Beyond Lines and Circles: Unveiling the Geometric Reasoning Gap in Large Language Models

Spyridon Mouselinos
University of Warsaw
s.mouselinos@uw.edu.pl

Mateusz Malinowski[*]
Moonvalley AI
mateusz@moonvalley.ai

Henryk Michalewski[*]
Google DeepMind
University of Warsaw
henrykm@google.com

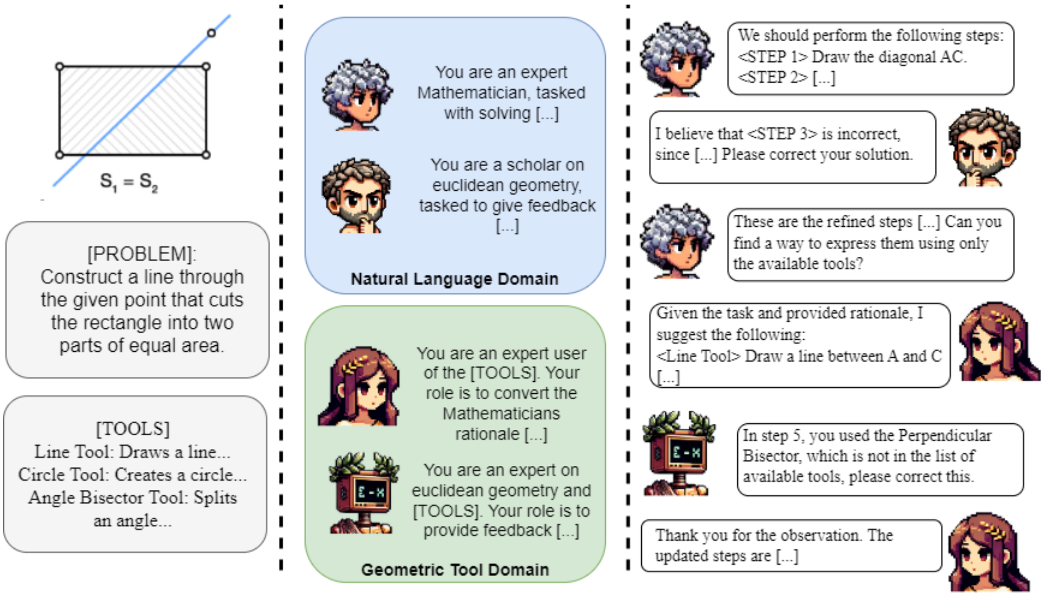

$S_1 = S_2$
[PROBLEM]: Construct a line through the given point that cuts the rectangle into two parts of equal area.
[TOOLS]
Line Tool: Draws a line...
Circle Tool: Creates a circle...
Angle Bisector Tool: Splits an angle...
You are an expert Mathematician, tasked with solving [...]
You are a scholar on euclidean geometry, tasked to give feedback [...]
Natural Language Domain
You are an expert user of the [TOOLS]. Your role is to convert the Mathematicians rationale [...]
You are an expert on euclidean geometry and [TOOLS]. Your role is to provide feedback [...]
Geometric Tool Domain
We should perform the following steps: <STEP 1> Draw the diagonal AC. <STEP 2> [...]
I believe that <STEP 3> is incorrect, since [...] Please correct your solution.
These are the refined steps [...] Can you find a way to express them using only the available tools?
Given the task and provided rationale, I suggest the following: <Line Tool> Draw a line between A and C [...]
In step 5, you used the Perpendicular Bisector, which is not in the list of available tools, please correct this.
Thank you for the observation. The updated steps are [...]

Figure 1—Drawing inspirations from the Ancient Greek Academy, we divide the reasoning pipeline into three stages. From left to right: The current geometric construction task is broken down into the image, its task description, and available tools. Our framework employs four LLM-based agents, each prompted with a specific role and task. A collaborative multi-round discussion is conducted where the geometric construction is effectively solved, reflecting the Academy's collective approach towards problem-solving and reasoning.

Abstract

Large Language Models (LLMs) demonstrate ever-increasing abilities in mathematical and algorithmic tasks, yet their geometric reasoning skills are underexplored. We investigate LLMs' abilities in constructive geometric problem-solving, – one of the most fundamental steps in developing human mathematical reasoning, revealing notable challenges in this domain. LLMs exhibit biases in variable names, struggle with 2D spatial relationships and planning, and hallucinate object placements. To this end, we introduce a framework that enhances LLMs' reasoning potential through a multi-agent system conducting internal dialogue. This work underscores LLMs' limitations in geometric reasoning and improves their capabilities through self-correction, collaboration, and diverse role specializations.

1 Introduction

Recent advancements in Large Language Models (LLMs) are groundbreaking, demonstrating increasing proficiency in complex mathematical and algorithmic tasks. Despite this, LLMs still face significant challenges in constructive geometry, a field fundamental to human mathematical reasoning involving tool usage, planning, and spatial reasoning.

Our investigation in this domain reveals several intriguing aspects. In instruction following, LLMs often exhibit a bias towards the style of the examples rather than focusing on the reasoning necessary for solving these problems. Furthermore, LLMs capable in maths do not necessarily show

[*] Equal Contribution

Findings of the Association for Computational Linguistics: EACL 2024, pages 6192–6222
November 12-16, 2024 ©2024 Association for Computational Linguistics

proficiency in geometrical problems, suggesting that algebraic reasoning does not directly translate to spatial and tool-based problem-solving. Another observation is that the choice of variable names in geometric constructions can affect the length and quality of the solutions, pointing to a potential bias where variable names carry unintended semantic weight. Moreover, despite being provided with visual aids, multimodal LLMs such as GPT4-V demonstrate difficulties interpreting 2D spatial relationships. They can identify objects in a scene but struggle to integrate them into a coherent plan involving tools or steps.

We propose a framework to overcome these challenges. Our solution includes appropriate renaming to mitigate naming biases and an adaptive prompt selection mechanism to focus the LLM on relevant information, avoiding overload. The model builds on past geometric tasks, enhancing its reasoning and context awareness. A critical factor in our approach is using simulacra-based conversational agents (Park et al., 2023) with specialized roles, some acting as reasoners while others as solvers or tool users. This cross-domain dialogue leverages the strengths of each agent type and fosters a more effective problem-solving approach than traditional role-playing methods.

Contributions. The main contributions of our work can be summarized in three points:
First, we are the first to provide an extensive analysis of the state-of-the-art leading LLMs' surprising difficulties in solving fundamental constructive geometric problems, highlighting a critical gap in their reasoning capabilities.
Second, we introduce three methods that assist LLMs in overcoming current limitations in the geometry domain. Our dynamic prompting mechanism builds on previous interactions instead of uninformative static prompts, our variable renaming technique neutralizes biases from variable name conventions, and our scene description prompt enhances LLMs' abilities to understand and manipulate spatial relationships.
Third, we present a novel simulacra-based system that leverages role and domain differentiation to integrate tool usage, instruction following, and collaborative problem-solving. This system surpasses non-collaborative methods and standard simulacra variants, demonstrating adaptability and promising results across various mathematical domains beyond geometry.

2 Related Work

Our approach is inspired by various research directions, briefly described here.

Prompt Engineering The advanced reasoning abilities of multi-billion-parameter LLMs (OpenAI, 2023; Google, 2023; Chowdhery et al., 2022; Brown et al., 2020) have transformed prompt engineering into sophisticated interactions for eliciting detailed responses. Works like (Wei et al., 2022; Kojima et al., 2022; Zhang et al., 2023b) use intermediate reasoning steps in prompts, improving performance in arithmetic and symbolic reasoning tasks. Diverging from hand-crafted prompts, (Reynolds and McDonell, 2021; Zhou et al., 2023c; Shin et al., 2020) propose automated prompt generation methods, exhibiting better results in reasoning tasks. In multi-agent scenarios, (Li et al., 2023) introduce Inception Prompting, enabling collaborative environments under role assignment. Recently, (Hao et al., 2023; Xie et al., 2023; Yao et al., 2023a) use tree-search and self-evaluation for exploration and strategic lookahead, while (Yao et al., 2023b) unifies planning and acting, prompting models to generate reasoning traces and actions.

Simulacra - Conversational Agents The concept of 'Agents' as entities exhibiting emergent intelligence through collective interaction was introduced by (Minsky, 1986). This idea has been extensively applied in reinforcement learning (Sukhbaatar et al., 2016; Havrylov and Titov, 2017; Dafoe et al., 2020; Bard et al., 2020; Sheng et al., 2020; Hosseini-Asl et al., 2020; Du et al., 2021). LLMs are considered potential agents due to their global knowledge and conversational skills (Huang et al., 2022; Andreas, 2022; Lo et al., 2023). (Park et al., 2022) demonstrated LLMs' effectiveness in complex social scenarios. Recent works (Li et al., 2023; Hong et al., 2023; Qian et al., 2023) systematize the concept of simulacra, providing frameworks for effective communication. (Wu et al., 2023; Chen et al., 2023; Lin et al., 2023; Zhou et al., 2023b) add functionalities like visualization and dynamic agent generation, while (Wang et al., 2023) introduce a benchmark for fine-grained role-playing, suggesting training on role-specific contexts.

Geometric problems While mathematics and algorithms remain predominant in reasoning challenges, the exploration of geometry has been limited. Key datasets (Seo et al., 2015; Chen et al., 2022; Zhang et al., 2023a; Lu et al., 2021) feature multiple-choice formats with annotated diagrams. Common approaches convert problems into relational sets in a domain-specific language (DSL) or as formal structural clauses, with reasoning executed by a symbolic solver or a DSL-trained model. The recent and parallel work, AlphaGeometry (Trinh et al., 2024), achieved impressive results on IMO-level geometry problems, using an LLM trained on synthetic DSL data to interface a theorem-proof engine where reasoning is delegated. That level of competence is possible as geometry is complete and decidable (Tarski, 1959). Contrary to that, we define all the necessary modules that generate and verify hypotheses using open-ended LLMs, showing how to improve their geometric abilities without changing their weights. In constructive geometry, tasks require planning, reasoning, and tool usage, drawing inspiration from (Macke et al., 2021; Wong et al., 2022). These works focus on Euclidea (Euclidea; PyEuclidea), a dataset with progressively challenging geometric problems. Our work proposes an alternative to symbolic solvers and tree-based search algorithms, enhancing LLMs' reasoning capabilities in this domain.

3 Preliminaries

In this section, we present the datasets, models, metrics, and definitions central to our framework.

Euclidea Our primary benchmark is the geometry game Euclidea (Euclidea), an online construction challenge with eight geometric tools and progressively complex problems. We use the Python version (PyEuclidea), which includes ninety-eight challenges across ten levels and a custom API for solution verification. We also compile a natural language version of the Euclidea dataset, including solutions from `https://euclidea.fandom.com/wiki/Euclidea_Wiki`, which we will make accessible for future research.

Euclid's Elements We train open-source LLMs on Euclid's Elements, the seminal work on geometry. It presents fundamental axioms and tools, progressively synthesizing more complex tools through constructions. Exposure to Elements aligns LLMs with geometry tasks, theoretically containing the knowledge to solve the challenges. We use the English translation of Euclid's Elements (Fitzpatrick, 2007-2008) found here.

Models In our setup, we examine the performance of seven LLMs. LLamaV2 (Touvron et al., 2023) shows an impressive performance in reasoning, maths, and coding. We test its 7B and 13B variants. Additionally, we include Mistral (Jiang et al., 2023) and its fine-tuned variant Zephyr (Tunstall et al., 2023), two 7B LLMs with performance comparable to larger checkpoints of other open-source LLMs. MetaMath (Yu et al., 2023) specializes in mathematical and algebraic reasoning, achieving state-of-the-art results on the Math (Hendrycks et al., 2021) and GSM8k (Cobbe et al., 2021) challenges among all open-source LLMs. We test two variants: the LlamaV2-13B and the Mistral-7B. We also include OpenAI's ChatGPT and GPT-4 for their superior performance in reasoning and problem-solving tasks. Finally, GPT-4's visual-language capabilities enable us to assess the role of visual inputs in solving geometric challenges.

Performance metrics In constructive geometry, multiple reasoning paths can lead to a correct result. In some cases, even reordering the steps of a solution without harming its correctness is possible. Instead of requiring an exact match to the ground truth, we use the pass@k metric (Kulal et al., 2019), which measures the existence of a correct completion among k independent generations. We adopt an updated unbiased version proposed in (Chen et al., 2021). We validate generated solutions using the Euclidea Python API, presenting the average of ten runs with different seeds. We choose sampling temperatures of 0.2 / 0.6 for pass@1 / pass@50 during generation, after hyperparameter search.

4 Method

This section introduces the components of our proposed framework, each addressing the limitations of LLMs in solving geometrical problems.

4.1 Prompting for Geometric Reasoning

For each geometric challenge, we prompt our LLMs with a description of the available tools, their expected operation, and task requirements. We employ a few-shot setup to enhance our models' accuracy and reduce erroneous interpretations of tool functionalities. Specifically, we maintain a memory bank of previously encountered problems and select the most relevant ones for each new task. This approach, called Adaptive-Shot, ensures consistent exposure to intricate problems and diverse tools, fostering nuanced and context-aware reasoning. Our mechanism employs a Sentence Transformer to compare the similarity between the current level's description and available tools and those of all other levels. After filtering out low-similarity candidates, the remaining examples are presented back to the model, which is tasked to identify the top five most valuable examples, integrating them into the final few-shot prompt.

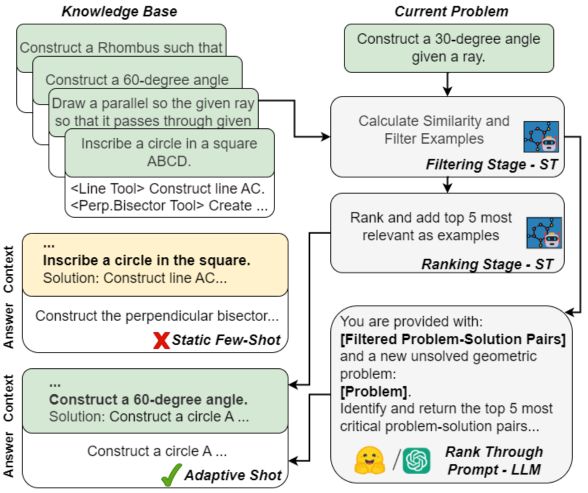

Knowledge Base
Construct a Rhombus such that
Construct a 60-degree angle
Draw a parallel so the given ray so that it passes through given
Inscribe a circle in a square ABCD.
<Line Tool> Construct line AC.
<Perp.Bisector Tool> Create ...
Current Problem
Construct a 30-degree angle given a ray.
Calculate Similarity and Filter Examples
Filtering Stage - ST
Rank and add top 5 most relevant as examples
Ranking Stage - ST
You are provided with:
[Filtered Problem-Solution Pairs] and a new unsolved geometric problem:
[Problem].
Identify and return the top 5 most critical problem-solution pairs...
Rank Through Prompt - LLM
Context
Answer
...
Inscribe a circle in the square.
Solution: Construct line AC...
Construct the perpendicular bisector...
Static Few-Shot
Context
Answer
...
Construct a 60-degree angle.
Solution: Construct a circle A ...
Construct a circle A ...
Adaptive Shot

Figure 2—The Adaptive few-shot mechanism: Given the problem ***Construct a 30-degree angle given a ray***, we initially filter our knowledge base for similar examples. Then we either rank and return the top five most similar results as our prompt - Adaptive-Shot (ST) or prompt the LLM to filter them out by itself - Adaptive-Shot (Self). Our proposed method guides the model by building upon similar or useful demonstrations, leading to increased performance.

Our method refines the model's understanding of geometric concepts and enhances its ability to effectively apply this knowledge to new and more complex problems. For further examples, we refer the reader to the Appendix Section D.

4.2 From Single Models To Simulacra

In the following stage, inspired by various studies (Li et al., 2023; Park et al., 2022), we employ a multi-agent setup, extending its application into new territories. Our methodology innovatively introduces agents differentiated not merely by their persona but also by their distinct functional roles within the problem-solving process, pioneering the disentanglement of reasoning from planning or tool usage. The first agent set, which we refer to as the natural language solver S_{NL}, generates rationales for approaching the problem in natural language. The geometric tool solver S_{GT} interprets these rationales and converts them to a series of steps using exclusively the available geometric tools.

The second set, called validators, is instrumental in assessing the proposed rationales and geometric tool steps, thus introducing a new layer of roles. Like solvers, validators receive domain-specific prompts, distinguishing them as natural language or geometric tool agents. However, unlike solvers who use the adaptive-shot mechanism, validators are prompted with propositions from Euclid's Elements and a static collection of incorrect examples alongside their rectifications. Depending on their domain, these are expressed in natural language or geometric tool steps. They engage in dialogue with solvers, providing feedback through up to five rounds of interaction. Validators approve or recommend modifications to the solver's steps, prompting refinements.

Figure 1 shows the interaction between agents: The natural language solver suggests a solution plan, refined by the natural language validator . This plan is then converted into tool steps by the geometric tool solver and validated by the geometric tool validator .

4.3 Enhancing Spatial Awareness

Building on the collaborative dynamic between solvers and validators, we identified a significant limitation in their ability to conceptualize spatial relationships in geometric problems. This issue manifests through actions like attempting to connect non-aligned points with a straight line or assuming unverified relationships between objects.

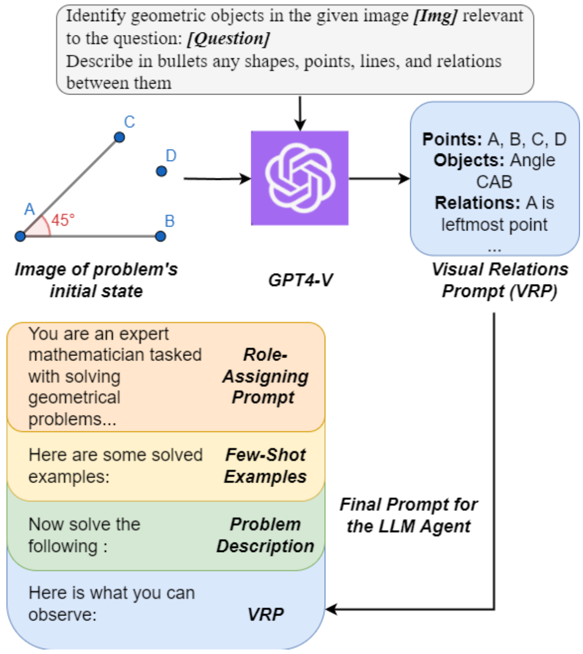

Identify geometric objects in the given image [Img] relevant to the question: [Question]
Describe in bullets any shapes, points, lines, and relations between them
C
D
A
45°
B
Image of problem's initial state
GPT4-V
Points: A, B, C, D
Objects: Angle CAB
Relations: A is leftmost point
...
Visual Relations Prompt (VRP)
You are an expert mathematician tasked with solving geometrical problems...
Role-Assigning Prompt
Here are some solved examples:
Few-Shot Examples
Now solve the following :
Problem Description
Here is what you can observe:
VRP
Final Prompt for the LLM Agent

Figure 3—VRP extraction using GPT-4V: An auxiliary prompt with the problem question and an initial state image is presented to GPT-4V, which returns a list of ***Points***, ***Objects***, and their ***Relations*** as bullet points. This information is then added to the overall agent prompt.

To address this, we introduce an auxiliary Vision-Language Large Model (VLLM), specifically GPT4-V, for its superior performance and ease of use to enhance scene comprehension. The VLLM is used not as the primary reasoner but as a scene analysis tool. It is prompted with an image-problem pair and asked to describe the geometric elements, their interrelations, and spatial orientations. This description, the Visual Relations Prompt (VRP), is added to each agent's prompt. The VRP disentangles spatial recognition from geometric problem-solving, reducing the need for recurrent interactions with visual extractors. It is both cost-efficient and flexible, enabling models without innate visual capabilities to utilize the VRP and enhance their decision-making abilities. Illustrated examples of the VRP can be found in Appendix Section G.

4.4 Mitigating Naming Biases

LLMs can adopt social biases from humans (Wallace et al., 2019; Liang et al., 2021), or be negatively affected by language bias in their reasoning process (Lin et al., 2020; Mouselinos et al., 2023). We observe a similar bias in the terminology used for geometric entities. For instance, when constructing a target named 'E' in contexts with 'A,' 'B,' and 'C,' models often create an intermediate 'D' before proceeding to 'E,' leading to unnecessary complexity. Likewise, choosing a target variable earlier in the alphabetical sequence than the required minimum steps to solve the problem can introduce faulty rationales (e.g., choosing 'C' as the target of a five-step solution can lead to early stopping on an intermediate generated 'C' point, abruptly ending the construction).

Thus, we propose a simple strategy to address this issue: substituting the target variable with 'X,' a universal symbol for unknowns in mathematics. This strategy encourages models to seek the most direct solutions, as demonstrated in Figure 4.

5 Experiments

We present a comprehensive performance analysis on the Euclidea dataset in Table 1. Our results encompass three testing setups:

In **Few-shot**, models are prompted with the task, tool descriptions, and five solved examples using geometric tools.

In **Finetuned**, all open-source models are finetuned using Euclid's "Elements" to acquire foundational knowledge. Aside from tutorial levels, Euclidea challenges do not directly overlap with "Elements" problems. We use the same setup for testing as in Few-shot.

Simulacra refers to our proposed multi-agent framework, equipped with an adaptive few-shot mechanism, visual relations prompting, and variable renaming.

Initially, all models performed modestly in our few-shot experimental setup, with noticeable improvements after fine-tuning. This outcome aligns with our expectation that familiarity with standard mathematical and reasoning scenarios does not ensure proficiency in constructive geometry tasks. We posit that fine-tuning with Euclid's "Elements" represents the upper limit of improvement achievable by open-source models, constrained by the dataset's size. MetaMath-Mistral 7B is the most promising among the open-source options, which we further examine with our multi-agent setup. Our results reveal a significant performance boost in larger models (ChatGPT / GPT4) and, notably, in MetaMath-Mistral 7B, under our proposed framework, surpassing the few-shot ChatGPT in performance. This finding underscores the adaptability and effectiveness of our approach across a spectrum of model sizes.

We also compare against two prior studies on the Euclidea dataset. The model by (Macke et al., 2021) combines a Masked-RCNN detector with an iterative deep search algorithm, using the Euclidea API to verify each step until a solution is

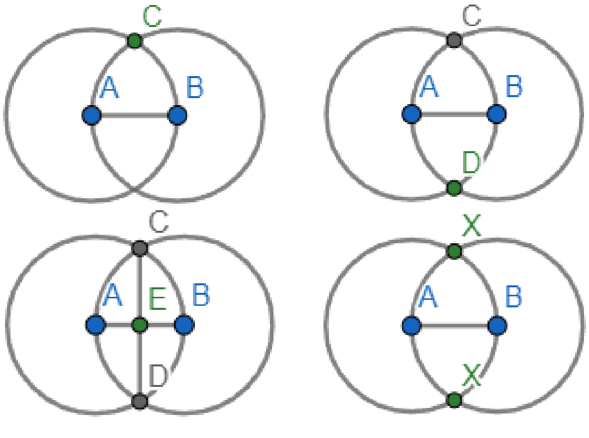


Figure 4—Visualized GPT-4 reasoning paths for the problem: *"Find a point [Target] that is equidistant from given points A and B."* Four illustrations depict different reasoning paths based on the target name. Naming the target **C** leads to a 3-step solution: draw circle A with radius AB, draw circle B with radius AB, and mark their intersection as C. Naming it **D** adds a fourth step, marking D as the final answer. Naming it **E** introduces variables C and D first, resulting in a 5-step solution with E on the perpendicular bisector of AB. For **X**, the model reverts to a 3-step process, marking any intersection arbitrarily and offering multiple solutions for X.

found within a pre-defined depth limit. Euclid-Net (Wong et al., 2022) uses "Deep visual reasoning with backtracking," where a neural network assists in step selection. These models rely on precise visual component extractors and real-time feedback during their trial-and-error process. To assess the possible benefits of real-time feedback, we introduce Feedback mode (+FB), where validators have access to ground truth answers instead of relying solely on their internal knowledge but are restricted to simply informing solvers about the correctness of their steps, avoiding any solution leak. Our findings show only a slight performance improvement, indicating that solvers struggle to develop complex solutions even with ground truth hints, highlighting the nuanced complexities of applying LLMs to constructive geometry.

Method	pass@1	pass@50
Few-Shot		
LlamaV2 (7B)	3.1 (± 0.5)	4.4 (± 0.7)
LlamaV2 (13B)	4.4 (± 0.4)	7.5 (± 0.8)
MetaMath (13B)	4.7 (± 0.4)	8.1 (± 0.8)
Mistral (7B)	5.1 (± 0.6)	8.7 (± 1.1)
Zephyr-α (7B)	5.3 (± 0.5)	8.9 (± 1.1)
MetaMath-Mistral (7B)	8.9 (± 0.7)	13.4 (± 1.2)
ChatGPT	11.7 (± 1.1)	18.6 (± 1.5)
GPT4	21.2 (± 1.3)	38.3 (± 1.4)
GPT4-V	22.8 (± 1.2)	38.5 (± 1.4)
Finetuned		
LlamaV2 (7B)	3.7 (± 0.6)	5.1 (± 0.7)
LlamaV2 (13B)	4.9 (± 0.4)	8.7 (± 0.8)
MetaMath (13B)	5.3 (± 0.5)	9.2 (± 1.0)
Mistral (7B)	6.9 (± 0.7)	9.7 (± 1.1)
Zephyr-α(7B)	6.6 (± 0.6)	9.5 (± 1.2)
MetaMath-Mistral(7B)	9.4 (± 0.9)	16.2 (± 1.3)
Ours - Simulacra		
MetaMath-Mistral (7B)	14.9 (± 1.3)	21.1 (± 1.7)
ChatGPT	32.3 (± 1.3)	61.4 (± 2.1)
GPT4	38.9 (± 1.1)	67.7 (± 2.2)
GPT4-V	37.1 (± 1.4)	65.9 (± 2.0)
FB + MetaMath-Mistral (7B)	15.1 (± 1.5)	21.4 (± 1.6)
FB + ChatGPT	35.6 (± 1.7)	63.5 (± 2.2)
FB + GPT4	41.2 (± 1.6)	71.2 (± 2.0)
FB + GPT4-V	40.3 (± 1.7)	70.6 (± 2.3)
Supervised Visual Component + Exhaustive Search		
Method		Accuracy
(Macke et al., 2021) (LOO-levels)		44.1
(Macke et al., 2021) (LOO-packs)		45.5
(Wong et al., 2022) Euclid-Net		75.5

Table 1: Results on Euclidea. Ours refers to the $S_{NL} - S_{GT}$ with VRP, variable renaming, and Adaptive Shot (Self). LOO stands for "Leave-One-Out": The model is either trained on the rest of the levels in the same pack (LOO-level) or the rest of the packs in the dataset (LOO-packs). GPT4-V refers to the multimodal use of GPT4 without the use of VRP in the Simulacra experiments.

6 Ablation Studies

This section presents ablation studies that underpin our model's development, as detailed in the methods section. These studies highlight the iterative refinement and integration of model elements, addressing limitations observed in LLMs during geometric problem-solving.

6.1 Hallucinations and Context Dependence

In our exploration, we initially tested LLMs in a zero-shot manner using only tool descriptions as context, which often resulted in hallucinated tool functionalities. To mitigate this, we transitioned to a few-shot setup, providing solved examples to demonstrate proper tool usage. This adjustment reduced the incidence of tool hallucinations as models benefited from clear demonstrations. However, a new problem emerged: models began replicating entire reasoning processes and were heavily influenced by the step sequences in the few-shot examples. We hypothesized that this behavior stemmed from the instruction-following training nature of LLMs, where they are predisposed to mimic styles and patterns seen in the prompts. In our case, models were not mimicking styles but inadvertently replicating entire reasoning processes.

Our adaptive shot method solved these challenges. Unlike random or static few-shot examples, our adaptive shot method ensures that the most valuable demonstrations are selected for the models at each step. This dynamic approach significantly reduced hallucinations and repetitive reasoning patterns. We also compared Adaptive-Shot (Self) with a simpler variant, Adaptive-Shot (ST), where the Sentence Transformer selected the examples after the filtering stage. The Adaptive-Shot (Self) method proved slightly more effective, albeit with the trade-off of more API calls than the Adaptive-Shot (ST), showcasing that our prompting method exceeds the Sentence Transformer's contextual abilities.

The results, as shown in Table 2, validate the performance benefits of our proposed method, demonstrating its superiority in addressing the limitations observed in earlier setups.

6.2 Effectiveness of Domain and Role Division

Interestingly, LLMs proved quite successful using geometric tools when prompted to generate ideas—not specific steps. Although they made convenient assumptions, their overall reasoning was still accurate. This led us to question the root cause of the discrepancy between knowing the solution as a plan in natural language and failing to execute it accurately using strict and abstract geometric tools.

We tested whether we benefit from those worlds' synergy: an LLM could focus on generating ideas in natural language, and another could specialize in transforming these ideas into a series of geometric tool steps. By employing such a multi-agent setup, as seen in Table 3, we observed a significant performance gain – 13.6 → 21.5 pass@1 – when comparing a single agent (S_{GT}) operating directly with geometric tools against a duo of collaborative agents with differentiated domains ($S_{NL} - S_{GT}$). The introduction of validators, the second set of agents in our framework, plays a crucial role in assessing the proposed rationales and geometric tool steps. Their continuous feedback loop with solvers, correcting superficial mistakes and preventing errors, leads to a further performance boost of 22.2 → 28.1 pass@1 in our proposed multi-agent configuration. Our approach of disentangling reasoning and execution facilitates efficient dialogue between agents. Each agent entirely focuses on its domain, transmitting only the necessary information to the next step while being receptive to feedback. This method enables more accurate and reliable problem-solving compared to traditional multi-agent techniques.

Method	pass@1	pass@50
Zero-Shot	5.9 (± 1.9)	9.6 (± 2.7)
Few-Shot (Tutorial)	7.2 (± 1.4)	15.9 (± 3.1)
Few-Shot (Alpha)	11.4 (± 1.8)	18.6 (± 3.2)
Few-Shot (Beta)	12.7 (± 2.2)	20.8 (± 3.6)
Adaptive-Shot (ST)	13.3 (± 1.8)	21.2 (± 3.0)
Adaptive-Shot (Self)	**13.6** (± 1.8)	**21.5** (± 2.7)

Table 2: Effectiveness of in-context examples: ChatGPT 3.5-Turbo on Alpha and Beta levels. Our adaptive method showcases its ability to employ useful examples from the already seen levels, leading to increased performance than static, hard-coded alternatives.

Configuration	Agents	Domains	pass@1	pass@50
S_{GT}	1	1	13.6 (± 1.8)	21.5 (± 2.7)
SV_{GT}	2	1	17.9 (± 1.2)	34.9 (± 2.3)
$S_{NL} - S_{GT}$	2	2	22.2 (± 1.3)	46.7 (± 2.0)
$SV_{NL} - SV_{GT}$	4	2	**28.1** (± 1.1)	**53.5** (± 2.4)

Table 3: ChatGPT 3.5-Turbo on Alpha and Beta levels. **S**: Solver, **V**: Validator, **NL**: Natural language, **GT** Geometric tools. Our proposed multi-agent setup with role and domain differentiation showcases significant performance gains against a typical multi-agent setup $S_{NL} - S_{GT}$ vs S_{GT}. This holds also true with the introduction of validators that further boost the overall performance of the framework $SV_{NL} - SV_{GT}$ vs SV_{GT}.

6.3 Visual Aids in Spatial Reasoning

Another noteworthy finding was that LLMs often created new geometric objects without acknowledging their overlap with existing ones. Moreover, they occasionally suggested steps that violated geometric rules or led to repetitive movements. We hypothesized that LLMs' difficulty with geometric reasoning in 2D spaces stems from a lack of exposure to such setups, typically operating in a unidimensional, left-to-right manner. This raised the question: Could introducing a visual signal bridge this reasoning gap?

We prompted GPT4-V with simple freehand sketches of geometric objects to explore this. The model successfully identified these, including subtle aspects like right angles indicated by small corner squares. In this way, we first established that the model can indeed understand geometric scenes. Continuing with a more complex test, we presented GPT4-V with an image/problem pair and asked for the first solution step. We then deliberately performed an incorrect step, drew it on the image, and presented it back to the model. The model often validated these erroneous steps in this setup, suggesting a disconnect between scene understanding and geometric reasoning.

The findings in Table 4 illustrate this observation: Comparing GPT4 with its multimodal variant, GPT4-V, revealed a marginal improvement, suggesting that visual signals assist during initial scene understanding. Building on these insights, we compared our proposed VRP method, VRP-GPT4, to the multimodal approach GPT4-V and observed an additional performance boost. Here lies the key advantage of our VRP method: Not only does it match, if not slightly surpass, the effectiveness of the multimodal approach, but it enables the transfer of scene understanding benefits to non-visually capable models like ChatGPT, which significantly improved when enhanced with VRP.

6.4 Impact of Geometry Nomenclature

LLMs mirror the human convention of alphabetical naming in mathematical contexts. The choice of target variables later in the alphabetical order leads to longer and more inaccurate solutions. This is a byproduct of their training on human-generated texts, where entities in algebraic or geometric contexts typically adhere to an alphabetical naming convention -labeling a triangle as ABC rather than EOA. Similarly, variables associated with the assignment of solutions, like X, would theoretically condition the LLM to find its value, possibly assigning it to any given constructed object. We designed an experiment with 20 geometric problems to empirically validate this hypothesis, each requiring 3 to 5 solution steps. In these problems, we manipulated the target variable in three distinct ways: maintaining the original name (+0), replacing it with the letter X (+X), and renaming it using a letter 1, 2, or 3 positions further in the alphabetical sequence (+1, +2, +3). As depicted in Table 5, models perform worse when the target variable is shifted by one or two letters in the alphabet. Interestingly, this tendency diminishes when the target is more than three letters away and is further reduced with the substitution of 'X,' underscoring its effectiveness as a neutral, bias-mitigating variable.

Domain	Method	pass@1	pass@50
Language	ChatGPT	13.6 (± 1.8)	21.5 (± 2.7)
	GPT4	23.9 (± 0.9)	44.8 (± 1.6)
	ChatGPT*	28.1 (± 1.2)	53.5 (± 1.7)
	GPT4*	33.7 (± 1.0)	62.0 (± 1.3)
Multimodal	GPT4-V	24.2 (± 1.4)	45.1 (± 1.7)
	VRP-ChatGPT	19.4 (± 1.1)	37.1 (± 1.3)
	VRP-ChatGPT*	34.5 (± 0.8)	59.2 (± 1.2)
	VRP-GPT4	25.3 (± 0.9)	48.6 (± 1.4)
	VRP-GPT4*	**38.8** (± 0.9)	**64.6** (± 1.2)

Table 4: Multimodal prompt effectiveness. Experiments on the Alpha and Beta pack levels. The ($\star$) symbol refers to an $SV_{NL} - SV_{GT}$ multi-agent configuration.

	ChatGPT	ChatGPT*	VRP-ChatGPT*
+0	10.7 (± 2.1)	48.1 (± 2.3)	60.7 (± 2.9)
+1	10.1 (± 2.4)	46.3 (± 2.1)	57.9 (± 2.5)
+2	10.1 (± 2.4)	47.5 (± 1.9)	57.9 (± 3.0)
+3	10.6 (± 1.2)	47.9 (± 1.7)	59.2 (± 2.3)
+X	10.5 (± 1.7)	48.6 (± 1.9)	61.1 (± 2.2)
	GPT4	GPT4*	VRP-GPT4*
+0	35.2 (± 2.5)	52.9 (± 2.1)	65.1 (± 2.2)
+1	32.8 (± 2.6)	50.4 (± 2.2)	63.4 (± 3.2)
+2	31.6 (± 2.8)	50.7 (± 2.7)	63.6 (± 2.9)
+3	32.4 (± 1.1)	51.8 (± 1.4)	64.7 (± 2.1)
+X	37.5 (± 1.3)	53.3 (± 1.5)	66.2 (± 2.4)

Table 5: Qualitative results on the effect of variable renaming. Results refer to the pass@50 metric. The star symbol ($\star$) refers to $SV_{NL} - SV_{GT}$ configuration.

6.5 Generalisation to different datasets

While our focus has been primarily on geometric problems, our framework is designed with adaptability in mind. Transitioning to a different toolset from the Euclidea tools, currently represented in our framework as text-based Name: Operation lists, requires minimal adjustment. This flexible design can theoretically introduce any new toolset by simply integrating it into the domain-specific agents' persona prompts with slight modifications to align with their unique capabilities. To this end, we tested its performance against three datasets involving mathematical reasoning: GSM8K, SVAMP, and the Geometry split from the MATH dataset. The toolset consisted of all basic math operations (+, -, *, /) and a Python interpreter tool that could sequentially execute any steps suggested before its call. We were inspired by (Zhao et al., 2023), who identified diverse reasoning patterns with Chain of Thought (COT) (Wei et al., 2022) versus Program-Aid (PAL) (Gao et al., 2023) methodologies. While COT is recognized for its creativity and flexibility in devising solutions, PAL is noted for its enhanced accuracy in numerical computations. This differentiation of domains was an ideal fit for our framework: A pair of Solver-Validator agents (SV_{NL}) initially planned the solution of a math problem in the natural language domain. The produced rationale is passed to another pair (SV_{MPT}), which utilizes the set of tools to formulate precise solutions. Finally, we employed the Adaptive Few-Shot mechanism with examples from each dataset's training split. Results in Table 6 show our multi-agent setup performing close to state-of-the-art methods despite not being primarily designed for these tasks. The MetaMath-Mistral model also saw significant performance boosts using our method, even in non-geometric setups.

Method	Model	GSM8K	SVAMP	Geometry
Ours	MM-Mist	84.3	79.7	21.6
	ChatGPT	88.4	86.1	40.2
	GPT4	96.9	95.8	56.3
MetaMath (Yu et al., 2023)	MM-Mist	77.7	75.8	18.4
MS (SC, K=15)	ChatGPT	89.2	85.2	N/A
(Zhao et al., 2023)	GPT4	96.8	95.8	N/A
PHP (SC, K=40)	ChatGPT	85.1	83.1	25.4
(Zheng et al., 2023)	GPT4	95.5	91.9	41.9
CSV (K=1 / K=16) (Zhou et al., 2023a)	GPT4-Code	92.9 / 97.0	N/A	54.0 / 64.9

Table 6: Results of the multi-agent framework on mathematical datasets. MM-Mist: MetaMath-Mistral 7B. N/A refers to not reported results.

7 Conclusions

Our study highlights the challenges LLMs face in constructive geometry, including limited skill transfer from other mathematical domains, inadequacies of typical prompting techniques, and a lack of 2D spatial reasoning. We identify that existing LLMs struggle with geometric tasks without inductive solid biases like theorem provers (Trinh et al., 2024). Furthermore, our research demonstrates that a multi-agent system with role and domain specializations can effectively address geometric problems and perform equally well in other domains. We hope our work serves as a foundation for developing training systems that operate in multi-agent settings, which is crucial for domains requiring deep, specific, and accurate cognitive processing.

8 Limitations

We hope to attract more attention to this domain and look forward to seeing future work improving our multi-agent setup or shifting the training paradigm of newer-generation LLMs to foster deeper reasoning in mathematical and geometric domains. However, we are aware of the potential risk associated with our work, specifically the cost of experiments involving closed-source LLMs behind APIs. These costs primarily stem from extensive communication rounds and large context sizes, which can lead to substantial expenses. Researchers planning to replicate or extend our work must consider this, as it could significantly impact their research budgets and timelines. Furthermore, we acknowledge the possible limitations of applying our method to domains where planning, tool usage, and/or validation might not be critical. Simpler methods could be proven cheaper and better in these domains.

Finally, we do not identify any ethical considerations associated with our proposed ideas and suggested methods or any possible malicious or unintended harmful uses.

References

Jacob Andreas. 2022. Language models as agent models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5769–5779, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. 2020. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F. Karlsson, Jie Fu, and Yemin Shi. 2023. Autoagents: A framework for automatic agent generation. *Preprint*, arXiv:2309.17288.

Jiaqi Chen, Tong Li, Jinghui Qin, Pan Lu, Liang Lin, Chongyu Chen, and Xiaodan Liang. 2022. Unigeo: Unifying geometry logical reasoning via reformulating mathematical expression. In *The 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

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

Allan Dafoe, Edward Hughes, Yoram Bachrach, Tantum Collins, Kevin R. McKee, Joel Z. Leibo, Kate Larson, and Thore Graepel. 2020. Open problems in cooperative ai. *Preprint*, arXiv:2012.08630.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Yali Du, Bo Liu, Vincent Moens, Ziqi Liu, Zhicheng Ren, Jun Wang, Xu Chen, and Haifeng Zhang. 2021. Learning correlated communication topology in multi-agent reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, page 456–464, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Euclidea. Euclidea game.

Richard Fitzpatrick. 2007-2008. *Euclid's Elements of Geometry*. I.L. Heiberg, Ed. & Trans.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10764–10799. PMLR.

Google. 2023. Bard.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *ArXiv*, abs/2305.14992.

Serhii Havrylov and Ivan Titov. 2017. Emergence of language with multi-agent games: Learning to communicate with sequence of symbols. In *5th International Conference on Learning Representations (ICLR 17, workshop track)*. 5th International Conference on Learning Representations, ICLR 2017 ; Conference date: 24-04-2017 Through 26-04-2017.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. *Preprint*, arXiv:2308.00352.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191. Curran Associates, Inc.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022. Inner monologue: Embodied reasoning through planning with language models. *Preprint*, arXiv:2207.05608.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *ArXiv*, abs/2205.11916.

Sumith Kulal, Panupong Pasupat, Kartik Chandra, Mina Lee, Oded Padon, Alex Aiken, and Percy S Liang. 2019. Spoc: Search-based pseudocode to code. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*, pages 6565–6576. PMLR.

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. 2020. Birds have four legs?! NumerSense: Probing Numerical Commonsense Knowledge of Pre-Trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6862–6868, Online. Association for Computational Linguistics.

Jiaju Lin, Haoran Zhao, Aochi Zhang, Yiting Wu, Huqiuyue Ping, and Qin Chen. 2023. Agentsims: An open-source sandbox for large language model evaluation. *Preprint*, arXiv:2308.04026.

Yat Long Lo, Christian Schroeder de Witt, Samuel Sokota, Jakob Nicolaus Foerster, and Shimon Whiteson. 2023. Cheap talk discovery and utilization in multi-agent reinforcement learning. *Preprint*, arXiv:2303.10733.

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.

J. Macke, J. Sedlar, M. Olsak, J. Urban, and J. Sivic. 2021. Learning to solve geometric construction problems from images.

Marvin Minsky. 1986. *The Society of Mind*. Simon & Schuster, Inc., USA.

Spyridon Mouselinos, Mateusz Malinowski, and Henryk Michalewski. 2023. A simple, yet effective approach to finding biases in code generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11299–11329, Toronto, Canada. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22.

Joon Sung Park, Lindsay Popowski, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2022. Social simulacra: Creating populated prototypes for social computing systems.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

PyEuclidea. Pyeuclidea.

Chen Qian, Xin Cong, Wei Liu, Cheng Yang, Weize Chen, Yusheng Su, Yufan Dang, Jiahao Li, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *Preprint*, arXiv:2307.07924.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA. Association for Computing Machinery.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1476, Lisbon, Portugal. Association for Computational Linguistics.

Junjie Sheng, Xiangfeng Wang, Bo Jin, Junchi Yan, Wenhao Li, Tsung-Hui Chang, Jun Wang, and Hongyuan Zha. 2020. Learning structured communication for multi-agent reinforcement learning.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Alfred Tarski. 1959. What is elementary geometry? In *Studies in Logic and the Foundations of Mathematics*, volume 27, pages 16–29. Elsevier.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Trieu Trinh, Yuhuai Wu, Quoc Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment. *Preprint*, arXiv:2310.16944.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Zekun Moore Wang, Zhongyuan Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Man Zhang, Zhaoxiang Zhang, Wanli Ouyang, Ke Xu, Wenhu Chen, Jie Fu, and Junran Peng. 2023. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. *arXiv preprint arXiv: 2310.00746*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Man Fai Wong, Xintong Qi, and Chee Wei Tan. 2022. Euclidnet: Deep visual reasoning for constructible problems in geometry. *Adv. Artif. Intell. Mach. Learn.*, 3:839–853.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. Self-evaluation guided beam search for reasoning. *Preprint*, arXiv:2305.00633.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Ming-Liang Zhang, Fei Yin, and Cheng-Lin Liu. 2023a. A multi-modal neural geometric solver with textual clauses parsed from diagram. In *International Joint Conference on Artificial Intelligence*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023b. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Xu Zhao, Yuxi Xie, Kenji Kawaguchi, Junxian He, and Qizhe Xie. 2023. Automatic model selection with large language models for reasoning. *Preprint*, arXiv:2305.14333.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023a. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. *Preprint*, arXiv:2308.07921.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. 2023b. Agents: An open-source framework for autonomous language agents. *Preprint*, arXiv:2309.07870.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023c. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

A Information on Models and Datasets

Model Name	Link	LICENSE
LlamaV2 (Touvron et al., 2023)	`https://github.com/facebookresearch/llama`	LLAMA 2 COMMUNITY LICENSE AGREEMENT
MetaMath (Yu et al., 2023)	`https://github.com/meta-math/MetaMath`	Apache License 2.0
Mistral (Jiang et al., 2023)	`https://github.com/mistralai/mistral-src`	Apache License 2.0
Zephyr-α (Tunstall et al., 2023)	`https://huggingface.co/HuggingFaceH4/zephyr-7b-alpha`	MIT
MetaMath-Mistral (Yu et al., 2023)	`https://github.com/meta-math/MetaMath`	Apache License 2.0
ChatGPT (Ouyang et al., 2022)	OpenAI - API usage of *gpt-3.5-turbo-16k*	N/A
GPT4 (OpenAI, 2023)	OpenAI - API usage of *gpt-4-32k*	N/A
GPT4-V (OpenAI, 2023)	OpenAI - API usage of *gpt-4-vision-preview*	N/A

Table 7: URL and Licenses of used Large Language Models / APIs.

Dataset Name	Link	LICENSE
Euclid-Elements (Fitzpatrick, 2007-2008)	`https://farside.ph.utexas.edu/books/Euclid/Elements.pdf`	CC BY-NC-SA 4.0
Python Port of Euclidea (PyEuclidea)	`https://github.com/mirefek/py_euclidea`	MIT
Euclidea Wiki Page	`https://euclidea.fandom.com/wiki/Euclidea_Wiki`	CC BY-SA
GSM8k (Cobbe et al., 2021)	`https://github.com/openai/grade-school-math`	MIT
SVAMP (Patel et al., 2021)	`https://github.com/arkilpatel/SVAMP`	MIT
Geometry / MATH (Hendrycks et al., 2021)	`https://github.com/hendrycks/math/`	MIT

Table 8: URL and Licenses of used Datasets.

The licenses associated with the models and datasets used in this work are consistent with their intended use in academic and research contexts. Specifically, the LLAMA 2 Community License Agreement and the Apache License 2.0 are permissive licenses that allow for extensive use in research and academic settings, provided that proper attribution is given. Similarly, the MIT License, the CC BY-NC-SA 4.0, and the CC BY-SA also support usage in academic work, allowing for modification and distribution under certain conditions. The OpenAI API usage adheres to OpenAI's terms of service, which permits their application in research.

B Information on Experimental Setup

Our experimental setup consisted of 1x NVIDIA A100 GPU. Regarding the fine-tuning results on Euclid's Elements of Table 1, we trained all LLMs using the bitsandbytes library (`https://github.com/TimDettmers/bitsandbytes`) and 4-bit quantization with the QLoRA technique (Dettmers et al., 2023). The Euclid-Elements dataset we created consists of approximately 1 million characters, or equivalently 290k thousand ChatGPT / GPT4 tokens.

Below, the reader can find the full hyperparameter list:

Hyperparameter	**Value**
Training Epochs	3
Batch Size	32
Accumulation Steps	4
Learning Rate	2e-5
Warmup Ratio	0.03
Scheduler	Cosine
Max Gradient Norm	0.3
Weight Decay	0.001
Lora Alpha	16
Lora Dropout	0.1
Lora R	64
Use 4bit	True
BnB_4bit_compute_dtype	Float16
BnB_4bit_quant_type	NF4
Gradient Checkpointing	True
Optimizer	Paged_adamw_32bit

Table 9: Hyperparameters and their values

Regarding the training objective, we used the typical causal language modeling loss. Moreover, we used a validation split of 10% sampled uniformly across different book chapters. We monitored the validation loss every 500 steps as our metric for early stopping.

Regarding the API calls to OpenAI models, gpt3.5-turbo-16k was used for the ChatGPT experiments, gpt4-32k was used for the GPT4 experiments, and the extraction of Visual Relation Prompts, The endpoint called gpt4-vision-preview was used. Our API calls were subject to throttling limits, and waiting loops were introduced to avoid service interruptions. We conducted most of our ablation studies and early experiments with ChatGPT to avoid massive waiting times and reduce the high experimental cost. The total experiment time was approximately 500 hours, and our total costs were around 2000 USD.

Finally, regarding the choice of 0.2 and 0.6 temperature values for pass@1 and pass@50, we experimented with different combinations of values for solvers on the Alpha pack of the Euclidea dataset and a 100 problem subset of GSM8K. For pass@1, values between 0.2 and 0.3 worked best, while for pass@50, values between 0.6 and 0.8 also worked great. We finalized our decision on the values 0.2 and 0.6 for all solver experiments. The temperature of validator agents was 0.8 in all cases, based on the idea that we wanted more expressiveness and different reasoning pathways to be considered when validating a solution. We found that setting the temperature too low (0-0.2) led validators to accept more solutions verbatim, defying their purpose.

C Tools and problems from Euclidea dataset

There are 10 tools available in Euclidea, although not all of them can be used on every level. Tools become available progressively as the difficulty increases. Here is a list of all available tools:

1. **Move Tool**: Moves a geometric object.
2. **Point Tool**: Marks a point and labels it.
3. **Line Tool**: Draws a line between two points or a ray from a starting point.
4. **Circle Tool**: Constructs a circle using a specific point as the center.
5. **Perpendicular Bisector Tool**: Creates the perpendicular bisector of a segment between two points.
6. **Perpendicular Tool**: Draws a line perpendicular to a given one at a specific point.
7. **Angle Bisector Tool**: Creates a line that bisects a given angle.
8. **Parallel Tool**: Draws a line parallel to a given line or segment.
9. **Compass Tool**: Uses a compass to construct a circle with a radius equal to a given segment.
10. **Intersect Tool**: Marks the intersection between two geometric objects.

Here are some sample problems from different difficulty levels:

Given the rectangle ABCD with $AB > AD$. Inscribe a rhombus in the rectangle so that they share a diagonal.
Available Tools: [Line Tool, Circle Tool, Perpendicular Bisector Tool, Intersect Tool]
Solution:
Perpendicular Bisector Tool: Construct the perpendicular bisector of AC, intersecting AB at E and CD at F.
Line Tool: Construct line AF.
Line Tool: Construct line CE.

Problem: Rhombus in Rectangle - Pack: Alpha

Given the side AB. Construct a rhombus with the given side and an angle of 45° in a vertex.
Available Tools: [Line Tool, Circle Tool, Perpendicular Bisector Tool, Perpendicular Tool, Intersect Tool, Angle Bisector Tool]
Solution:
Perpendicular Tool: Construct the perpendicular to AB from A; let C be a distinct arbitrary point on that perpendicular.
Angle Bisector Tool: Construct the angle bisector of BAC.
Circle Tool: Construct the circle with center A and radius AB, intersecting the angle bisector at D and line AC at E with E on the side of A opposite from C.
Perpendicular Tool: Construct the perpendicular to AC through D.
Line Tool: Construct line BE.

Problem: Lozenge - Pack: Gamma

Let A be the vertex. Construct two rays that divide the given angle of 54 degrees into three equal parts.
Available Tools: [Line Tool, Circle Tool, Perpendicular Bisector Tool, Intersect Tool, Parallel Tool, Compass Tool]
Solution:
Circle Tool: Construct a circle with center B on either line and radius AB. Circle B intersects with line AB at point C, and the other line at point D.
Circle Tool: Construct a circle with center D and radius AD. Circle D intersects with circle B at point E.
Line Tool: Draw line AE. This is the first angle trisector.
Circle Tool: Draw circle with center E and radius EC. Circle E intersects with circle B at point F.
Line Tool: Draw line AF. This is the second angle trisector.

Problem: Angle of 54 - Pack: Theta

D Static Versus Adaptive Few-Shot

In this section, we will provide a more detailed overview of our adaptive few-shot mechanism.

Initially, we collect a set of previously solved problems alongside their solutions, which we refer to as our knowledge base. This set can be acquired in multiple ways: In the case of our GSM8K / SVAMP / Geometry-Math experiments of Table 6, we use all problems belonging to the provided training split.

In the case of the Euclidea experiments of Table 1, we begin with a small set of five handcrafted geometric challenges. Euclidea's problems vary in difficulty and are grouped into increasingly difficult level packs. Since we are not provided with a training split, we add all problems belonging to previously seen packs to our knowledge base during the solution of a level. In this way, we continuously increase the size of our base and the availability of more diverse and complex techniques that our agents can utilize.

The second stage involves using a Sentence Transformer to reduce the size of our knowledge base. For this, we compare the cosine similarity scores of our current problem and the problems in our knowledge base and keep entries with scores over 0.5 or the top fifteen, whichever leads to fewer examples.

The third and final stage involves using the Sentence Transformer again, which chooses the top five most similar examples to build the final few-shot prompt. This procedure is referred to as Adaptive-Shot (ST) in our experiments. An alternative to this is to use the examples from the second step and prompt our solver-LLM to filter the top five most relevant examples by itself. We call this alternative Adaptive-Shot (Self). Note here that the second step of the procedure can be skipped during Adaptive-Shot (ST) since instead of filtering, we can directly return the top five results. However, it is necessary for the Adaptive-Shot (Self) method, where we need to respect the context size limits of the tested LLMs.

Below, we present an example of three different few-shot prompts: A static-handcrafted version, an adaptive prompt compiled by a Sentence Transformer (Adaptive ST), and finally, an adaptive prompt filtered by the model itself (Adaptive Self). Our model is ChatGPT, and the problem at hand is: *Construct the circumcircle of a given triangle ABC.*

Construct the circumcircle of a given triangle ABC.

Problem

[Line Tool, Circle Tool, Intersection Tool, Perpendicular Bisector Tool]

Available Tools

[...]
Example 1:
Description: Construct an equilateral triangle given one side. Let AB be the given side.
Tool List: [Line Tool, Circle Tool, Intersection Tool, Perpendicular Bisector Tool]
Solution:
Circle Tool: Construct a circle with center A and radius AB.
Circle Tool: Construct a circle with center B and radius AB, intersecting circle A at point C.
Line Tool: Construct lines AC and BC. Triangle ABC is equilateral.

Example 2:
Description: Construct a right angle at a point on a line. Given point A on a ray.
Tool List: [Line Tool, Circle Tool, Intersection Tool, Perpendicular Bisector Tool]
Solution:
Circle Tool: Construct a circle with center A and an arbitrary radius, intersecting the ray at points B and C.
Perpendicular Bisector Tool: Construct the perpendicular bisector of BC.

Example 3:
Description: Bisect a given segment. Let AB be the given segment.
Tool List: [Line Tool, Circle Tool, Intersection Tool]
Solution:
Circle Tool: Construct a circle with center A and radius AB.
Circle Tool: Construct a circle with center B and radius AB, intersecting circle A at points C and D.
Line Tool: Construct line CD, intersecting AB at point E. Point E is the midpoint of AB.

Example 4:
Description: Construct a 45-degree angle given a right angle. Let angle XYZ be a right angle with vertex Y.
Tool List: [Line Tool, Circle Tool, Intersection Tool, Perpendicular Bisector Tool]
Solution:
Circle Tool: Construct a circle with center Y and an arbitrary radius, intersecting XY at point A and YZ at point B.
Perpendicular Bisector Tool: Construct the perpendicular bisector of AB, intersecting XY at point C and YZ at point D.
Line Tool: Construct lines YC and YD. Angles XYC and YYD are 45 degrees.

Example 5:
Description: Construct the perpendicular from a point to a line. Given point A and line l.
Tool List: [Line Tool, Circle Tool, Intersection Tool, Perpendicular Bisector Tool]
Solution:
Circle Tool: Construct a circle with center A intersecting line l at points B and C.

Problem Solution [ChatGPT]:
Circle Tool: Construct a circle at point A with radius equal to the distance between point A and any of the other two vertices, let's say point B.
Circle Tool: Construct a circle at point B with radius equal to AB.
Intersection Tool: Use the Intersection Tool to find the two intersection points between the two circles. These points are the circumcircle's endpoints.
Line Tool: Connect the two intersection points to form the circumcircle of triangle ABC.

Static Prompt ✗

[...]
Example 1:
Description: Inscribe a circle in the square.
Tool List: [Line Tool, Circle Tool, Perpendicular Bisector Tool]
Solution:
Line Tool: Construct line AC.
Perpendicular Bisector Tool: Construct the perpendicular bisector of AB, intersecting AB at E and AC at F.
Circle Tool: Construct a circle with center E and radius EF.

Example 2:
Description: Construct an equilateral triangle that is circumscribed about the circle O with radius R and contains the given point A on its circumference.
Tool List: [Line Tool, Circle Tool, Perpendicular Bisector Tool, Perpendicular Tool]
Solution:
Line Tool: Construct line OA, intersecting the circle at B
Perpendicular Tool: Construct the perpendicular to OA from A
Circle Tool: Construct the circle with center B and radius OB, intersecting the circle at C and D, and line OA at E
Line Tool: Construct line CE
Line Tool: Construct line DE

Example 3:
Description: Construct the midpoint of the segment that connects the midpoints of the diagonals of the quadrilateral.
Tool List: [Line Tool, Circle Tool, Perpendicular Bisector Tool, Perpendicular Tool]
Solution:
Construct the perpendicular bisector of AB, intersecting AB at E
Construct the perpendicular bisector of CD, intersecting CD at F
Construct the perpendicular bisector of EF
Construct line EF

Example 4:
Description: Inscribe an equilateral triangle in the circle using the given point as a vertex. The center of the circle is not given.
Tool List: [Line Tool, Circle Tool, Perpendicular Bisector Tool, Perpendicular Tool]
Solution:
Circle Tool: Construct a circle with center A and an arbitrary radius, intersecting the original circle at B and C
Circle Tool: Construct the circle with center B and radius AB, intersecting circle A at D and E
Perpendicular Bisector Tool: Construct the perpendicular bisector of CD, intersecting the original circle at F
Perpendicular Bisector Tool: Construct the perpendicular bisector of CE, intersecting the original circle at G
Line Tool: Construct line FG

Example 5:
Description: Circumscribe a square about the circle O. Two of its sides should be parallel to the given line.
Tool List: [Line Tool, Circle Tool, Perpendicular Bisector Tool, Perpendicular Tool]
Solution:
Perpendicular Tool: Construct the perpendicular to the line through O, intersecting the circle at A and B
Perpendicular Tool: Construct the perpendicular to AB from A
Perpendicular Tool: Construct the perpendicular to AB from B
Circle Tool: Construct circle with center A and radius OA, intersecting the perpendicular to AB from A at C and D
Perpendicular Tool: Construct the perpendicular to CD from C
Perpendicular Tool: Construct the perpendicular to CD from D

Problem Solution [ChatGPT]:
Perpendicular Bisector Tool: Construct the perpendicular bisector of side AB. Label the intersection with AB as D.
Perpendicular Bisector Tool: Construct the perpendicular bisector of side BC. Label the intersection with BC as E.
Perpendicular Bisector Tool: Construct the perpendicular bisector of side AC. Label the intersection with AC as F.
Intersection Tool: Label the intersection of the perpendicular bisectors as O.
Circle Tool: Create a circle with center O and radius OA.

Adaptive ST ✓

[...]
Example 1:
Description: Construct a segment connecting the sides of the angle to get a triangle whose perpendicular bisectors are intersected in the point O.
Tool List: [Line Tool, Circle Tool, Perpendicular Bisector Tool]
Solution:
Circle Tool: Construct the circle with center O and radius OA, intersecting the sides of the angle at B and C
Line Tool: Construct line BC

Example 2:
Description: Construct an equilateral triangle that is circumscribed about the circle O with radius R and contains the given point A on its circumference.
Tool List: [Line Tool, Circle Tool, Perpendicular Bisector Tool, Perpendicular Tool]
Solution:
Line Tool: Construct line OA, intersecting the circle at B
Perpendicular Tool: Construct the perpendicular to OA from A
Circle Tool: Construct the circle with center B and radius OB, intersecting the circle at C and D, and line OA at E
Line Tool: Construct line CE
Line Tool: Construct line DE

Example 3:
Description: Construct the midpoint of the segment that connects the midpoints of the diagonals of the quadrilateral.
Tool List: [Line Tool, Circle Tool, Intersection Tool, Perpendicular Bisector Tool, Angle Bisector Tool, Perpendicular Tool]
Solution:
Perpendicular Bisector Tool: Construct the perpendicular bisector of AB, intersecting AB at E
Perpendicular Bisector Tool: Construct the perpendicular bisector of CD, intersecting CD at F
Perpendicular Bisector Tool:Construct the perpendicular bisector of EF Line Tool: Construct line EF

Example 4:
Description: Inscribe an equilateral triangle in the circle using the given point as a vertex. The center of the circle is not given.
Tool List: [Line Tool, Circle Tool, Perpendicular Bisector Tool, Perpendicular Tool]
Solution:
Circle Tool: Construct a circle with center A and an arbitrary radius, intersecting the original circle at B and C
Circle Tool: Construct the circle with center B and radius AB, intersecting circle A at D and E
Perpendicular Bisector Tool: Construct the perpendicular bisector of CD, intersecting the original circle at F
Perpendicular Bisector Tool: Construct the perpendicular bisector of CE, intersecting the original circle at G
Line Tool: Construct line FG

Example 5:
Description: Construct a segment connecting the sides of an angle to get a triangle whose orthocenter is in the given point O.
Tool List: [Line Tool, Circle Tool, Perpendicular Bisector Tool, Perpendicular Tool]
Solution:
Perpendicular Bisector Tool: Construct the perpendicular to one side R1 of the angle through O, intersecting the second side at A
Perpendicular Bisector Tool: Construct the perpendicular to the second side R2 of the angle through O, intersecting the first side at B
Line Tool: Construct line AB

Problem Solution [ChatGPT]:
Perpendicular Bisector Tool: Construct the perpendicular bisector of side AB. Label the intersection with AB as D.
Perpendicular Bisector Tool: Construct the perpendicular bisector of side BC. Label the intersection with BC as E.
Circle Tool: Create a circle with center D and radius AD.

Adaptive Self ✓

We observe that the static prompt leads to an incorrect solution, with the model misinterpreting the meaning of a circumcircle, and blindly performing step sequences similar to those presented in the prompt.

In the case of Adaptive ST, the prompt consists of examples more closely related to the task at hand, leading to a correct solution. However, the model still suggests unnecessary steps, which can be attributed to style-mimicking tendencies introduced by instruction tuning.

Finally, when the model is tasked to filter the examples by itself - Adaptive Self, we observe an overlap in the selected examples with the Sentence Transformer method. Here, the model suggests a correct sequence of steps, free from any unnecessary or repetitive tool usage.

E Target Variable Name Bias

Below, we present the completions obtained from GPT4 when presented with the problem: *"Find a point that is equidistant from given points A and B."*, as presented visually in Figure 4.

Given two points, A and B, find a point **[Target]** so that its distance from A is equal to its distance from B.

Problem

[Line Tool, Circle Tool, Intersection Tool, Perpendicular Bisector Tool]

Available Tools

Circle Tool: Construct a circle with center A and radius AB.
Circle Tool: Construct another circle with center B and radius AB.
Intersect Tool: Identify the points where the two circles intersect. These intersection points are equidistant from A and B. Let's call one of these intersection points C.
The point C, found at the intersection of the two circles, satisfies the condition that its distance from A is equal to its distance from B.

Target = C

Circle Tool: Construct a circle with center A and radius AB.
Circle Tool: Construct a circle with center B and radius AB.
Intersect Tool: Find the intersection points of the two circles, let's call them C and D. These points are equidistant from A and B.
By following these steps, point D will be at an equal distance from both points A and B.

Target = D

Circle Tool: Use the circle tool to draw a circle with center A and radius AB.
Circle Tool: Use the circle tool again to draw a circle with center B and radius AB.
Intersect Tool: The two circles will intersect at two points, which we can call point C and point D.
Line Tool: Draw the line CD.
Choose any point on line CD (other than C or D) and label it point E.

Target = E

Circle Tool: Construct a circle with center A and radius AB.
Circle Tool: Construct a circle with center B and radius AB.
Intersection Tool: The intersection points of the two circles represent the points that are equidistant from A and B. These intersection points are the solutions for point X.
In this case, **there will be two solutions for X**, each lying on the circumference of the circles where they intersect.

Target = X

Table 10: Completions for different name choices of the target variable.

F Baselines

We introduce two baselines to analyze the complexity of the geometric construction problem. In the first baseline, which we will call "Longest Common Solution" (LCS), we identify the top five longest common sequences of steps between ground truth solutions. Then, for each given problem, we uniformly sample from these sequences, adjusting the variables of each step and the tool usage to the current task. If a sampled step does not apply to the current state of the problem, the sampled sequence is discarded, and a new one is sampled instead. This baseline corresponds to the success rate of an agent who memorized a set of fixed solution steps and applied them to each given problem.

Additionally, we propose a second baseline named 'N-Gram Rollouts' (N-Gram), which begins by creating a database of uni-, bi-, and tri-grams derived from the tools used in ground-truth answers. For each problem, our method involves a two-phase iterative process. Initially, we choose either a single tool ($n = 1$) or a sequence of tools ($n > 1$) from our database. Following this, we select the geometric variables upon which these tools will be applied. To facilitate this, we maintain a memory initially populated with variables given in the problem statement. It is important to note that tool application varies in complexity, with some tools requiring a single variable (e.g., constructing a ray from point A) and others necessitating two (e.g., drawing a line between points A and B). For each tool or sequence of tools selected, we sample the required number of variables from memory, with recent variables weighted more heavily than older ones, following an exponential decay schema. Any new variables a tool generates (such as a new point) are added to this memory. This process is repeated until a predefined number of steps is reached.

Method	**Correct tool sequence**	**Fully correct solution**
LCS	0.17	0.02
1-Gram	0.14	0.11
2-Gram	0.12	0.09
3-Gram	0.07	0.06

Table 11: Performance of Baselines. Results refer to pass@1 estimated after 1000 trials.

According to Table 11, finding an entirely correct solution randomly is almost 0.1% probable. Note here that finding the correct tool sequence score is significantly higher. However, this is expected since many problems share common starting steps (e.g., drawing a line or a circle).

G Visual Relations Prompt

Here are two examples presenting the benefits of VRP inclusion. In Example 1, the models are devoid of visual input and seem to struggle to find a correct solution to the problem. On the contrary, in example 2, the models are now equipped with the VRP and complete the challenge successfully.

Example 1:

Given the rectangle, ABCD, and the point E located over the side AB at a random position, construct a line through E that cuts the rectangle into two parts of equal area.

Problem

[Line Tool, Circle Tool, Perpendicular Bisector Tool, Angle Bisector Tool, Intersect Tool]

Available Tools

Line Tool: Construct line EF, parallel to AB.
Perpendicular Bisector Tool: Construct the perpendicular bisector of line EF.
Intersect Tool: Mark the intersection of the perpendicular bisector and line AB as point G.
Line Tool: Construct line EG.✗

MetaMath-Mistral

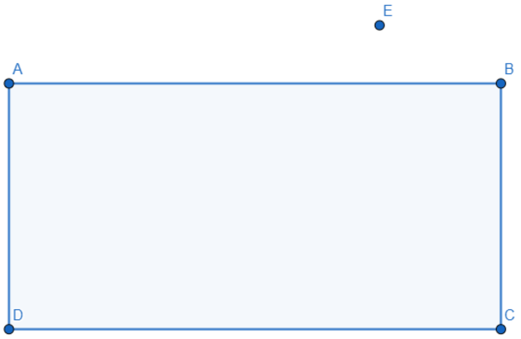
Perpendicular Bisector: Construct the perpendicular bisector of AB. This bisector will cut the rectangle into two equal parts.
Intersect Tool: Extend this perpendicular bisector line to intersect side CD. Label this intersection point as F.
Draw a line through point E and point F. This line will cut the rectangle into two parts of equal area. ✗

ChatGPT

Line Tool: Construct Line AE.
Perpendicular Bisector: Construct the perpendicular bisector of CD.
Intersect Tool: Mark the intersection of the perpendicular bisector with CD and mark it as H. This point H is the midpoint of CD.
Line Tool: Draw a line from point E to point H. ✗

GPT4

Example 2:



Image

Given the rectangle, ABCD, and point E located over the side AB at a random position, construct a line through E that cuts the rectangle into two parts of equal area.

Problem

[Line Tool, Circle Tool, Perpendicular Bisector Tool, Angle Bisector Tool, Intersect Tool]

Available Tools

Here is the full description of all the relations between geometric objects:
Shapes:
Rectangle: ABCD
Points:
A: Top-left corner of the rectangle.
B: Top-right corner of the rectangle.
C: Bottom-right corner of the rectangle.
D: Bottom-left corner of the rectangle.
E: Point outside of the rectangle.
Lines:
AB: Top horizontal line of the rectangle.
BC: Right vertical line of the rectangle.
CD: Bottom horizontal line of the rectangle.
DA: Left vertical line of the rectangle.
Relations:
AB is parallel to CD.
AD is parallel to BC.
The length of AB is equal to the length of CD.
The length of AD is equal to the length of BC.
E is an isolated point, not connected to any lines of the rectangle ABCD.

Visual Relations Prompt

Perpendicular Bisector Tool: Construct the perpendicular bisector of the line AB.
Perpendicular Bisector Tool: Construct the perpendicular bisector of the line CD.
Perpendicular Bisector Tool: Construct the perpendicular bisector of the line BC.
Perpendicular Bisector Tool: Construct the perpendicular bisector of the line DA.
Intersect Tool: Find the intersection of the perpendicular bisectors of AB and CD and name it F.
Intersect Tool: Find the intersection of the perpendicular bisectors of BC and DA and name it G.
Line Tool: Draw a line from E to F.✓

MetaMath-Mistral

Line Tool: Draw a diagonal of the rectangle AC.
Perpendicular Bisector Tool: Construct the perpendicular bisector of AC.
Intersect Tool: Label the intersection point of the perpendicular bisector and AC as M.
Line Tool: Draw a line segment from point E to point M. This line, EM, will cut the rectangle ABCD into two parts of equal area since it passes through the midpoint of the diagonal AC.✓

ChatGPT

Line Tool: Draw a diagonal of the rectangle AC.
Perpendicular Bisector Tool: Construct the perpendicular bisector of AC.
Intersect Tool: Label the intersection point of the perpendicular bisector and AC as M.
Line Tool: Draw a line segment from point E to point M. This line, EM, will cut the rectangle ABCD into two parts of equal area since it passes through the midpoint of the diagonal AC.✓

GPT4

When tasked with tackling the same geometrical problem, our three agents (MetaMath-Mistral / ChatGPT / GPT4) resort to a "convenient" placement of point E outside rectangle ABCD. All of them try to utilize the fact that the perpendicular bisector of any edge of a rectangle cuts the rectangle into two equal areas. Then, a series of erroneous steps are suggested, trying to connect point E with this line.

After introducing the VRP, we observed that all agents managed to correct their responses. Placing geometric objects in the prompt helps them better understand the problem's nature. Agents no longer resort to steps that suggest splitting the rectangle area into two equal parts, ignoring point E, but rather respect the particular spatial circumstances the problem presents. Note here that the initial solutions are valid if the random position of point E happens to align with the midpoint of any side. The initial tendency to produce such solutions can be attributed to the fact that the splitting of a rectangle must have been observed more frequently in their training data through the use of diagonals or lines connecting midpoints.

H Multi-Agent Dialogue Examples

Illustration of our proposed $SV_{NL} - SV_{GT}$ multi-agent setup. We present the agent dialogue leading to a successful solution. The use of a single ChatGPT instance is not sufficient to solve this particular problem. However, the combined power of multiple agents can.

According to our proposed method, we employ 4 agents:

1. Natural Language Solver (NL Solver)
2. Natural Language Validator (NL Validator)
3. Geometric Tools Solver (GT Solver)
4. Geometric Tools Validator (GT Validator)

The four different agents interact with each other from top to bottom and from left to right.

Given the square ABCD, inscribe a circle in the square.

Problem

[Line Tool, Circle Tool, Perpendicular Bisector Tool, Intersect Tool]

Available Tools

[...] To inscribe a circle in the square ABCD means to draw a circle such that all four sides of the square are tangent to the circle. The circle should touch each side of the square at exactly one point. I suggest the following steps:
<STEP>1: Draw the square ABCD.
<STEP>2: Draw the diagonal AC of the square. The diagonal AC will pass through the center of the circle.
<STEP>3: Construct the perpendicular bisector of the diagonal AC. This line will intersect the diagonal at the center of the circle.
<STEP>4: Draw a line segment from the center of the circle to one of the vertices of the square (e.g., A or C). This line will be the radius of the circle.
<STEP>5: Use the compass to measure the length of the line segment drawn in the previous step.
<STEP>6: Set the compass to the measured length and draw a circle with the center at the intersection of the diagonal AC and the perpendicular bisector.

NL Solver

According to the task and the provided rationale, I am confident regarding the validity of the following steps:
<STEP>1: Correct.
<STEP>2: Correct.
<STEP>3: Correct.
Please try to correct the remaining steps.

NL Validator

[...]<STEP>4: Draw a line segment from the center of the circle to one of the points where the circle is tangent to the square. This line segment will be the radius of the circle.

Following these corrected steps, we will successfully inscribe a circle in the square ABCD.

NL Solver

Given the suggested expert rationale and the available tools I suggest the following steps:
<Line Tool>1: Draw a line segment from point A to point B.
<Line Tool>2: Draw a line segment from point A to point C.
<Perpendicular Bisector Tool>3: Draw the perpendicular bisector of AB.
<Intersect Tool>4: Mark the intersection point of the bisector and AB as E.
<Intersect Tool>5: Mark the intersection of the perpendicular bisector and the diagonal AC as O.
<Circle Tool>6: Create a circle with center O and radius AO.

GT Solver

According to the steps given, the available tools and the suggested rationale, I am confident of the validity of the following steps:
<Line Tool>1: Correct.
<Line Tool>2: Correct.
<Perpendicular Bisector Tool>3: Correct.
<Intersect Tool>4: Correct.
<Intersect Tool>5: Correct.
Please try to correct the solution according to my suggestions.

GT Validator

[...] <Circle Tool>6: Create a circle with center O and radius OE. ✓

GT Solver

ChatGPT Simulaca

Below, we present another example where we showcase the benefits of incorporating variable renaming in multi-agent scenarios. Simulacra are not immune to the variable name bias: In the example below, the variables **A**, **B**, and **C** are used. **A** and **B** mark the two initial points of the problem, while **C** stands as the target to be found. Initially, the NL Solver comes up with a promising 5-step rationale. However, the NL Validator corrects the proposed final step as a mistake. Since **A**,**B**, and **C** are consecutive letters in the alphabet, and **C** has already appeared in the fourth step, the validator marks the final step as redundant. The new variable **D** generated at the fifth step is after **C** (in terms of alphabetical ordering), and thus the problem must have been already solved. This error is propagated to the GT agents, leading to an eventually wrong final response.

Let $|AB| = 1$. Construct a point C on the line AB such that the length of AC is equal to $\sqrt{2}$.Do not use arbitrary numbers but only existing lengths and sizes in your solution

Problem

[Line Tool, Circle Tool, Perpendicular Bisector Tool, Perpendicular Tool, Angle Bisector Tool, Intersect Tool]

Available Tools

[...] To create a line segment whose length is the square root of 2, we should utilize an existing segment of length 1, here AB. This requires forming a right-angled triangle where one side measures 1, and determining the hypotenuse's length which will be the square root of 2. I suggest the following steps:
<STEP>1: Draw a straight line segment AB of any length. This will serve as our unit length.
<STEP>2: At point B, construct a perpendicular line to AB.
<STEP>3: From point B, draw a line segment BC on the perpendicular line with the same length as AB.
<STEP>4: Connect points A and C to form a right triangle ABC. The length of AC, the hypotenuse, is the square root of 2.
<STEP>5: Finally, mark the point D on the line AB such that AD = AC This will give us the line segment AD of length square root of 2. ✓

NL Solver

According to the task and the provided rationale, I am confident regarding the validity of the following steps:
<STEP>1: Correct.
<STEP>2: Correct.
<STEP>3: Correct.
<STEP>4: Correct.
Please try to correct the solution according to my suggestions.

NL Validator

I suggest we should re-write the steps so that all suggested changes are incorporated.
[...] <STEP>5: Point C is the required point on the line segment extended from AB such that the length AC equals the square root of 2.✗

NL Solver

Given the suggested expert rationale and the available tools I suggest the following steps:
<Circle Tool>1: Use the circle tool to draw a circle with center A and radius AB.
<Perpendicular Tool>2: Draw a perpendicular line to AB at point B using the Perpendicular Tool.
<Circle Tool>3: Draw a circle with center B and radius BA.
<Intersect Tool>4: Use the Intersection Tool to find point C, which is the intersection of the circle centered at B and the line perpendicular to AB.
<Line Tool>5: Draw a line from A to C using the Line Tool. ✗

GT Solver

According to the steps given, the available tools and the suggested rationale, I am confident of the validity of the following steps:
<Circle Tool>1: Correct.
<Perpendicular Tool>2: Correct.
<Circle Tool>3: Correct.
<Intersect Tool>4: Correct.
<Line Tool>5: Correct.

Great job!

GT Validator

Thank you!

GT Solver

GPT4 Simulacra

Now, let us rename the target variable and present the problem to our multi-agent setup again. Instead of the consecutive letter **C**, we swap the final target with a far-away (in terms of alphabetical ordering) as well as a common mathematical symbol **X**. We observe that the validator now approves the suggested rationale, leading to a successful generation.

Let $|AB| = 1$. Construct a point X on the line AB such that the length of AX is equal to $\sqrt{2}$. Do not use arbitrary numbers but only existing lengths and sizes in your solution.

Target Variable Renaming

[...] Based on the problem description, we are tasked with constructing a point on a given line segment such that the length of the segment from the initial point to the new point equals the square root of 2 without using explicit numerical values. Here's how we can approach this:
<STEP>1: Draw a straight line segment AB of any length (this will be our unit length as we are not to use numbers).
<STEP>2: Construct a circle with center at A and radius AB.
<STEP>3: Draw a line perpendicular to AB at point B. This line intersects the circle A at a new point D.
<STEP>4: Draw a line segment AD. The length of AD is the square root of 2 (according to Pythagoras' theorem).
<STEP>5: Take point X such that AX = AD. Now AX is equal to the square root of 2.✓

NL Solver

According to the task and the provided rationale, I am confident regarding the validity of the following steps:
<STEP>1: Correct.
<STEP>2: Correct.
<STEP>3: Correct.
<STEP>4: Correct.
<STEP>5: Correct.
Your solution is valid!

NL Validator

Thank you for your help!

NL Solver

Given the suggested expert rationale and the available tools I suggest the following steps:
<Line Tool>1: Use the line tool to draw the line AB.
<Circle Tool>2: Draw a circle with center B and radius BA.
<Perpendicular Tool>3: Draw a perpendicular line at point B.
<Intersect Tool>4: Use the Intersection Tool to find point D, which is the intersection of the circle centered at B and the line perpendicular to AB.
<Line Tool>5: Draw the line AD.
<Line Tool>6: Draw a line from A to X. ✗

GT Solver

According to the steps given, the available tools and the suggested rationale, I am confident of the validity of the following steps:
<Line Tool>1: Correct.
<Circle Tool>2: Correct.
<Perpendicular Tool>3: Correct.
<Intersect Tool>4: Correct.
<Line Tool>5: Correct.
Please try to correct the solution according to my suggestions.

GT Validator

[...] <Circle Tool>6: Draw a circle with center A and radius AD.
<Intersect Tool>7: Mark the point of intersection of the circle with the line AB as X. X is the desired point. ✓

GT Solver

GPT4 Simulacra

I Human Study On Euclidea Dataset

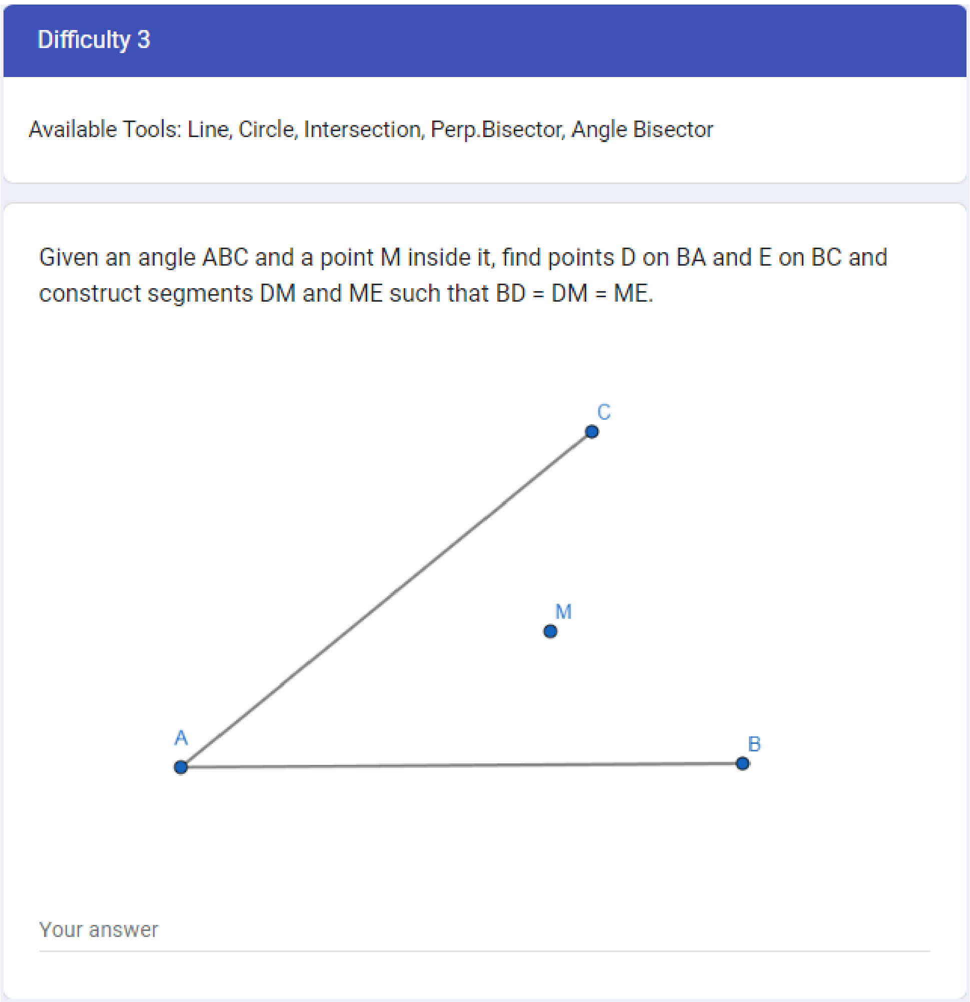
While there is no concrete data on human performance - assuming an adequate level of geometric knowledge background - on the Euclidea challenge, Macke et al. (2021) describe these challenges as formidable even for well-prepared mathematicians, including International Mathematics Olympiad (IMO) participants, whereas Wong et al. (2022) characterizes the dataset as notably difficult. To validate these claims, we conducted an empirical study involving ten individuals with mathematical or computer science backgrounds, of graduate level, and experience with geometrical problem solving. The participants collaborated voluntarily and without any financial compensation.

Given the progression of the original Euclidea game, which requires solving preceding problems to advance, we decided to sample 20 problems across all packs. The outcomes revealed a success rate ranging from 20% to 35% - 5 to 7 correctly solved problems - with the average performance at 30.5%, indicating the high level of challenge presented by the tested geometry tasks. This finding underscores the domain's complexity. Table 12 contains the problems that were presented to participants, while Figure 5 showcases an example. Here is the list of the problems presented to the participants:

Name	Pack
Angle of 60	Alpha
Rhombus in Rectangle	Alpha
Angle of 30	Beta
Circle in Rhombus	Beta
Three equal segments v1	Gamma
Center of Quadrilateral	Gamma
Square Root of 3	Delta
Equilateral Triangle in Circle	Delta
Square in Square	Epsilon
Regular Hexagon	Epsilon
Given Angle Bisector	Zeta
Symmetry of Four Lines	Zeta
Annulus	Eta
Angle of 75	Eta
Egyptian Triangle	Theta
Torricelli Point	Theta
Harmonic Mean of Trapezoid Bases	Iota
Minimum Perimeter v2	Iota
Rotation 60	Kappa
Inner Tangent	Kappa

Table 12: Euclidea problems included in the human study.

Here is what a question in the questionnaire looked like:



Difficulty 3

Available Tools: Line, Circle, Intersection, Perp.Bisector, Angle Bisector

Given an angle ABC and a point M inside it, find points D on BA and E on BC and construct segments DM and ME such that BD = DM = ME.

Your answer

Figure 5—Example of a given problem in the human study: On top, the available tools are provided to the participant. Below, an image containing the annotated initial setup of the problem is provided. The participant is then asked to use the provided tools and respond with their solution.

J On the use of AI assistants

We disclose that AI writing assistant tools were used during the creation of this work. However, these tools were strictly used to refine, summarize, and check the accuracy of grammar and syntax. No AI assistant was involved in generating ideas or constructing arguments to support our perspectives or claims. All original thoughts and analyses are entirely our own.