Optimizing Geometry Problem Sets for Skill Development

M. Bouzinier[1][2], S. Trifonov[2]

The majority of this work was performed in 1991 - 1996 by *R.K Gordin, M. Bouzinier, S. Trifonov, and I.F. Sharygin*. Unfortunately, I.F. Sharygin passed away on the 12 of March, 2004 and R.K. Gordin is unable to continue working on this project because of reasons beyond our control. We acknowledge a major intellectual contribution by I.F. Sharygin and the primary role that R.K. Gordin has played in developing the product, however for the reasons above we cannot list them as the authors.

Abstract

This article describes an ontology and methodology for annotating and organizing Euclidean Geometry problems, developed in the early 1990s and implemented as a software tool. While the majority of this work — including the ontology and solution graph paradigm — was completed over thirty years ago, we argue that it has renewed relevance in the context of modern artificial intelligence. In particular, we explore the hypothesis that this established framework can facilitate automated solution validation and feedback when paired with contemporary large language models, thereby supporting teachers and self-learners in geometry education. We document the original architecture and its enduring value, and outline pathways for bridging historical educational resources with next-generation AI techniques.

History

In the early 1990s we developed a software tool to aid high school teachers in teaching Euclidean Geometry (Gordin et al., 2000). The tool helped educators in creating problem sets that illustrated the practical application of specific problem-solving skills and provided students with additional assignments to master these skills. Initially released as a standalone application for DOS-based personal computers, the software featured approximately 5,000 geometry problems, all in Russian. Over the years, a new team of developers transitioned the tool into a web application, expanding the problem database to about 16,000 problems today.

To optimize the selection of problems for constructing these sets, we designed a comprehensive ontology for problem annotation. This ontology includes roughly two dozen annotation classes and subclasses, covering attributes such as difficulty, problem type, topic, purpose, and more. Further details about the ontology are provided in the [Appendix 1](#).

---

[1] Harvard University, Research Computing and Data Services
[2] Forome Association

Annotating Problem Solutions

The annotation of geometry problems and their solutions is most effective when supported by a rigorous and well-structured framework. To ensure that educational content can be consistently categorized, easily searched, and adapted to evolving curricula, it is essential to establish a formal ontology — a systematic classification of all relevant elements and their relationships within the domain. Specifically, in the context of problem-solving, this means identifying and organizing the key elements involved. We group these elements into three main classes:

- **Facts**. Euclidean axioms, derived theorems, lemmas, and other provable statements within the Euclidean geometry framework.
- **Geometric Objects**: Figures and concepts either given in problem statements or constructed during the problem-solving process.
- **Methods**: Specific techniques or strategies that can be applied to arrive at a solution.

Within the triadic structure of our ontology — facts, objects, and methods — the designation of **objects** as a distinct category plays a fundamental role. The composition of the object category is typically the most stable aspect of the ontology. While the lists of facts (such as theorems or lemmas) and methods (problem-solving strategies) gradually stabilize with ongoing development and extensive annotation of problem collections, they may remain somewhat flexible and subject to extension and revision as the ontology evolves. In contrast, changes to the set of objects are exceedingly rare, yet when they do occur, they can significantly alter the overall structure of the subject domain. Objects serve as the conceptual building blocks; modifying this set has deep ramifications for the organization and interpretation of both facts and methods across the ontology. For instance, introducing a new geometric object—such as a "nine-point circle"—requires re-examining related facts and may enable new methods, illustrating how fundamental changes to the object category cascade through the ontology.

Notably, while "Methods" served historically as the core building blocks of the ontology, there is a nuanced difference between a **method** as an element defined in the ontology and a **skill** as used in education. Methods refer to well-defined, specific techniques — such as constructing an altitude, using the area method, or applying a sequence of steps to prove congruence. Skills, in the pedagogical sense, encompass more than correct execution of methods; they include the ability to recognize when and how to apply these methods in varied contexts, to recall and deploy relevant facts, to identify pertinent objects, and to creatively combine multiple elements in problem-solving. In short, a student's skill is demonstrated in their mastery of selecting, integrating, and applying methods, facts, and objects to solve a wide array of problems.

It is important to acknowledge that many problems can be approached and solved using various methods. Good educators should encourage diverse approaches to problem solving, rather than insist on solutions that mirror those demonstrated in class. Nevertheless, curricula seek to develop specific skills, so problem selection should align with these learning objectives.

Throughout the curriculum, students continually reinforce skills they already possess while being introduced to new ones. For effective skill illustration or practice, problems should meet these criteria:

1. **Existing Skills**: The problem should require only those skills students have already acquired.
2. **Target Skill Integration**: The skill being taught should play an essential role in the solution.
3. **Skill Necessity or Efficiency**: There should be no reasonable alternative solution that excludes the target skill, or else the skill should significantly simplify the solution.

To support efficient problem selection and curriculum alignment, we represent solutions as **Solution Graphs** — directed acyclic graphs (DAGs) whose nodes correspond to skills (facts, objects, or methods). Effective problem design requires a path through the graph that passes through required skills and does not allow any shortcut bypassing newly introduced skills. This approach enables teachers to track the learning trajectory and ensure curricular coherence at every stage. Recent AI systems for geometry, such as AlphaGeometry (Lu et al., 2024), similarly represent solutions as proof graphs for automated search and validation. Likewise, (Qian et al., 2025), represent formal proofs as directed acyclic graphs, where each node corresponds to an intermediate statement or construction and edges represent logical dependencies.

Our experience over the past thirty years has shown that, in some cases, this methodology empowered self-motivated students to develop significant problem-solving skills independently, with only occasional teacher consultation. The granularity of method-based tracking and automated feedback made self-directed skill development and mastery possible.

Today's Relevance: from building sets to validating proofs

The original tool, though innovative, remains a niche product. This is primarily due to two factors. First, constructing problem sets is typically a task for curriculum developers, not an everyday responsibility for teachers. Teachers might be creating personalized training assignments, but this happens infrequently in practice. Secondly, Euclidean Geometry has become less prominent in modern Western education. Ironically, the latter factor also underscores why the tool's foundation might hold greater relevance today than it did 30 years ago.

Euclidean Geometry teaches students both logical reasoning and manipulation with abstract concepts. At the same time, two dimensional geometry is intuitive and allows for easy illustration of both logical frameworks and abstractions through text and geometric drawings. This makes it an ideal medium for teaching these skills to young students when their brains are most receptive. In an era where AI is expected to offload many routine tasks, proficiency in logical reasoning and high-level abstraction is becoming increasingly crucial. Thus, Euclidean

Geometry presents itself as a unique and valuable subject in contemporary school curricula. Recent developments in AI, such as AlphaGeometry (Lu et al., 2024), underscore both the feasibility and the pedagogical potential of automated geometry problem solving and proof generation.

One barrier to its popularity might be the challenge of verifying solutions to geometry problems, a task requiring significant expertise and tedious work. These problems often require proofs that can be composed in multiple ways. Demanding that young students produce strictly formalized proofs for automated validation is impractical. Moreover, geometric construction problems, usually described less formally than proofs, further complicate automatic validation.

We might speculate that a software tool to assist teachers in validating student solutions would make popularizing the teaching of Euclidean Geometry more feasible, leading to including the subject in the majority of school curricula. We believe the methodology for problem solution annotation we developed three decades ago could significantly aid in automating solution verification.

With the explosion in popularity of AI in general, and large language models (LLM) in particular, many recent works have focused on generating and validating formal proofs for mathematical problems, including Euclidean geometry. For instance, AlphaGeometry (Lu et al., 2024) underscores the critical role of structured geometric knowledge in enabling automated proof search and validation. Their system relies on a comprehensive database of geometric theorems, facts, and construction techniques, and effectively organizes proof steps as a DAG. However, the authors stop short of formalizing their technique as an ontology-based approach; that is, the explicit classification and systematization of geometric knowledge into foundational objects, facts, and methods remains implicit rather than formally defined. Furthermore, in AlphaGeometry, the primary purpose of constructing a DAG is to search for a single valid proof, rather than to classify and organize all possible approaches to solving a problem, as emphasized in our ontology-based methodology. As summarized in (Sinclair et al., 2017), there is a growing recognition of the importance of adaptive technologies and ontologically-informed resources for geometry instruction, tracking, and assessment.

A path forward to automating solution validation

Milestones

We identify two important milestones on the path to fully automating solution validation:

- **Computer Aided Validation.** This approach can substantially reduce the time teachers spend validating student proofs and geometric constructions through partial automation. While still requiring some, though reduced, level of human participation for validating unusual and creative solutions to difficult problems, it promises to automate the validation of simpler problems effectively.

- **Interactive Feedback for Self-Learners**. By enabling interactive feedback while students are writing proofs, we can highlight incorrect or irrelevant statements. Additionally, providing hints can help students progress when they encounter difficulties.

Methodology

Our proposal centers around several key strategies.

1. **Integration of Solution Graphs and Proof Validation Tools**. We aim to merge the concept of Solution Graphs with existing formal proof validation tools.
2. **AI-Driven Annotation**. Recent advances in large language models have demonstrated considerable promise in semantic parsing and educational feedback. Notably, state-of-the-art systems such as AlphaGeometry (Lu et al., 2024) have shown that neural-symbolic approaches can automate geometry proof annotation and verification at the Olympiad level. By training advanced AI models, specifically large language models (LLMs), we can automate the annotation of problems by constructing Solution Graphs. In U.S. curricula, proofs are often rendered in a statement-reason form rather than free text. This structured approach significantly streamlines and facilitates automation. While construction problems lack a universally accepted format, making them more challenging to parse, we remain optimistic about achieving this goal.
3. **Mapping Solutions to Solution Graphs**. Another task for AI involves mapping a student's solution to a known Solution Graph. Like the annotation task, this is a semantic parsing activity well-suited for LLMs.

In this context, the statement-reason form is interpreted as a series of tuples (e.g., statement, reason). The goal is to translate these into a formal logical language supportive of proof validation. Languages such as Lean (Qian et al., 2025; Murphy et al., 2024; Song et al., 2025, L. de Moura et al, 2021) and Coq (Narboux, et al, 2012; The Coq Reference Manual, 2023, The Coq proof assistant, 2023) are prime candidates for consideration. The validation process can be selectively applied to sub-sequences of these tuples, distinguishing between correct, possibly incorrect, and irrelevant statements.

Given our proactive knowledge of the solution graph, we can preemptively identify potential correct statements, regardless of their explicit presence in a student's solution. This allows us to discern:

- Statements that are incorrect or unproven.
- Statements that are correct but unproven due to inaccurate reasons.
- Statements that are correct and proven but likely irrelevant.
- Statements that are both correct and relevant, potentially leading to a valid solution.

Achieving our first milestone involves providing feedback to teachers on solutions that may still require manual validation, while also offering hints to students to reach the second milestone.

Interactive Input Modes for Proof Solutions

In the quest to provide effective interactive feedback, particularly for self-learners, one possible approach involves supporting multiple modes for entering statement-reason proofs. Several broad approaches can be discussed:

Constructed Statements (Text-based). Here, students assemble formalized statements using a controlled interface—this might involve drop-down selections, templates, or textual prompts guiding the formation of statements such as "M is the midpoint of segment AB" or "Triangle ABC is congruent to triangle LMN." This approach can be implemented through a graphical user interface (GUI) that assists with precise statement composition, but the end product — a formalized statement — is stored and processed as structured text. It is well suited to automated validation and provides scaffolding for learners struggling with mathematical language.

Constructed Statements (Figure-based): In this mode, students interact directly with a geometric figure — drawing points, constructing circles, or marking congruent angles — often with the help of dynamic geometry software. Statements about the construction are linked to the graphical actions. This mode is intuitive for simpler problems but can break down for advanced or degenerate cases, where, for example, significant elements (such as certain centers or intersections) may lie far outside the "visible" diagram or become indistinguishable.

Write-in Statements. Students write their own formal or semi-formal assertions and justifications directly, without the constraints or guidance of structured entry. This allows for greatest flexibility and is important for advanced problems. Recent advances in semantic parsing have shown the possibility of converting human-written statements in Euclidean geometry into formal Lean 4 code (Murphy et al., 2024; Qian et al., 2025; Song et al., 2025). These advances make automated feedback and formal proof checking increasingly feasible.

The system configuration could allow the disabling of constructed mode to focus solely on written-in submissions. Small errors, such as spelling mistakes, can be rectified using existing technologies, ensuring validation of student answers is as accurate as possible.

The most effective implementation strategy, and the degree to which these input modes can support rich educational feedback, remains an open question for future research and development.

Acknowledgments

We have used OpenAI GPT 4.1 and Anthropic Claude 4 Sonnet to check for grammar, improve style and check the text for consistency.

Appendix 1. Euclidean Geometry Ontology

Skills

Geometric Facts

The ontology includes around 200 geometric facts, encompassing theorems, lemmas, statements, and formulas from the geometry curriculum. It also features well-known propositions frequently utilized in competitive and Olympiad problem-solving. Below are some examples of these geometric facts:

- **Parallel Postulate**. Given a line and a point not on the line, there is exactly one line parallel to the given line that passes through the point.
- **Pythagorean Theorem**. In a right triangle, the square of the length of the hypotenuse is equal to the sum of the squares of the lengths of the other two sides.
- **Chord-Chord Product Theorem**. If two chords AB and CD intersect at point E inside a circle, the products of the lengths of the segments of each chord are equal: AE × EB = CE × ED
- **Euler Line Theorem**. In any non-equilateral triangle, the centroid, circumcenter, and orthocenter are collinear, lying on what is known as the Euler Line. This line also passes through the nine-point center.

Geometric Objects

The ontology encompasses around 130 geometric objects, covering fundamental figures encountered in a typical geometry curriculum. These include basic shapes such as lines, planes, triangles, circles, and various polygons, as well as three-dimensional forms like prisms and pyramids. It also addresses standard spatial configurations such as parallel lines, intersecting circles, tangents, and combinations of geometric bodies in space. Furthermore, the ontology incorporates quantities, sets, and transformations. Here are some additional examples:

- Right triangle
- Isosceles triangle
- Inscribed circle
- Circumscribed circle
- Point of intersection of medians in a triangle
- Circumscribed quadrilateral
- Centrally symmetrical figure
- Nine-point circle

Methods: specific techniques to solve geometric problems

The ontology includes over 80 methods specifically designed to solve or prove geometric problems. These techniques range from auxiliary constructions — such as extending a segment, drawing a perpendicular or parallel line, and straightening a broken line — to widely recognized problem-solving strategies like the area method, volume method, and methods involving auxiliary circles, equal triangles, and similar triangles. The ontology also covers the application of geometric transformations, trigonometry, and basic geometric constructions using a compass and ruler. Here are some specific examples:

- Constructing a Triangle by 3 Sides
- Symmetry About a Line by Folding the Drawing
- Proving Perpendicularity Using Equality of Adjacent Angles
- Using a Segment as a Diameter for a Circle
- Selecting a Midpoint of a Segment
- Finding Congruent Triangles
- Finding Similar Triangles
- Using a Median's Continuation with an Equal Segment
- Replacing Rotation with the Composition of Two Axial Symmetries
- Double Application of the Cosine Theorem

Problem Provenance

The ontology provides information about the origins of each problem, offering context and background through the following elements:

- **Authors**: The individuals credited with creating the problem, where known.
- **Sources**: The publications (books or journals) where the problem was first published, providing a bibliographic reference.
- **Named Problems**: Specific problems that carry distinct names, such as "Varignon's Theorem," "Erdős' Inequality," and "Leibniz's Formula." Typically, each name uniquely corresponds to a single problem.
- **Public Exams or Competitions**: The exams or competitions where the problem has been featured, indicating its level of recognition and use in assessments.

Difficulty

The difficulty of each problem is rated on a scale from 1 to 40, reflecting a somewhat subjective assessment by the tool's authors. The perceived difficulty of a problem for a student may be different from this assessment depending on the student's prior learning, problem-solving experience, mastery of curricular topics, and familiarity with material beyond the standard curriculum. Problems are categorized as follows:

- **1-10**: Basic school-level tasks that align with standard classroom material.
- **11-20**: Tasks suitable for competitions and advanced math classes, offering moderate challenges.
- **21-30**: Challenging tasks typically seen in Olympiad contexts, requiring a higher level of problem-solving skill.
- **31-40**: Very difficult problems, often demanding deep mathematical insight and advanced techniques.

Attributes

Key problem

Values: Yes, Perhaps, No

A key problem vividly illustrates a particular idea. Its solution may initially seem challenging, but it becomes significantly simpler once the underlying concept is understood. The significance of these problems lies in their ability to clarify concepts that, while frequently encountered and crucial, may appear trivial and unnoticed in their simplest form. Key problems typically serve as precursors to a larger set of related problems within the system.

Key problems are generally categorized into two types:

- **Problem-Theorem**: These problems convey a useful fact commonly applied in solving various geometric challenges. They exemplify a fundamental truth or principle that aids in problem-solving.
- **Problem-Method**: These problems highlight a specific method, technique, or idea. The solution to such problems showcases this central concept in its purest form, devoid of any extraneous ideas or complex technical details.

Synthetic problem

Values: Yes, Perhaps, No

A synthetic problem requires the integration of multiple ideas. Solving such problems necessitates identifying and combining these ideas effectively. Typically, synthetic problems span different topics and demand the use of diverse techniques, theorems, extensive calculations, and case enumeration, making them ideal for competitive exams.

Technical problem

Values: Yes, Perhaps, No

Technical problems are designed to practice standard techniques and methods, providing opportunities to reinforce foundational skills.

Aesthetically Pleasant Problem

Values: Yes, Perhaps, No

The beauty of a geometric problem can manifest in various ways — through its formulation, the underlying fact, the elegance of the idea, or the synthesis of ideas. Even a seemingly complex problem can possess aesthetic potential. This characteristic is subjective, yet consensus often exists regarding the beauty of certain tasks, which are typically concise and leverage unique ideas.

A task can also be considered beautiful because there is a beautiful solution. A beautiful solution though, often is hard to generalize. Sometimes this is the only known task where the idea for such a solution can be used. A beautiful solution is almost always a short solution.

Educational problem

Values: Yes, Perhaps, No

Educational problems aim to teach typical problem-solving methods as encountered in school lessons, finals, and entrance exams. These problems effectively demonstrate the application of specific ideas and methods, enabling students to test and master particular techniques. While the "educational" label can apply broadly, these problems are essential for instructional purposes.

Competition Problem

Values: Yes, Perhaps, No

Competition problems encompass those offered at Olympiads and math contests. This category includes both well-known and lesser-known problems, the latter of which might be used to construct school or local competitions or assignments for battles and tournaments. Known problems help in preparing advanced students for higher-level competitions.

Formal problem

Values: Yes, Perhaps, No

The solution to a formal problem is straightforward conceptually but challenging to express formally. Each problem dictates its proof level, beyond which its geometric facts are deemed "obvious." Proving all related facts might lead to cumbersome solutions, underscoring the balance between thoroughness and simplicity.

Cumbersome problem

Values: Yes, Perhaps, No

Cumbersome problems feature solutions that are clear yet involve multi-tiered processes, extensive calculations, or numerous case studies. Such tasks, common in competitive exams, are methodologically valuable, developing computational endurance and strategic planning. They often comprise "nested problems" where solving each layer systematically is crucial.

Important problem

Values: Yes, Perhaps, No

An important problem effectively demonstrates a particular technique or method and is essential for studying related topics. Some of these problems highlight edge cases, less apparent scenarios, or alternative perspectives, fostering attentiveness and the ability to recognize problem-solving traps. Others are recommended to consolidate skills introduced by **key problems**, promoting optimal developmental dynamics in learning.

Attributes Determining Problem Type and Answer Form

These attributes, which are generally mutually exclusive, determine both the type of a problem and the expected form of its answer. Each attribute is assigned a Yes/No value:

- **Computational Problem**: Involves calculations to arrive at a numerical answer or quantitative result.
- **Geometric Construction**: Requires constructing figures, often using tools like a compass and ruler, to achieve a specific geometric configuration.
- **Geometric Proof**: Demands a logical argument to demonstrate the truth of a given geometric statement.
- **Locus Problem**: Concerns finding a set of points that satisfy certain conditions within a geometric space.
- **Finding Maximum or Minimum**: Involves identifying the largest or smallest value for a particular quantity under given constraints.
- **Problem on Cutting Geometric Figures**: Entails dividing a figure into specified parts or configurations.
- **Geometric Inequalities**: Focuses on proving or deriving inequalities involving geometric quantities.

References

1. R. K. Gordin and I. F. Sharygin, "Компьютерные технологии в обучении геометрии," 2000. [Online]. Available: https://olimpiada.ru/article/775

2. S. Lu, J. M. Han, J. Rute, et al., "AlphaGeometry: an AI system for the Olympiad-level geometry problem solving," Nature, vol. 629, pp. 313–319, 2024. [Online]. Available: https://doi.org/10.1038/s41586-023-06747-5

3. Y. Qian, J. Clune, C. Barrett, and J. Avigad, "Lean-auto: An Interface between Lean 4 and Automated Theorem Provers," arXiv preprint arXiv:2505.14929, 2025. [Online]. Available: https://arxiv.org/abs/2505.14929

4. N. Sinclair, M. G. Bartolini Bussi, M. de Villiers, K. Jones, U. Kortenkamp, A. Leung, and K. Owens, "Geometry education, including the use of new technologies: A survey of recent research," in Proc. 13th Int. Congress of Mathematical Education, G. Kaiser, Ed., Cham: Springer, 2017, pp. 277–287. [Online]. Available: https://doi.org/10.1007/978-3-319-62597-3_18

5. L. Murphy, K. Yang, J. Sun, Z. Li, A. Anandkumar, and X. Si, "Autoformalizing Euclidean Geometry," arXiv preprint arXiv:2405.17216, 2024. [Online]. Available: https://arxiv.org/abs/2405.17216

6. C. Song, Z. Wang, F. Pu, H. Wang, X. Lin, J. Liu, J. Li, and Z. Liu, "LeanGeo: Formalizing Competitional Geometry problems in Lean," arXiv preprint arXiv:2508.14644, 2025. [Online]. Available: https://arxiv.org/abs/2508.14644

7. L. de Moura, S. Kong, J. Avigad, F. van Doorn, and S. von Raumer, "The Lean theorem prover (version 4)," 2021. [Online]. Available: https://github.com/leanprover/lean4

8. J. Narboux, "Automated theorem proving in geometry: A survey," Mathematics in Computer Science, vol. 6, no. 3, pp. 329–349, 2012. doi: 10.1007/s11786-012-0119-4

9. The Coq Development Team, "The Coq Reference Manual (version 8.17)," 2023. [Online]. Available: https://coq.inria.fr/refman/

10. The Coq Development Team, "The Coq proof assistant (version 8.17)," 2023. [Online]. Available: https://github.com/coq/coq