygon A
- RadiusOf: RadiusOf(A)
- DiameterOf: DiameterOf(A)
- CircumferenceOf: CircumferenceOf(A) // Perimeter of circle A
- AltitudeOf: AltitudeOf(A) // Altitude of polygon A
- HypotenuseOf: HypotenuseOf(A) // Hypotenuse of triangle A
- SideOf: SideOf(A) // Side of square A
- WidthOf: WidthOf(A) // Width of quadrilateral A
- HeightOf: HeightOf(A) // Height of quadrilateral A
- LegOf: LegOf(A) // Leg of trapezoid A
- BaseOf: BaseOf(A) // Base of polygon A
- MedianOf: MedianOf(A) // Median of polygon A
- IntersectionOf: IntersectionOf(A,B) // Intersection of shapes A and B
- MeasureOf: MeasureOf(A) // Measure of angle A
- LengthOf: LengthOf(A) // Length of line A
- ScaleFactorOf: ScaleFactorOf(A,B) // Scale factor of shape A to shape B

Table 6: Prompt for Zero-Shot Geometry Predicates Generation Part-3

Prompt for Zero-Shot Geometry Predicates Generation Part-4

BINARY GEOMETRIC RELATIONS:

- PointLiesOnLine: PointLiesOnLine(Point(),Line(1,2))
- PointLiesOnCircle: PointLiesOnCircle(Point(),Circle())
- Parallel: Parallel(Line(),Line())
- Perpendicular: Perpendicular(Line(),Line())
- IntersectAt: IntersectAt(Line(),Line(),Line(),Point())
- BisectsAngle: BisectsAngle(Line(),Angle())
- Congruent: Congruent(Polygon(),Polygon())
- Similar: Similar(Polygon(),Polygon())
- Tangent: Tangent(Line(),Circle())
- Secant: Secant(Line(),Circle())
- CircumscribedTo: CircumscribedTo(Shape(),Shape())
- InscribedIn: InscribedIn(Shape(),Shape())

A-IsXOf-B GEOMETRIC RELATIONS:

- IsMidpointOf: IsMidpointOf(Point(),Line()) // Point A is midpoint of line B
- IsCentroidOf: IsCentroidOf(Point(),Shape()) // Point A is centroid of shape B
- IsIncenterOf: IsIncenterOf(Point(),Shape()) // Point A is incenter of shape B
- IsRadiusOf: IsRadiusOf(Line(),Circle()) // Line A is radius of circle B
- IsDiameterOf: IsDiameterOf(Line(),Circle()) // Line A is diameter of circle B
- IsMidsegmentOf: IsMidsegmentOf(Line(),Triangle()) // Line A is midsegment of triangle B
- IsChordOf: IsChordOf(Line(),Circle()) // Line A is chord of circle B
- IsSideOf: IsSideOf(Line(),Polygon()) // Line A is side of polygon B
- IsHypotenuseOf: IsHypotenuseOf(Line(),Triangle()) // Line A is hypotenuse of triangle B
- IsPerpendicularBisectorOf: IsPerpendicularBisectorOf(Line(),Triangle()) // Line A is perpendicular bisector of triangle B
- IsAltitudeOf: IsAltitudeOf(Line(),Triangle()) // Line A is altitude of triangle B
- IsMedianOf: IsMedianOf(Line(),Quadrilateral()) // Line A is median of quadrilateral B
- IsBaseOf: IsBaseOf(Line(),Quadrilateral()) // Line A is base of quadrilateral B
- IsDiagonalOf: IsDiagonalOf(Line(),Quadrilateral()) // Line A is diagonal of quadrilateral B
- IsLegOf: IsLegOf(Line(),Trapezoid()) // Line A is leg of trapezoid B

Table 7: Prompt for Zero-Shot Geometry Predicates Generation Part-4

Prompt for Zero-Shot Geometry Predicates Generation Part-5

NUMERICAL ATTRIBUTES AND RELATIONS:

- SinOf: SinOf(Var)
- CosOf: CosOf(Var)
- TanOf: TanOf(Var)
- CotOf: CotOf(Var)
- HalfOf: HalfOf(Var)
- SquareOf: SquareOf(Var)
- SqrtOf: SqrtOf(Var)
- RatioOf: RatioOf(Var), RatioOf(Var1,Var2)
- SumOf: SumOf(Var1,Var2,...)
- AverageOf: AverageOf(Var1,Var2,...)
- Add: Add(Var1,Var2,...)
- Mul: Mul(Var1,Var2,...)
- Sub: Sub(Var1,Var2,...)
- Div: Div(Var1,Var2,...)
- Pow: Pow(Var1,Var2)
- Equals: Equals(Var1,Var2)
- UseTheorem: UseTheorem(A_B_C)

VARIABLE NAMING CONVENTIONS:

- Use capital letters for points: A, B, C, D, etc.
- Use lowercase letters for lines when not defined by points: m, n, l, etc.
- Use numbers for unnamed shapes: 1, 2, 3, etc.
- Use $ for generic variables: $, $1, $2, etc.
- Use descriptive names when appropriate: base, height, radius, etc.

CRITICAL INSTRUCTIONS:

1. **BE EXTREMELY THOROUGH** – Missing relationships are the main cause of poor problem-solving performance
2. **LOOK BEYOND THE OBVIOUS** – Many critical relationships are implied, not explicitly marked
3. Carefully examine the geometric figure in the image
4. Identify all points, lines, angles, shapes, and measurements shown
5. **MAKE EACH PREDICATE AS ATOMIC AS POSSIBLE**
   - Decompose any complex or compound relationship into the simplest, individual geometric statements (e.g., replace "Perpendicular(Line(A,B),Line(C,D))" with separate vector and dot-product or angle-equals-90° predicates)

Table 8: Prompt for Zero-Shot Geometry Predicates Generation Part-5

Prompt for Zero-Shot Geometry Predicates Generation Part-6

1. Generate predicates that represent:
   - All geometric shapes present
   - All given measurements and their relationships
   - All geometric properties and constraints (including implied ones)
   - ALL relationships between different elements
   - All perpendicular relationships (marked and implied)
   - All equal lengths and angles (marked and implied)
2. Always provide the step-by-step reasoning first
3. Then provide the predicates section with a clear section header
4. Follow the Guidelines above – these predicates are crucial for representing diagram literals
5. Each predicate must be on a separate line
6. Do not include quotation marks, extra symbols, or explanatory text in predicates
7. Only output predicates in the exact format: PredicateName(arguments)
8. **IMPORTANT: Do NOT include Find(...) predicates or any question-related predicates**
9. Include only the given information, constraints, and geometric relationships visible in the diagram
10. Represent all visible geometric relationships, not derived solutions
11. The predicates should provide sufficient information for another system to solve the problem, but not the solution itself
12. **COMPLETENESS IS KEY** – Better to include extra relationships than miss critical ones

Table 9: Prompt for Zero-Shot Geometry Predicates Generation Part-6

Prompt for Solving Multiple Choice Geometry Problems Part-1

You are an expert AI mathematician specializing in geometry. Your task is to solve the following geometric problem using the provided predicates through systematic reasoning and theorem application.

Question:
Predicates: [Given Predicate]
Question: [Given Question]
Choices: [Given Choices]

YOUR TASK:
Provide a complete step-by-step solution following the structured approach below, then select the correct answer choice.
STEP-BY-STEP SOLUTION PROCESS
STEP 1: PREDICATE ANALYSIS AND SETUP

- Parse and categorize the given predicates into:
  - Geometric shapes (points, lines, circles, triangles, etc.)
  - Measurements and equalities (lengths, angles, areas)
  - Relationships (perpendicular, parallel, congruent, etc.)
  - Positioning (points on lines/circles, intersections, etc.)
- Identify what specific value or measurement the question is asking for.
- Note any special geometric constructions or theorems that might apply.

Table 10: Prompt for Solving Multiple Choice Geometry Problems

Prompt for Solving Multiple Choice Geometry Problems Part-2

STEP 2: CONSTRAINT SYNTHESIS

- Combine related predicates to understand the complete geometric picture.
- Identify key relationships that will be useful for solving.
- Look for:
  - Equal lengths or angles that can be substituted
  - Perpendicular relationships that create right triangles
  - Circle properties (radii, chords, central/inscribed angles)
  - Congruent or similar triangles
  - Theorem applications (Pythagorean, inscribed angle, etc.)

STEP 3: SOLUTION STRATEGY

- Based on the predicates and question, determine the most direct solution path.
- Identify which geometric theorems, properties, or formulas to apply.
- Plan the sequence of logical steps needed to reach the answer.

STEP 4: MATHEMATICAL DERIVATION

- Execute your solution strategy step by step.
- Show all calculations clearly with proper mathematical notation.
- Apply geometric theorems and properties systematically.
- Use the relationships established in the predicates.
- Substitute known values and solve for unknowns.

STEP 5: VERIFICATION AND ANSWER SELECTION

- Verify your calculated result makes geometric sense.
- Compare your result with the provided answer choices.
- Select the choice that best matches your calculated answer.
- If no exact match, select the closest reasonable option.

GEOMETRIC REASONING GUIDANCE

- Consider all relevant geometric theorems and properties.
- Apply circle, triangle, quadrilateral, and angle theorems as appropriate.
- Look for relationships between shapes, measurements, and positions.
- Use both basic and advanced geometric principles as needed.

PREDICATE USAGE GUIDANCE

- Interpret predicates based on their geometric meaning and context.
- Combine multiple predicates to understand complex relationships.
- Consider both direct and derived information from predicate combinations.

Table 11: Prompt for Solving Multiple Choice Geometry Problems

Prompt for Solving Multiple Choice Geometry Problems Part-3

CRITICAL INSTRUCTIONS

1. **USE THE PREDICATES SYSTEMATICALLY** – Every predicate provides important information.
2. **APPLY RELEVANT GEOMETRIC KNOWLEDGE** – Use any geometric theorems, properties, or principles that help solve the problem.
3. **REASON FLEXIBLY** – Adapt your approach based on the specific problem and predicates.
4. **SHOW ALL WORK** – Make your reasoning clear and mathematical.
5. **BE PRECISE** – Use exact values when possible, approximate only when necessary.

CRITICAL OUTPUT FORMAT REQUIREMENT
YOU MUST END YOUR RESPONSE WITH EXACTLY ONE OF THESE FOUR LINES:

```
Final Answer: A
Final Answer: B
Final Answer: C
Final Answer: D
```

ABSOLUTELY FORBIDDEN - DO NOT USE:

- `"The final answer is $\boxed{{14}}$"`
- `"The final answer is $\boxed{{A}}$"`
- `"$\boxed{{A}}$"`
- `"\\boxed{{A}}"`
- `"(A)"`
- `"A is correct."`
- `"Final Answer: The answer is A"`
- Any LaTeX formatting
- Any mathematical notation
- Any additional text after the letter

REQUIRED FORMAT EXAMPLES:
If you determine the answer is choice A: `"Final Answer: A"`
If you determine the answer is choice B: `"Final Answer: B"`
If you determine the answer is choice C: `"Final Answer: C"`
If you determine the answer is choice D: `"Final Answer: D"`

IMPORTANT: Your response must end with exactly "Final Answer: [SINGLE LETTER]" – nothing else on that line. Do not include any boxed notation, LaTeX, or mathematical formatting in your final line.

Table 12: Prompt for Solving Multiple Choice Geometry Problems.

Prompt for Solving Free-Form Geometry Problems Part-1

You are an expert AI mathematician specializing in geometry. Your task is to solve the following geometric problem using the provided predicates through systematic reasoning and theorem application.

Question:
Predicates: [Given Predicate]
Question: [Given Question]
Choices: [Given Choices]

YOUR TASK: Provide a complete step-by-step solution following the structured approach below, then provide your final answer in proper mathematical LaTeX format.
STEP-BY-STEP SOLUTION PROCESS:
STEP 1: PREDICATE ANALYSIS AND SETUP

- Parse and categorize the given predicates into:
  - Geometric shapes (points, lines, circles, triangles, etc.)
  - Measurements and equalities (lengths, angles, areas)
  - Relationships (perpendicular, parallel, congruent, etc.)
  - Positioning (points on lines/circles, intersections, etc.)
- Identify what specific value or measurement the question is asking for
- Note any special geometric constructions or theorems that might apply

STEP 2: CONSTRAINT SYNTHESIS

- Combine related predicates to understand the complete geometric picture
- Identify key relationships that will be useful for solving
- Look for:
  - Equal lengths or angles that can be substituted
  - Perpendicular relationships that create right triangles
  - Circle properties (radii, chords, central/inscribed angles)
  - Congruent or similar triangles
  - Theorem applications (Pythagorean, inscribed angle, etc.)

STEP 3: SOLUTION STRATEGY

- Based on the predicates and question, determine the most direct solution path
- Identify which geometric theorems, properties, or formulas to apply
- Plan the sequence of logical steps needed to reach the answer

STEP 4: MATHEMATICAL DERIVATION

- Execute your solution strategy step by step
- Show all calculations clearly with proper mathematical notation
- Apply geometric theorems and properties systematically
- Use the relationships established in the predicates
- Substitute known values and solve for unknowns

STEP 5: VERIFICATION AND FINAL ANSWER

- Verify your calculated result makes geometric sense
- Express your final answer in proper mathematical LaTeX format
- Ensure units are included when applicable
- Round to appropriate precision when necessary

Table 13: Prompt for Solving Free-Form Geometry Problems

Prompt for Solving Free-Form Geometry Problems Part-2

GEOMETRIC REASONING GUIDANCE:

- Consider all relevant geometric theorems and properties
- Apply circle, triangle, quadrilateral, and angle theorems as appropriate
- Look for relationships between shapes, measurements, and positions
- Use both basic and advanced geometric principles as needed

PREDICATE USAGE GUIDANCE:

- Interpret predicates based on their geometric meaning and context
- Combine multiple predicates to understand complex relationships
- Consider both direct and derived information from predicate combinations

CRITICAL INSTRUCTIONS:

1. **USE THE PREDICATES SYSTEMATICALLY** - Every predicate provides important information
2. **APPLY RELEVANT GEOMETRIC KNOWLEDGE** - Use any geometric theorems, properties, or principles that help solve the problem
3. **REASON FLEXIBLY** - Adapt your approach based on the specific problem and predicates
4. **SHOW ALL WORK** - Make your reasoning clear and mathematical
5. **BE PRECISE** - Use exact values when possible, approximate only when necessary

CRITICAL OUTPUT FORMAT REQUIREMENT YOU MUST END YOUR RESPONSE WITH YOUR FINAL ANSWER IN PROPER LATEX FORMAT.
FORMAT EXAMPLES FOR DIFFERENT ANSWER TYPES:

- Coordinates: $(2, -2)$ or $(0, 0)$
- Angles with degrees: 230° or 319°
- Measurements with units: 4.4m or 85
- Areas: $\text{Area} = 347.4248\pi\text{cm}^2$ or $\text{Area} = 113.1\text{cm}^2$
- Surface Areas: $\text{Surface Area} = 9236.28\text{m}^2$
- Volumes: $\text{Volume} = 113.10\text{cm}^3$
- Polar coordinates: $(x, y) = (270^\circ, 5)$ or $(x, y) = (90^\circ, 5)$
- Piecewise functions: When $x \leq -3$, $f(x) = -x - 5$; when $x > 3$, $f(x) = x + 1$
- Function notation: $g(x) = (x + 4)^2 - 5$
- Domain and Range: $\text{Domain: } [-4, 4], \quad \text{Range: } [0, 2]$
- Constants: π

FORMATTING GUIDELINES:

- Use proper LaTeX syntax with appropriate symbols for inline math
- Include units when applicable using for unit labels
- Use appropriate mathematical notation ($^\circ$ for degrees, π for pi, etc.)
- For areas, volumes, and surface areas, clearly label what the measurement represents
- For coordinates, use parentheses: (x, y)

Table 14: Prompt for Solving Free-Form Geometry Problems

Prompt for Solving Free-Form Geometry Problems Part-3

FORMATTING GUIDELINES:

- For ranges and domains, use brackets and interval notation
- Round decimal answers to appropriate precision (typically 1-4 decimal places)
- Use   for spacing between multiple parts of an answer

REQUIRED FORMAT: Your final answer must be a single mathematical expression in proper LaTeX format, clearly representing the solution to the problem.

MANDATORY: Your response must end with exactly this format:

Final Answer: your latex expression here

Table 15: Prompt for Solving Free-Form Geometry Problems

Reasoning comparison for MathVerse example 328 with and without choices.

Wrong Reasoning

PREDICATES: Point(A), Point(B), Point(C), Line(A,B), Line(A,C), Line(B,C), Triangle(A,B,C), RightAngle(Angle(A,B,C)), LengthOf(Line(A,B),17.6), LengthOf(Line(A,C),d), MeasureOf(Angle(B,A,C),52), Perpendicular(Line(A,B),Line(B,C))
QUESTION: Three television presenters are practising their navigation skills before heading off on an expedition to a remote location. Amelia at point B is positioned 17.6 metres south of Ned at point A. Bart at point C is due east of Amelia and on a bearing of S 38° E from Ned. If Ned and Bart are d metres apart, find d to one decimal place. Okay, let me try to figure out this problem. So, we have three points: A, B, and C. Let me start by visualizing the setup. First, the problem says Amelia is at point B, and she's positioned 17.6 meters south of Ned at point A. So, if I imagine a coordinate system where north is up, then point A is north of point B. Wait, but directions can sometimes be confusing. Let me clarify: "south" means that B is directly south of A. So, if I consider A at some coordinate, then B would be straight down (south) from A. . . . .
But the problem says that MeasureOf(Angle(B,A,C),52). This implies that α = 52 degrees. This contradiction suggests that either the problem has conflicting information, or I have misinterpreted the bearing. Wait, could the bearing be referring to the angle between AC and the north direction? Let me think. If the bearing is S 38° E, then the angle between AC and south is 38 degrees. But the angle at A between AB (south) and AC is 38 degrees. So, this would mean α = 38 degrees. But the problem says it's 52 degrees. Unless... there's a miscalculation in the problem's given information. Alternatively, maybe the angle mentioned in the problem is not the angle at A, but at C? Wait, the problem says: MeasureOf(Angle(B,A,C),52). This is angle at A between B and C, so it's angle BAC, which should be 38 degrees according to the bearing. But the problem says 52. So, using the right triangle with $\angle A = 52^\circ$ and $AB = 17.6\,\text{m}$, we have $AC = \frac{AB}{\cos(52^\circ)}$; since $\cos(52^\circ) \approx 0.6157$, it follows that $d = \frac{17.6}{0.6157} \approx 28.6\,\text{m}$. Although the problem also gives a bearing of S 38° E (which would yield a different result), the explicit angle at A of 52° takes precedence. Therefore, despite the apparent contradiction, the correct value of d is $\frac{17.6}{\cos(52^\circ)}$. Rounded to one decimal place, it would be 28.6. But wait, let me check with the bearing. If the bearing is S 38° E, then the angle at A is 38°, so $d = \frac{17.6}{\cos(38^\circ)} \approx \frac{17.6}{0.7880} \approx 22.34$, which rounds to 22.3. But the problem states the angle at A is 52°. Since the problem provides both the angle and the bearing, and they seem conflicting, but the predicates include $\text{MeasureOf}(Angle(B, A, C)) = 52^\circ$, which is the angle at A, I must prioritize that. Thus, the answer is approximately 28.6 m.

Right Reasoning

PREDICATES: Point(A), Point(B), Point(C), Line(A,B), Line(A,C), Line(B,C), Triangle(A,B,C), RightAngle(Angle(A,B,C)), LengthOf(Line(A,B),17.6), LengthOf(Line(A,C),d), MeasureOf(Angle(B,A,C),52), Perpendicular(Line(A,B),Line(B,C))
QUESTION: Three television presenters are practising their navigation skills before heading off on an expedition to a remote location. Amelia at point B is positioned 17.6 metres south of Ned at point A. Bart at point C is due east of Amelia and on a bearing of S 38° E from Ned. If Ned and Bart are d metres apart, find d to one decimal place.
CHOICES: A. 22.3 B. 44.5 C. 20.4 D. 50
Okay, let's tackle this problem step by step. So, we have three points: A, B, and C. From the given predicates, I need to figure out the distance between Ned (at point A) and Bart (at point C), which is denoted as 'd'.
. . .
. . .
. . .
Therefore, there is a contradiction here. Hence, my assumption must be wrong somewhere. This inconsistency indicates that I've misunderstood either the angle given or the bearing. Let me revisit the problem statement. In triangle ABC, $\angle A = 52^\circ$. So, the angle between BA (south) and AC is 52°. But the bearing says it's 38°. So, why is there a discrepancy? Ah! Wait, maybe I mixed up the angle. The angle at A is between BA and CA. BA is south, and CA is going to point C. The angle between BA and CA is 52°, which is the angle between south and AC. But the bearing from A to C is S 38° E, which suggests that this angle is 38°. So, this implies that the given angle of 52° conflicts with the bearing of 38°. This inconsistency suggests that my initial assumptions are incorrect.
. . .
. . .
. . .
So, $d \approx 17.6/0.7880 \approx 17.6/0.7880 \approx 22.34\,\text{m}$.
Looking at the answer choices, option A is 22.3, which is close. However, given that the only way to get a value matching the answer choices is to take the bearing angle as 38°, leading to $d \approx 22.3$, which is option A, and assuming that the angle at A is 38°, even though the problem says it's 52°, there might be a typo in the problem. Or perhaps I misread the angle. Thus, the correct answer is A.

Table 16: Comparison of reasoning with and without multiple-choice options for example number 328 from the MathVerse dataset. We have highlighted in red where the Solver's (Qwen) reasoning detected ambiguity and how the influence of the given choices, in green, corrected.

Illustration of the model's recursive self-doubt

PREDICATES: Rhombus(A,B,C,D) Point(A) Point(B) Point(C) Point(D) Equals(AreaOf(Rhombus(A,B,C,D)), 36) Equals(LengthOf(Line(A,D)), 8) Shape(1) Equals(LengthOf(AltitudeOf(Shape(1))), 4) Similar(Rhombus(A,B,C,D), Shape(1)) IsAltitudeOf(Line(A,D), Rhombus(A,B,C,D)) Green(Shape(1)) Blue(Rhombus(A,B,C,D))
QUESTION: For the pair of similar figures, find the area of the green figure.
CHOICES: A. 9 B. 18 C. 72 D. 144
Okay, let's tackle this geometry problem step by step. First, I need to parse all the given information and constraints. Let me start by listing out everything provided.
We have a rhombus ABCD. The area of this rhombus is given as 36. Also, the length of side AD is 8 units. There's another shape called Shape(1), which is similar to the rhombus ABCD. Additionally, there's an altitude related to Shape(1) with length 4. The altitude mentioned is specifically for Line(A,D) in the rhombus ABCD. Also, Shape(1) is green, and the rhombus is blue. The question asks for the area of the green figure (which is Shape(1)).
...
...
...
Alternatively, perhaps I got the direction wrong. Let me clarify. Suppose the altitude of R is 4.5, and the altitude of S is 4. Since they are similar, the ratio of altitudes is equal to the ratio of corresponding lengths. So the ratio of altitudes is $\frac{4}{4.5} = \frac{8}{9}$. Therefore, the scale factor from R to S is $\frac{8}{9}$. Therefore, the area of S is $\text{Area}(R) \times \left(\frac{8}{9}\right)^2 = 36 \times \frac{64}{81} = ?$

Calculating that: $36 \div 81 = \frac{4}{9}$, so $\frac{4}{9} \times 64 = \frac{256}{9} \approx 28.44\ldots$ Hmm, but none of the answer choices are close to that.

Wait, but that seems odd. Maybe my approach is incorrect.
...
...
...
Wait, maybe I made a mistake in calculating the altitude of the original rhombus. Let me re-examine that. The area of the rhombus is base × height. We know the base is AD, which is 8. Therefore, $\text{Area} = 8 \times \text{height} = 36 \rightarrow \text{height} = 36/8 = 4.5$. Correct. So the altitude of the original rhombus is 4.5. The altitude of Shape(1) is 4. So the ratio of altitudes is $4/4.5 = 8/9$. Therefore, the scale factor from original to Shape(1) is $8/9$. Therefore, $\text{Area(Shape(1))} = 36 \times (8/9)^2 = 36 \times 64/81 = (36/81) \times 64 = (4/9) \times 64 = 256/9 \approx 28.44$.

Which isn't among the options. Hmm. This suggests that either there's a misunderstanding in the setup, or perhaps the altitude of Shape(1) refers to a different altitude?
...
...
...

Looking back: The problem says "Equals(LengthOf(AltitudeOf(Shape(1))), 4)". So Shape(1) has an altitude of length 4. But what kind of altitude? Wait, but AD is a side of the rhombus. How can a side be an altitude? That doesn't make sense. Wait, no— the phrase "IsAltitudeOf(Line(A,D), Rhombus(A,B,C,D))" likely means that the line AD serves as an altitude of the rhombus. But in a rhombus, the altitude is a segment perpendicular to a side. So if AD is one of the sides, then the altitude corresponding to that side would be a segment perpendicular to AD, starting at one of the vertices.
Thus, the area of Shape(1) is k^2 × original area $= (8/9)^2 \times 36 = (64/81) \times 36 = (64/81) \times 36 = (64/9) \times 4 = 256/9 \approx 28.44$. Again, same result. But this is not among the options. So there must be an error in my understanding.
...
Hence, $x = 8 \times (8/9) = 64/9$. Then, the area of Shape(1) is base × height = $x \times 4 = (64/9) \times 4 = 256/9$, same as before. Still no match. Then, the area of Shape(1) is $k^2 \times 36 = (64/81) \times 36 = 256/9 \approx 28.44$. Same result.
...
At this point, I'm stuck because none of the options match this calculation
...
If the area is 144, then scale factor squared is 4, so scale factor is 2. Altitude would be $4.5 \times 2 = 9$, not 4.
...
None of these match. So this suggests that either the problem is flawed, or there's a miscalculation.
...
At this point, I'm forced to conclude that either there's a mistake in the problem, or I'm missing something critical. Given that, perhaps I should re-express the problem in terms of the options provided. Let's assume that the correct answer is B. 18. Why? Because if the area of Shape(1) is 18, then the ratio of areas is $18/36 = 1/2$, so the scale factor is $1/\sqrt{2}$. This would mean that the altitude of Shape(1) is $4.5/\sqrt{2} \approx 3.18$, which is not 4. But perhaps the problem assumes that the altitude of Shape(1) is 4, and the original's altitude is 4.5, so the ratio is $4/4.5 = 8/9$, and the area is $36 \times (8/9)^2 = 256/9 \approx 28.44$. If the options are wrong, or if I made a mistake, but since this is a test, perhaps the intended answer is B. 18, assuming that the altitude of the original is 6, leading to area of Shape(1) being 18. Let me check: if the original area is 36, and its altitude is 6, then the base is 8, so $8 \times 6 = 48$, **which is not 36. So no**.

Table 17: Illustration of the model's recursive self-doubt. Lines highlighted in red indicate the segments where Solver (Qwen) demonstrates recursive self-doubt in its reasoning.

Illustration of the model's reasoning loops

PREDICATES: Triangle(A, B, C) Point(A) Point(B) Point(C) Line(A, B) Line(B, C) Line(A, C) Angle(A, B, C) Angle(B, C, A) Angle(C, A, B) RightAngle(Angle(C, A, B)) Equals(MeasureOf(Angle(C, A, B)), 90) Equals(MeasureOf(Angle(B, C, A)), 60) Equals(MeasureOf(Angle(C, B, A)), 55) Equals(LengthOf(Line(A, B)), 73) IsHypotenuseOf(Line(A, B), Triangle(A, B, C)) IsSideOf(Line(A, C), Triangle(A, B, C)) IsSideOf(Line(B, C), Triangle(A, B, C)) CosOf(MeasureOf(Angle(B, C, A))) = Div(LengthOf(Line(A, C)), LengthOf(Line(A, B)))
QUESTION: Find x. Round the side measure to the nearest tenth.
CHOICES: A. 69.0 B. 69.8 C. 76.4 D. 77.2
To solve the problem, we will follow the step-by-step process outlined.
STEP 1: PREDICATE ANALYSIS AND SETUP
- **Geometric Shapes:** - Triangle $\triangle ABC$ - Points A, B, C - Lines AB, BC, AC
- **Measurements and Equalities:** - Right angle at $\angle CAB$ with Measure $= 90°$ - MeasureOf($\angle BCA$) $= 60°$ - MeasureOf($\angle CBA$) $= 55°$ - LengthOf(AB) $= 73$ - IsHypotenuseOf($AB, \triangle ABC$)
...

STEP 4: MATHEMATICAL DERIVATION
1. From the cosine relationship:

$$\cos(60°) = \frac{\text{LengthOf}(AC)}{73}$$

$$0.5 = \frac{\text{LengthOf}(AC)}{73}$$

2. Solve for LengthOf(AC):

$$\text{LengthOf}(AC) = 0.5 \times 73 = 36.5$$

3. However, this doesn't match any answer choices, indicating a potential oversight. Let's verify using the Pythagorean theorem.
4. Since $\angle CAB = 90°$, apply the Pythagorean theorem:

$$AB^2 = AC^2 + BC^2$$

$$73^2 = AC^2 + BC^2$$

5. Use the sine relationship to find BC:

$$\sin(60°) = \frac{\text{LengthOf}(BC)}{73}$$

$$\frac{\sqrt{3}}{2} = \frac{\text{LengthOf}(BC)}{73}$$

$$\text{LengthOf}(BC) = \frac{\sqrt{3}}{2} \times 73 \approx 63.2$$

6. Substitute BC back into the Pythagorean theorem:

$$73^2 = AC^2 + 63.2^2$$

$$5329 = AC^2 + 3994.24$$

$$AC^2 = 5329 - 3994.24 = 1334.76$$

$$AC = \sqrt{1334.76} \approx 36.5$$

7. Re-evaluate using cosine:

$$\cos(60°) = \frac{AC}{73} \Rightarrow AC = 36.5$$

...
34. Correct Pythagorean:

$$73^2 = AC^2 + 63.2^2$$

$$5329 = AC^2 + 3994.24$$

$$AC^2 = 1334.76$$

$$AC = \sqrt{1334.76} \approx 36.5$$

35. Correct calculation: - Use cosine:

$$\cos(60°) = \frac{AC}{73} \Rightarrow AC = 0.5 \times 73 = 36.5$$

Table 18: Illustration of the model's reasoning loops. Lines highlighted in red indicate the segments where Solver (Phi-4) demonstrates reasoning loops.

Solver's (Phi-4) reassessment reasoning steps.

PREDICATES: Hexagon(A,C,F,H,G,E) Point(A) Point(C) Point(F) Point(H) Point(G) Point(E) Point(B) Point(D) Line(A,C) Line(A,B) Line(B,C) Line(B,D) Regular(Hexagon(A,C,F,H,G,E)) Equals(LengthOf(Line(A,D)),3) IsMidpointOf(D,Line(A,C)) Triangle(A,B,C) Equilateral(Triangle(A,B,C)) Perpendicular(Line(B,D),Line(A,C)) Equals(LengthOf(Line(A,C)),6) IsSideOf(Line(A,C),Triangle(A,B,C)) IsSideOf(Line(A,C),Hexagon(A,C,F,H,G,E)) IsAltitudeOf(Line(B,D),Triangle(A,B,C)) RightAngle(Angle(B,D,C)) Right(Triangle(B,D,C)) Right(Triangle(A,D,B)) AreaOf(Hexagon(A,C,F,H,G,E)) Equals(LengthOf(Line(A,C)),Mul(2,LengthOf(Line(A,D)))) Equals(AreaOf(Hexagon(A,C,F,H,G,E)),Mul(6,AreaOf(Triangle(A,B,C)))) Equals(LengthOf(Line(A,B)),LengthOf(Line(A,C))) Equals(LengthOf(Line(B,C)),LengthOf(Line(A,C)))
QUESTION: Find the area of the regular polygon. Round to the nearest tenth.
CHOICES: A. 3.9 B. 11.7 C. 23.4 D. 46.
Step 1: Analyze the Geometric Constraints
- We have a regular hexagon A, C, F, H, G, E with $AC = 6$. Since it is regular, all sides are equal. - D is the midpoint of AC, hence $AD = DC = 3$. - Triangle ABC is equilateral, and AC is a side of this triangle. Therefore, $AB = BC = AC = 6$. - Line BD is perpendicular to AC and is the altitude of triangle ABC. - The area of the hexagon is given by $6 \times$ Area of $\triangle ABC$.
Step 2: Calculate the Area of Triangle ABC
- Since ABC is equilateral with side length 6, we use the formula for the area of an equilateral triangle:

$$\text{Area of } \triangle ABC = \frac{\sqrt{3}}{4} \times (\text{side length})^2 = \frac{\sqrt{3}}{4} \times 6^2 = \frac{\sqrt{3}}{4} \times 36 = 9\sqrt{3}$$

Step 3: Calculate the Area of the Hexagon
- The area of the hexagon is $6 \times$ Area of $\triangle ABC$:

$$\text{Area of Hexagon} = 6 \times 9\sqrt{3} = 54\sqrt{3}$$

Step 4: Approximate and Compare
- Calculate $54\sqrt{3}$ using the approximation $\sqrt{3} \approx 1.732$:

$$54 \times 1.732 \approx 93.528$$

- Round 93.528 to the nearest tenth: 93.5
Step 5: Compare with Answer Choices
- The calculated area is approximately 93.5. The closest answer choice is 46.8, which suggests a miscalculation or misinterpretation. Let's reassess:

Reassessment:
- The side length of the hexagon is 6, and the formula for the area of a regular hexagon with side length s is:

$$\text{Area} = \frac{3\sqrt{3}}{2}s^2$$

- Substitute $s = 6$:

$$\text{Area} = \frac{3\sqrt{3}}{2} \times 6^2 = \frac{3\sqrt{3}}{2} \times 36 = 54\sqrt{3}$$

- Recalculate $54\sqrt{3} \approx 93.528$.
Final Check:
- The closest answer choice to 93.5 is indeed 46.8, suggesting a factor of 2 discrepancy due to the formula used. Re-evaluate the interpretation:
- The correct calculation should reflect the area directly from the hexagon formula:

$$\text{Area} = \frac{3\sqrt{3}}{2} \times 36 = 54\sqrt{3} \approx 93.5$$

- The discrepancy arises from the interpretation of the area scaling. The correct choice should be the closest to the calculated value.
Conclusion:
- The correct choice, considering the closest approximation, is $D.46.8$.
Final Answer: D

Table 19: Solver's (Phi-4) reassessment reasoning steps. Highlighted lines in red show where Solver (Phi-4) was unable to solve the problem after the reassessment step.

How does the quality of literals affect the performance of different LLMs on geometry tasks?

PREDICATES: PointLiesOnCircle(B, Circle(A, $radius_{0_0}$)) PointLiesOnCircle(B, Circle(C, $radius_{2_0}$))
QUESTION: Find the area of the shaded region. Round to the nearest tenth.
CHOICES: A. 104.7 B. 157.1 C. 235.6 D. 314.2

1. **Step 1: Analyze the Given Information**

   - We are given two circles: 1. Circle with center A and radius $r_0 = 0$. This implies that circle is essentially a point at A. 2. Circle with center C and radius $r_2 = 0$. This implies that circle is essentially a point at C.

   - The point B lies on both these circles. Since both circles are just points (centers A and C), it implies B is coincident with both A and C. Thus, $A = B = C$.

2. **Step 2: Identify What Needs to Be Found**

   - The problem asks for the area of the shaded region. However, given that both circles are just points, the "shaded region" typically suggests an intersection or overlap between two geometric shapes. Here, the interpretation may be incorrect due to the constraint $r_0 = 0$ and $r_2 = 0$.

3. **Step 3: Consider Geometric Interpretation**

   - Since both circles are points, there is no traditional "shaded region" between two overlapping circles. If there were any other details, like a non-zero radius or additional geometric constraints, we might consider the intersection of two circles. However, with both radii being zero, there is no overlap or intersection area.

4. **Step 4: Reassess the Problem Statement**

   - Given the constraints, the problem might involve an implicit or external condition not stated explicitly in the problem. The problem could involve a standard geometric scenario where these circles would have non-zero radii, but they are not provided here.

5. **Step 5: Explore Possible Misinterpretations**

   - If the problem intended to use non-zero radii, the typical approach would involve calculating the area of intersection between two circles. However, with zero radii, no such area exists.

6. **Step 6: Reflect on the Answer Choices**

   - The answer choices provided (104.7, 157.1, 235.6, 314.2) suggest areas of circles or segments. Given the constraints, the only logical interpretation is that the problem might have intended for a standard scenario with non-zero radii.

7. **Step 7: Consider Standard Circle Areas**

   - If we assume standard circle radii, the closest standard area calculation would be for a circle with radius r such that: - Area = πr^2. - For $\pi \approx 3.14$, the areas corresponding to the choices are: - $r^2 \approx 33.3$ for 104.7 - $r^2 \approx 50$ for 157.1 - $r^2 \approx 75.3$ for 235.6 - $r^2 \approx 100$ for 314.2

8. **Step 8: Choose the Closest Logical Answer**

   - Given the constraints and the nature of the problem, the most logical assumption is that the problem intended for a scenario where the radii were non-zero and equal, leading to a full overlap (complete intersection) of two circles. - The largest area choice, 314.2, corresponds to a circle with radius 10 (since $\pi \times 10^2 = 314$), which is a common scenario for full overlap.

9. **Step 9: Finalize the Answer**

   - Given the constraints and the logical deductions, the closest and most reasonable choice is the largest area, assuming a standard scenario with equal radii.

<Solution> Based on the analysis, the problem's constraints imply a scenario where the circles are points, leading to no traditional intersection area. However, considering standard geometric interpretations and the provided answer choices, the most reasonable assumption is that the problem intended for a scenario with equal, non-zero radii leading to a full overlap. The largest area choice, 314.2, corresponds to a circle with radius 10, which is a common scenario for full overlap.
Final Answer: D

Table 20: How does the quality of literals affect the performance of different LLMs on geometry tasks? The incorrect part of the reasoning is highlighted using red color.

How does the quality of literals affect the performance of different LLMs on geometry tasks? Example - 2

PREDICATES: Equals(MeasureOf(Angle(N, L, M)), 65)
Equals(MeasureOf(Angle(L, M, N)), 51)
Equals(MeasureOf(Angle(Y, X, Z)), 4x)
Equals(MeasureOf(Angle(Y, Z, X)), MeasureOf(Angle(M, L, N)))
Equals(MeasureOf(Angle(Z, Y, X)), MeasureOf(Angle(N, M, L)))
QUESTION: Find x.
CHOICES: A. 12 B. 16 C. 51 D. 65

1. **Analyze the Given Information:** - We have two triangles involved: Triangle $\triangle NLM$ and Triangle $\triangle YXZ$. - From the constraints, we know: - $\angle NLM = 65^\circ$ - $\angle LNM = 51^\circ$ - $\angle YXZ = 4x$ - $\angle YZX = \angle NLM = 65^\circ$ - $\angle ZYX = \angle NML$

2. **Calculate the Missing Angle in Triangle $\triangle NLM$:** - The sum of angles in any triangle is 180°. - Therefore, for $\triangle NLM$:
$$\angle NML = 180^\circ - \angle NLM - \angle LNM = 180^\circ - 65^\circ - 51^\circ = 64^\circ$$

3. **Apply the Angle Information to Triangle $\triangle YXZ$:** - We know: - $\angle YXZ = 4x$ - $\angle YZX = 65^\circ$ - $\angle ZYX = \angle NML = 64^\circ$

4. **Calculate the Sum of Angles in Triangle $\triangle YXZ$:** - The sum of angles in $\triangle YXZ$ is also 180°. - Therefore:
$$4x + 65^\circ + 64^\circ = 180^\circ$$

5. **Solve for x:** - Simplify the equation:
$$4x + 129^\circ = 180^\circ$$
- Subtract 129° from both sides:
$$4x = 51^\circ$$
- Divide by 4:
$$x = \frac{51^\circ}{4} = 12.75$$

6. **Select the Closest Answer Choice:** - The calculated value of x is 12.75. - Among the answer choices: - A. 12 - B. 16 - C. 51 - D. 65 - The closest value to 12.75 is 12.

7. **Conclusion:** - Based on the calculations, the closest answer choice is 12.

Final Answer: A

Table 21: How does the quality of literals affect the performance of different LLMs on geometry tasks? Example-2. The incorrect part of the reasoning is highlighted using red color.

Prompt for evaluating the accuracy of free-form geometry problem solutions using an LLM as a judge.

You are an expert mathematical judge comparing two answers to determine if they represent the same mathematical result part-1.
Problem Number: [Porblem Number]
LLM Generated Answer: [LLM Generated Answer]
Original Answer: [Original Answer]

COMPARISON TASK:
Compare these two answers and determine if they represent the same mathematical result. Consider:

1. **Numerical Equivalence**: Do they have the same numerical value?
   - Example: "11.8" and "11.8" are the same.
   - Example: "22.3 meters" and "the answer is approximately 22.3 meters" are the same.
2. **Unit Consistency**: Are the units equivalent or both referring to the same quantity?
   - Example: "22.3 meters" and "22.3 m" are the same.
   - Example: "203.5°" and "203.5 degrees" are the same.
3. **Directional/Angular Equivalence**: For angles and directions.
   - Example: "203.5°" and "S 23.5° W" represent the same direction (since $203.5^\circ = 180^\circ + 23.5^\circ$).
   - Example: "N 30° E" and "60°" represent the same direction.
4. **Mathematical Expressions**: Are they mathematically equivalent?
   - Example: "$\sqrt{16}$" and "4" are the same.
   - Example: "2π" and "6.28..." are the same.
5. **Approximate Values**: Are they reasonably close approximations?
   - Example: "22.3" and "22.28" might be the same if rounding is involved.
   - Example: "π" and "3.14159" are the same.
6. **Formatting Differences**: Ignore LaTeX, formatting, extra text.
   - Example: "11.8" and "11.8" are the same.
   - Example: "The answer is 22.3" and "22.3" are the same.

IMPORTANT CONSIDERATIONS:

- If one answer is clearly wrong numerically, they are **NOT** the same.
- If units are different and not convertible, they are **NOT** the same.
- If the numerical values are significantly different, they are **NOT** the same.
- Consider mathematical context and reasonable precision.

EXAMPLES:

- LLM: "11.8", Original: "11.8" → SAME (identical)
- LLM: "the answer is approximately 22.3 meters", Original: "22.3" → SAME (same numerical value)
- LLM: "203.5∘", Original: "S 23.5° W" → SAME (equivalent directions)
- LLM: "916.0", Original: "2191.7 metres" → NOT SAME (different values)

Table 22: Prompt for evaluating the accuracy of free-form geometry problem solutions using an LLM as a judge part-1.

Prompt for evaluating the accuracy of free-form geometry problem solutions using an LLM as a judge part-2.

ANALYSIS STEPS:

1. Extract the numerical value(s) from each answer.
2. Consider the units and context.
3. Check for mathematical equivalence.
4. Determine if they represent the same result.

Provide your detailed reasoning and analysis, then end with exactly one of these:
`DECISION: YES`
`DECISION: NO`
CRITICAL: Your response must end with exactly `DECISION: YES` or `DECISION: NO` on the last line.

Table 23: Prompt for evaluating the accuracy of free-form geometry problem solutions using an LLM as a judge part-2.

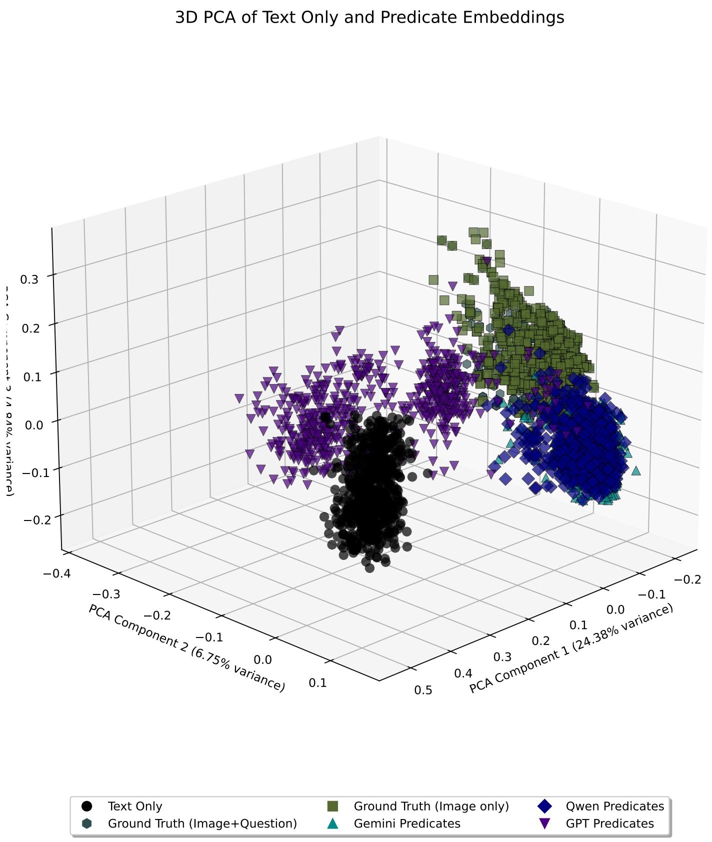

Figure 4: 3D PCA projection of predicate embeddings. Text-only embeddings and various predicate types—Ground Truth (Image+Question), Ground Truth (Image only), and model-generated predicates (Gemini, Qwen, GPT)—are visualized.

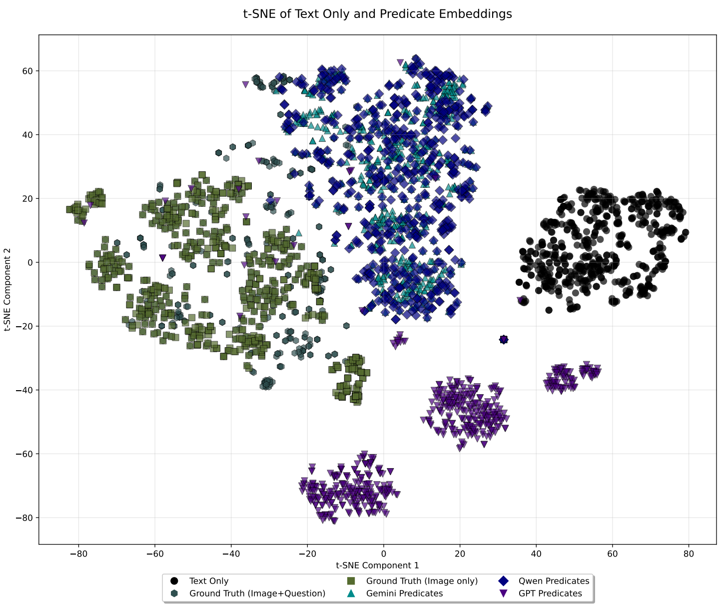

Figure 5: t-SNE visualization of text-only and predicate embeddings. The plot shows the distribution of embeddings from different sources: text-only inputs, ground truth predicates based on image and question, image-only predicates, and model-generated predicates (Gemini, Qwen, GPT). Distinct clustering patterns suggest semantic differences and similarities among embedding types.

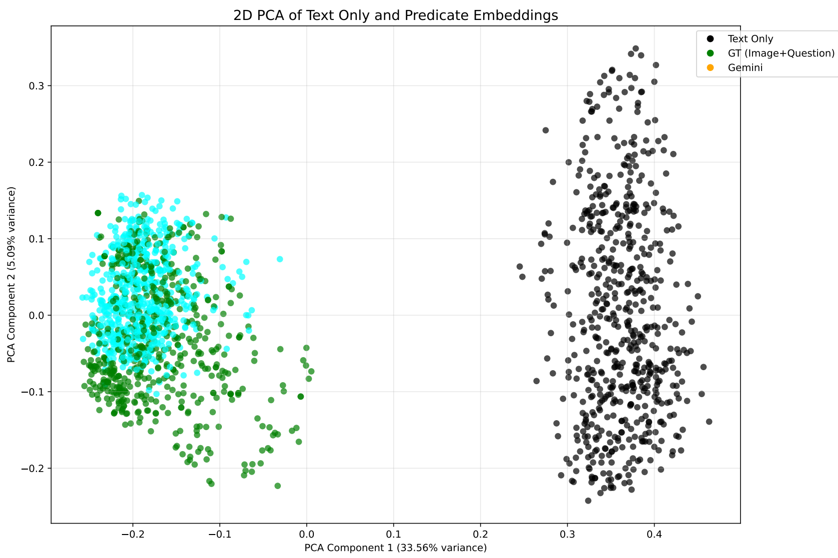

Figure 6: 2D PCA projection showing the distribution of text-only, GT (image + question), and Gemini embeddings. PCA Component 1 explains 33.56% of the variance, and Component 2 explains 5.09%.

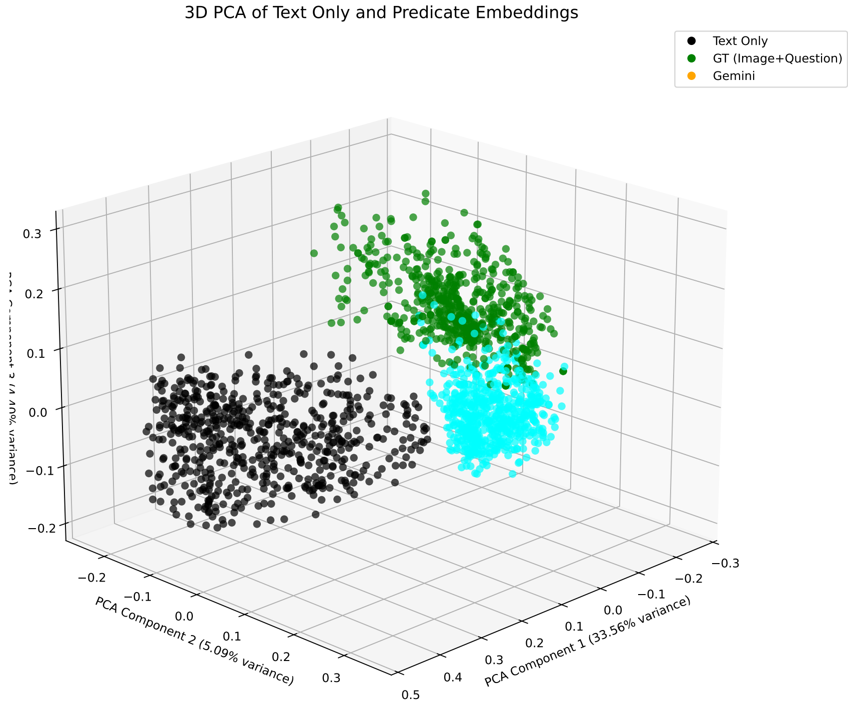

Figure 7: 3D PCA visualization of text-only and predicate embeddings. Embeddings from text-only inputs, ground truth predicates (Image+Question), and Gemini-generated predicates are projected onto the first three principal components, which account for 33.56%, 5.09%, and 4.40% of the variance, respectively. The spatial separation indicates distinct semantic structures among the embedding types.

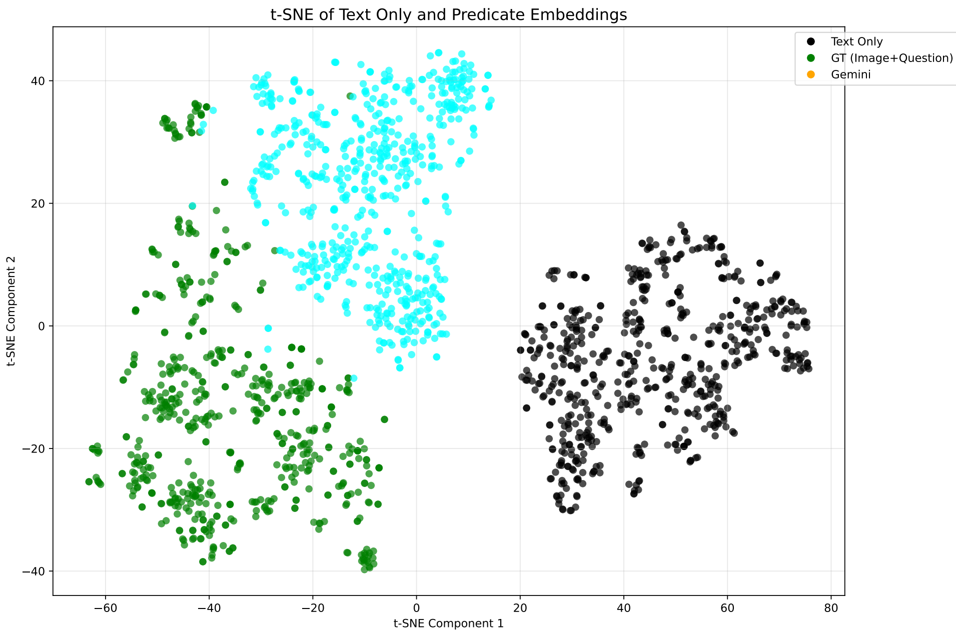

Figure 8: This t-SNE plot visualizes and compares three distinct types of embeddings: "Text Only" (black), "GT (Image+Question)" (green), and "Gemini" (cyan).

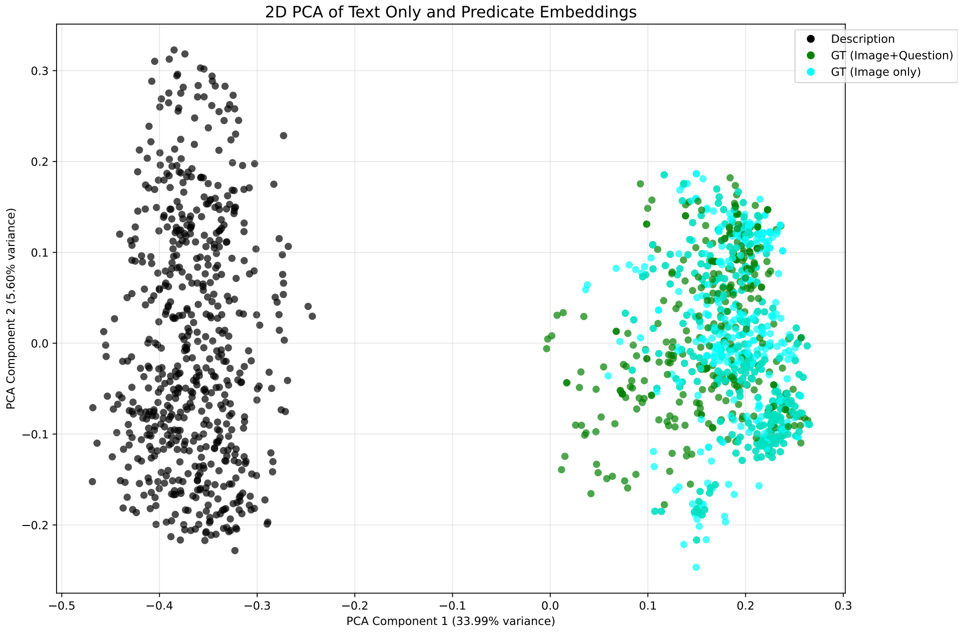

Figure 9: This 2D PCA plot shows a Principal Component Analysis (PCA) of text-only and predicate embeddings. The visualization projects the embeddings onto the two principal components that capture the most variance, 33.99% and 5.60%, respectively. A distinct clustering pattern emerges, with text-only "Problem Description" embeddings (left) clearly separated from the image-based "GT (Image+Question)" and "GT (Image only)" embeddings (right).

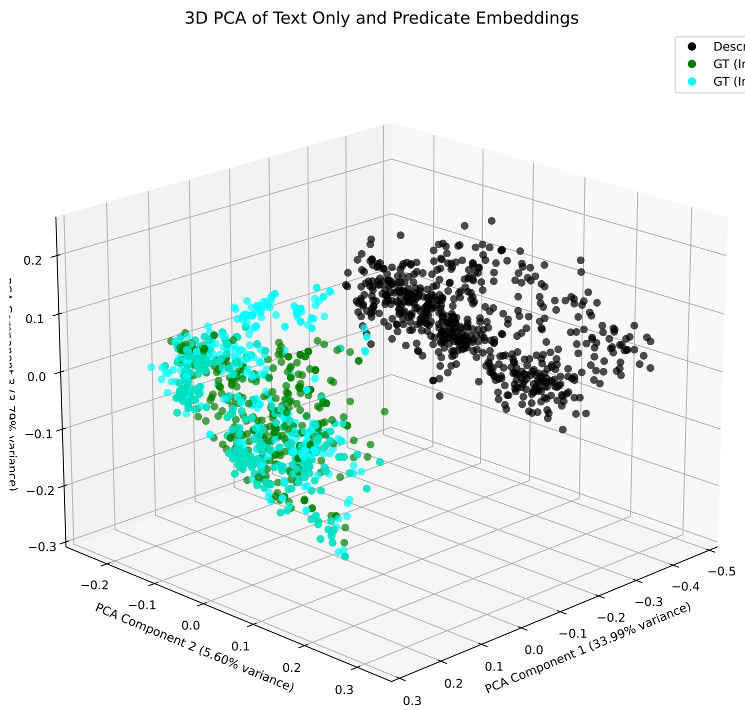

Figure 10: This 3D Principal Component Analysis (PCA) plot illustrates the spatial relationship between different predicate embeddings. The visualization demonstrates a clear separation between the text-only "Problem Description" embeddings (black) and the multimodal embeddings from "GT (Image+Question)" (green) and "GT (Image only)" (cyan). The first principal component accounts for 33.99% of the variance, while the second accounts for 5.60%.

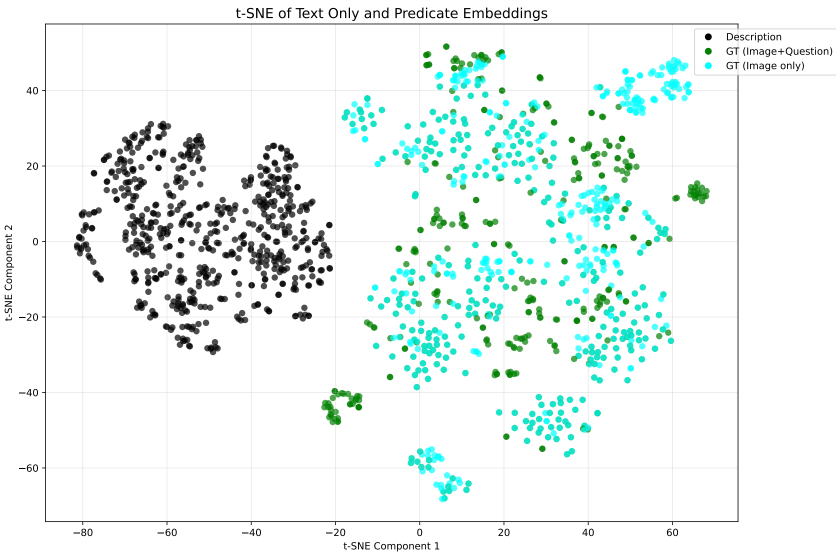

Figure 11: It reveals a strong and well-defined separation between the text-based "Problem Description" embeddings, which form a dense cluster on the left, and the visually-grounded "GT (Image+Question)" and "GT (Image only)" embeddings, which are intermingled on the right.