



FGeo-SSS: A Search-Based Symbolic Solver for Human-Like Automated Geometric Reasoning

Xiaokai Zhang ¹ , Na Zhu ^{1,2} , Yiming He ^{1,2} , Jia Zou ^{1,2} , Cheng Qin ^{1,2} , Yang Li ¹ , Zhenbing Zeng ³ , and Tuo Leng ^{1,2,*} 

¹ School of Computer Engineering and Science, Shanghai University, Shanghai, China

² Institute of Artificial Intelligence, Shanghai University, Shanghai, China

³ College of Sciences, Shanghai University, Shanghai, China

* Correspondence: tleng@shu.edu.cn

Abstract: Geometric problem solving (GPS) has always been a long-standing challenge in the fields of automated reasoning. Existing works struggle to construct a consistent and extensible formal system. This paper is the second in a series of our works. Based on the geometry formalization theory and the FormalGeo geometric formal system, we have developed the *Formal Geometric Problem Solver* (FGPS) in Python, which can serve as an interactive assistant for verifying problem-solving processes and as an automated problem solver that utilizes a variable search-based method and strategy. FormalGeo models the process of GPS as a series of applications of theorems, defined using geometric predicate logic. FGPS is capable of parsing and executing geometric predicate logic and automatically performs relational reasoning and algebraic computation. Then the process of GPS is structured into a hypertree with conditions as hypernodes and theorems as hyperedges. Based on FGPS, we can achieve readable, traceable, and verifiable automated solutions for geometric problems. We also have annotated the *formalgeo7k* dataset, which contains 6,981 geometry problems with detailed formal language descriptions and solutions. Experiments validate the correctness and utility of the FGPS. The forward search method combined with random search strategy achieves a 39.71% problem-solving success rate. The project is available at [here](https://github.com/tao-leng).

Keywords: Formal mathematics; Geometric problem solving; Geometric symbolic solver

1. Introduction

Geometric Problem Solving (GPS) has always been a long-standing challenge [1–3] in the fields of mathematical reasoning and Artificial Intelligence (AI), owing to the cross-modal forms of knowledge and the absence of automated solving methods. As depicted in Fig. 1, a geometry problem typically consists of geometric texts and images. The process of GPS can be seen as the application of theorems, where each application derives new conditions, ultimately leading to the derivation of the solution objective. Traditional methods of GPS can generally be divided into three categories [4]: synthesis methods, algebraic methods and invariant-based point elimination methods.

Synthesis methods, such as backward search method [5], forward chaining method [6] and deductive database method [7], is essentially a search-based method, which requires humans to predefine the search space or provide the system with a priori knowledge, namely theorems. Meanwhile, geometric problems need to be highly formalized to suit search formats. Theoretically, given a complete formal system and a priori knowledge, the search-based solver could provide solutions to any problem. However, in practice, we cannot determine if the given priori knowledge is complete. Additionally, due to the exponential explosion characteristic of search-based methods, it's impossible to provide a correct solution within reasonable memory and time constraints.

Algebraic methods based on coordinates, such as Wu's method [8], transform GPS into a computational problem, where geometric conditions of the problem are first con-

Citation: Xiaokai, Z.; Na, Z.; Yiming, H.; Jia, Z.; Cheng, Q.; Yang, L.; Zhenbing, Z.; Tuo, L. FGeo-SSS: A Search-Based Symbolic Solver for Human-Like Automated Geometric Reasoning. *Symmetry* **2024**, *1*, 0. <https://doi.org/>

Received:

Revised:

Accepted:

Published:

Copyright: © 2024 by the authors. Submitted to *Symmetry* for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

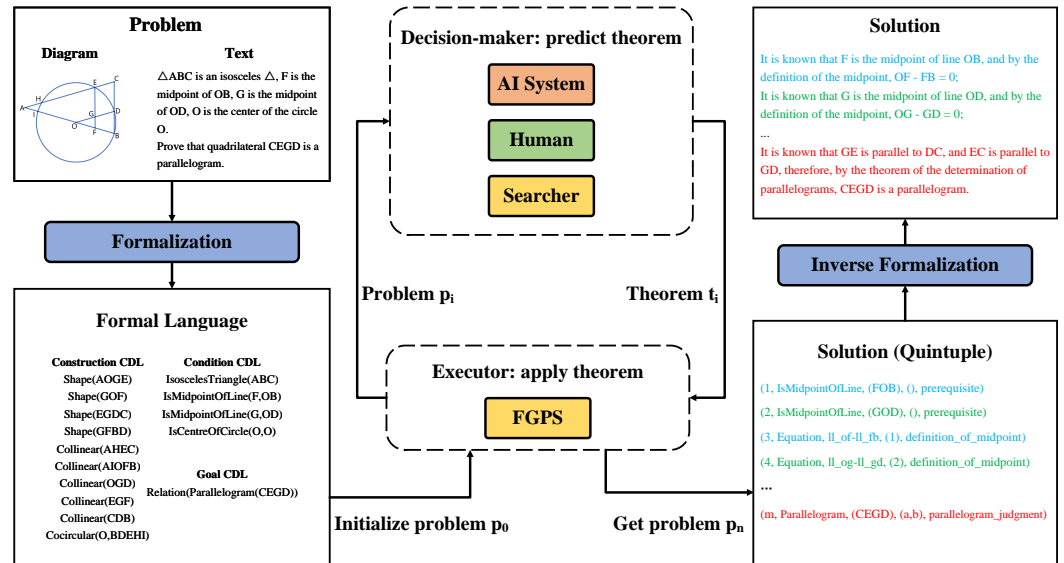


Figure 1. Interactive geometric problem solving based on FGPS.

verted into algebraic forms. Then, by utilizing the properties of algebraic computation, these algebraic expressions are simplified and solved, transforming complex algebraic expressions into forms understandable by humans, and thus interpreting their geometric meanings. These methods fully leverage the computational power of computers, but their problem-solving processes are not easily comprehensible to humans.

Invariant-based point elimination methods[9] finds that the solution methods for geometric problems are embedded in their geometric shape construction. This is a mechanized method but requires a lot of effort in discovering new invariants and shape construction, and the types of geometric problems that can be solved are limited.

The rapid development of AI has provided new ideas for GPS. New neural network architectures have given models the ability to model geometric knowledge. New learning and training methods have made it easier for people to teach computers geometric priori knowledge. Hence, AI-assisted synthesis methods, i.e., heuristic search methods, have attracted much attention from researchers. Such works can generally be divided into two categories [10]: symbolic approaches [11] and probabilistic approaches [12]. Symbolic methods require the prior construction of a formal system, and then geometric problem texts and images are parsed into a unified formal language. The AI predictor predicts theorems that may be needed in problem solving according to the formal language. The formal reasoner applies theorems and updates the known conditions of the problem. Direct theorem prediction can be modeled as a sequence generation task, and step-by-step theorem prediction can be modeled as a multi-classification task. Probabilistic methods model GPS as a sequence generation task with multimodal inputs, directly encoding geometric problem texts and images into a unified internal representation, then learning from human examples to generate problem-solving programs. These programs are subsequently executed by an executor to obtain the solution.

Nevertheless, both symbolic and probabilistic methods focus on the study of learning methods and model structures, neglecting the study of formal systems. Existing formal systems are roughly implemented using programming languages, and defining new predicates and theorems requires changing the solver's code, making it difficult to expand. The reasoning process cannot record process information, making the problem-solving process unreadable and unverifiable by humans. There is little research on the formal theory of geometry, and no systematic sorting of the structure and knowledge in the geometric field. These shortcomings severely limit the representational ability of formal systems, making it almost impossible to solve higher difficulty problems, such as International Mathematical

Olympiad (IMO) level geometric problems [13]. There is an urgent need for research on geometric formal systems and solvers.

Based on FormalGeo [14], We construct a search-based symbolic solver to addresses these issues. FormalGeo is a geometric formal system built on the basis of geometry formalization theory, featuring good readability. The syntax of the geometric formal language is similar to predicate logic and close to natural language, providing excellent readability and establishing a bridge for communication between humans and computers. The geometric formal language includes Geometric Definition Language (GDL) and Condition Declaration Language (CDL). FGPS parses GDL and CDL to configure the formal system and input geometric problem. FormalGeo transforms the process of GPS into a series of applications of theorems, defined using Geometric Predicate Logic (GPL). FGPS can parse and execute GPL, achieving traceable geometric relational reasoning and algebraic equation solving. The known conditions of a geometric problem are stored internally in FGPS as quintuples (condition ID, condition type, condition body, premises, and theorem). Based on the premises and theorem of conditions, we reorganize and structure them, transforming the process of GPS into a hypertree with conditions as hypernodes and theorems as hyperedges. Thus, we achieve formal language input of problems and structured output of solutions, ultimately realizing readable, traceable, and verifiable solutions for geometric problems.

FGPS can run in two modes: interactive and automated. As shown in Fig. 1, in the interactive mode, an external decision-maker is required to provide the next theorem information. FGPS applies the theorem and updates the problem status. When the external decision-maker is a human, FGPS can serve as an interactive proof assistant to help humans verify the proof process; when the decision-maker is AI, it can conveniently implement heuristic search methods. When FGPS operates in automatic mode, it can use various methods (forward search, backward search and heuristic search) combined with various search strategies (breadth-first search, depth-first search, random search and random beam search) to solve problems, automatically removing redundant theorems after achieving the solution goal.

Our contributions can be summarized as follows:

- We have crafted FGPS in Python. It serves as both an interactive assistant for verifying problem-solving processes and an automated problem solver, utilizing various methods and strategies. FGPS incorporates functions such as formal statement parsing, condition validity checks, automatic diagram construction, and condition expansion. It is capable of performing readable, traceable and verifiable algebraic equation solving and relational reasoning.
- We've annotated the formalgeo7k dataset, which contains 6,981 (expand to 133,818 using data augmentation method). Each problem comprises a complete natural language description, geometric shapes, formal language annotations, and theorem sequences annotations. Additionally, we have attempted to annotate 18 geometry problems of IMO-level, forming the dataset formalgeo-imo.
- We conducted experiments on formalgeo7k, comparing different search methods and strategies in terms of their problem-solving success rate, search time, and search step. The forward search method combined with random search strategy achieves a 39.7% problem-solving accuracy rate.

2. Related Works

Gelernter et al. developed the pioneering automated GPS system known as the Geometry Theorem Prover [5], which employed a backward search approach to solve pre-formalized problems. Nevins pointed out that the forward chaining method [6] can also be effective by efficiently representing the known conditions of the problem and limiting the typical application of those conditions. The development of geometry problem solving has led to the emergence of various downstream tasks, including geometry problem formalization [15,16], geometric knowledge extraction [17–21], geometric diagram parsing [22–24], geometric theorem proving [25–27], and geometry problem solving [28–33].

Wen-Tsun proposed the Wu's Method [8], which transforms geometric problem into a system of algebraic equations consisting of polynomials and inequalities and leverages various algebraic techniques to solve these equations. The study of algebraic approaches to geometry problems has given rise to a range of research achievements, such as Buchberger's Gröbner bases method [34], numerical parallel methods [35], polynomial system triangulation elimination algorithm [36], cylindrical algebraic decomposition for solving inequalities [37], dimensionality reduction methods [38], and software tools like GEOTHER [39].

Zhang proposed the point elimination method based on geometric invariants [9]. This approach employs constructive methods to describe problems and is capable of generating concise and meaningful readable proofs for a large number of non-trivial geometric problems. Subsequently, research on machine proofs of geometric theorems based on geometric invariants rapidly advanced [40–42], leading to the development of practical software tools such as Geometry Explorer [43], Geometry Expert [44] and Java Geometry Expert [45]. The method based on geometric invariant can also be extended to solid geometry [46] and non-Euclidean geometry [47].

GPS has been gaining more attention in the natural language processing community recently. Several geometry formal systems and datasets have been constructed, such as Geometry3K [11], GeoQA [12], GeometryQA [48]. Geometry3K [11] translates the known conditions of geometric problems into formal statements, defining theorems as a set of rules for converting between formal statements. This approach, referred to as Formal Language, is also used in GeoRE [19], which focuses on geometric relation extraction, and PGDP5k [49], which is designed for geometric image parsing. While these methods are intuitive, they lack theoretical guidance, are not comprehensive, and are not easily extensible with additional predicates and theorems. GeoQA [12] employs the formal method of Program, transforming the process of GPS into a sequence of programs consisting of variables and operators. Executing this program sequence yields the solution. Subsequent work extended the number and types of questions and rules, resulting in GeoQA+ [50], UniGeo [51], and PGPS9K [52]. These formal methods can represent algebraic and symbolic problem-solving processes, but compared to formal language methods, they are less intuitive and cannot represent traditional solutions. Additionally, adding new rules requires modifying the solver's code, making them less extensible. GeometryQA [48] employs a formal method known as the Expression Tree, which transforms the problem-solving process into a solving tree composed of operators and variables. This method is similar to the Program approach but is more structured. Shared benchmarks and datasets have significantly advanced research in AI-assisted GPS. Several AI systems, such as CL-based model [53], SCA [54], GeoDRL [10] and LANS [55], have been constructed to achieve higher problem-solving success rates.

3. FGPS

In this section, we introduce the specific implementation of FGPS. FGPS is built upon Geometry Formalization Theory and the FormalGeo formal system [14].

3.1. Architecture

As illustrated in Fig. 2, FGPS can be divided into 5 components:

- **Main** is control module of FGPS, invoking other modules to enable interactive problem-solving and automated problem-solving. The automated solving component implements both forward search and backward search, allowing for the configuration of various search strategies and defining interfaces for AI-assisted searches.
- **Engine** is the core component of FGPS, responsible for parsing and executing GPL and consists of two sub-modules, GPL Executor for relational reasoning and Equation Solver for algebraic computation.
- **Parser** facilitates bidirectional conversion between formal language and machine language. It consists of 3 sub-modules. GDL Parser parses GPD and GTDL into

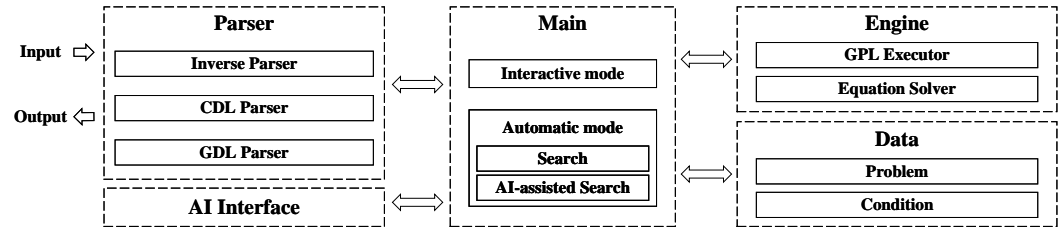


Figure 2. The components of FGPS.

machine language, enabling custom configuration of the Solver. CDL Parser parses the formal describing of problems into machine language for subsequent reasoning. Inverse Parser translates machine language back into formal language, facilitating the verification and checking of the solution process.

- **Data** preserves all details of the problem-solving process and comprises 2 sub-modules. The Problem module ensures the correctness and consistency of the problem input conditions, implementing automatic diagram construction, condition auto-expansion, and validity checks. The Condition module is responsible for data storage.
- **AI Interface** defines the interface for interaction between the AI system and FGPS. Both the AI Automatic Formalization and the AI Problem Solver can be seamlessly integrated with FGPS.

Guided by Geometry Formalization Theory and modular design, FGPS boasts excellent extensibility, allowing for easy customization of the formal system and the definition of external interfaces.

3.2. GPL Executor

The process of GPS can be represented as a sequence of theorem applications and theorems are defined using GPL. As a result, the process of GPS within FGPS is essentially the execution of GPL. GPL statements can consist of multiple logical conjunction words, geometric relations, and quantitative relations nested together. The application process can be divided into 3 steps:

1. In the GPL parsing phase, the solver expands complex GPL statements into Disjunctive Normal Form (DNF) using the distributive law. Each simple conjunction represents a branch of the theorem. This not only meets the requirements for backward reasoning and facilitates the generation of sub-goals but also speeds up theorem execution by skipping irrelevant branches.
2. In the GPL ordering phase, for each branch of the theorem, the solver adjusts the positions of geometric relations and quantitative relations within simple conjunctions according to the commutative law. The guiding principles for this adjustment are as follows: 1.Transforming relation composition into geometric constraints 2.Moving geometric constraints forward 3.Moving algebraic constraints backward. This approach not only helps filter out geometric relation elements that do not comply with the constraints, preventing the explosion of combinatorics caused by Cartesian product operations, but also reduces the number for algebraic equation solving, thereby improving theorem application speed.
3. In the GPL execution phase, the solver reads geometric and quantitative relations sequentially and performs relational reasoning in the order of their appearance.

The GPL execution process can be illustrated with an example. Suppose that we have a theorem defined as shown in Eq. 1, which includes 5 geometric relations $R_1(v_1, v_2)$, $R_2(v_2, v_3)$, $R_3(v_2)$, $R_4(v_2, v_3)$ and $R_5(v_2)$ and 1 quantitative relation $R_A(v_1, v_2)$.

$$R_1 \& (R_2 | (\sim R_3 | R_A) \& R_4 \& R_5) \quad (1)$$

During the GDL parsing phase, it is expanded into a DNF according to the distributive law, as shown in Eq. 2. This DNF consists of 3 simple conjunctions, with each simple conjunction serving as a theorem branch.

$$R_1 \& R_2 | R_1 \& \sim R_3 \& R_4 \& R_5 | R_1 \& R_A \& R_4 \& R_5 \quad (2)$$

In the GDL reordering phase, let's take branch $R_1 \& R_A \& R_4 \& R_5$ as an example. It adjusts the order of its statements according to the commutative law, resulting in the form shown in Eq. 3.

$$R_1 \& R_5 \& R_4 \& R_A \quad (3)$$

In the GPL execution phase, the GDL statements are read and executed in order, and the process is as shown in Eq. 4.

$$R_1 \& R_5 \& R_4 \& R_A \rightarrow R_{1,5} \& R_4 \& R_A \rightarrow R_{1,5,4} \& R_A \rightarrow R_{1,5,4,A} \quad (4)$$

3.3. Core Engine

3.3.1. Traceable Geometric Relational Reasoning

In the previous section, we transformed the execution process of GPL into two most fundamental operations, $R_1 \& R_2$ and $R_1 \& R_A$. R_1 and R_2 are geometric relations, and R_A is an quantitative relation. Assume the existing geometric relations $R_1(a, b) = \{(X, Y), (M, N)\}$, $R_2(b, c) = \{(Y, Z)\}$, quantitative constraint $R_A(a, b) : Equal(Attr(a), Attr(b))$ and the known algebraic equations being $\{attr_m - attr_n = 0\}$.

For the $R_1 \& R_2$ type of operation, first, a Cartesian product operation is executed to obtain preliminary reasoning results R' . Subsequently, intersection is performed on their point variables to obtain common variables, and each value of the common variables in R' is checked for correctness, filtering out the elements that meet the requirements. Then, a union of point variables is taken to reorganize the reasoning results.

Take $R_1(a, b) \& R_2(b, c)$ as example. First, perform the Cartesian product operation:

$$R_1(a, b) \& R_2(b, c) = \{(X, Y, Y, Z), (M, N, Y, Z)\} \quad (5)$$

Filter out the elements based on common variables:

$$R'(a, b, b, c) = \{(X, Y, Y, Z)\} \quad (6)$$

remove duplicates variables:

$$R_{1,2}(a, b, c) = \{(X, Y, Z)\} \quad (7)$$

For the $R_1 \& R_A$ type of operation, the intersection of point variables is taken to obtain common variables, and then the values of these common variables are substituted into the quantitative relation R_A , filtering out the elements that conform to the algebraic constraints.

Take $R_1(a, b) \& R_A(a, b)$ as example. First, construct equations:

$$R_1(a, b) \& R_A(a, b) = \{attr_a - attr_b = 0, attr_m - attr_n = 0\} \quad (8)$$

Construct a system of equations based on the known algebraic equations and check whether the equations hold true.

$$R'_A(a, b) = \{attr_m - attr_n = 0\} \quad (9)$$

Filter the elements in R_1 based on whether the equations hold true.

$$R_{1,A}(a, b) = \{(M, N)\} \quad (10)$$

In the above reasoning process, the premises of the new relations are recorded synchronously, achieving traceable geometric relational reasoning.

3.3.2. Minimum Dependency Equations Constructing

The known conditions of geometric problems can be categorized into geometric and quantitative relationships. Quantitative relationships eventually represented as a set of algebraic equations or inequalities. When performing algebraic constraint in the execution of GPL, the satisfaction of algebraic constraint under the known algebraic equations or inequalities of the problem is checked.

Algebraic constraints can be transformed into algebraic expressions represented by a , creating the target equation $g - a$. Among the several known equations X in the problem conditions, those relevant to $g - a$ are selected to construct the target equation group G , which is subsequently solved. If $g = 0$ is obtained as a solution, the algebraic constraints are satisfied. Typically, only a few equations in X are related to $g - a$, and this subset of equations is referred to as the minimum dependency equations.

The solving of equations accounts for the majority of the time spent in the entire process of solving geometric problems. Accelerating the equation solving process is crucial for enhancing the speed of geometric problem solving. To address this, we propose a method for constructing the minimum dependency equations. Without loss of generality, we examine the intermediate process of constructing G . At time t ($t = 1, 2, \dots$), G_t contains t equations and m unknowns, with the set of unknowns denoted as M_t . We need to select a candidate equation x_t from X to add to G_t in a way that increases the likelihood of obtaining a solution for the unknown g . The set of unknowns in x_t is represented as B_t . This process is repeated until $|M_t| = t$ or no new equations can be added.

$$|B_t \cap M_t| > 0 \quad (11)$$

$$\min(|B_t - M_t|) \quad (12)$$

$$\max(|B_t \cap M_t|) \quad (13)$$

The selection criteria for x_t are as follows:

1. B_t must intersect with M_t , as shown in Eq. 11. If they do not intersect, it implies that x_t is unrelated to G_t .
2. Under the condition of satisfying Eq. 11, adding x_t should introduce as few new unknown variables as possible, as depicted in Eq. 12. The closer the number of t and $|M_t|$ are, the higher the likelihood of solving G_t . In the initial stages of constructing G , which only contains $g - a$, $M_1 - 1 > 1$. The number of Added equation each time is a fixed value 1. If we aim to minimize the gap between t and $|M_t|$, we should try to introduce as few new equations when selecting x_t .
3. Under the condition of satisfying Eq. 11 and Eq. 12, the equation to be added should encompass more unknown variables, as demonstrated in Eq. 13. These additional unknown variables are often associated with other equations within G_t , providing more choices for simplifying G_t . If there are multiple equations that satisfy these conditions, we can choose any of them at random.

3.4. GPS methods

By invoking FGPS's core modules, we've developed both an interactive solver and a search-based problem solver. Next, we will introduce the forward search algorithm and the backward search algorithm.

Forward search (FW) starts from the known conditions of the problem and continuously apply theorems to derive new conditions until the goal is achieved. The search process involves the construction of a search tree, with nodes representing sets of known

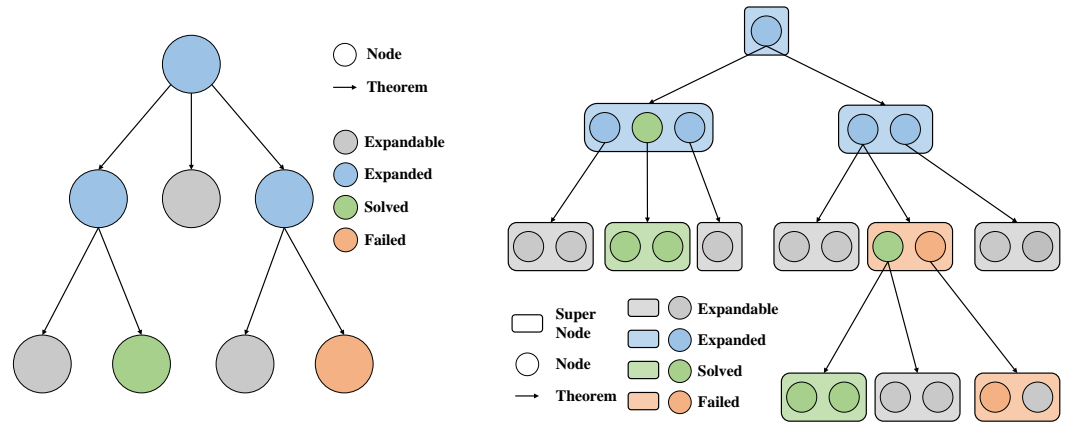


Figure 3. Forward search Tree (left). Backward search Tree (right).

conditions and edges denoting theorems, as depicted in Fig. 3. The description of the forward search algorithm is provided in Alg. 1. The function *get_expandable()* traverses the search tree based on pre-defined strategies (BFS, DFS, RS and RBS) and returns nodes with the EXPANDABLE state. The function *apply_theorem()* applies the theorem associated with the current node and returns whether the problem solved. The function *get_theorem_seqs()* returns a list of theorems applied from the root node to the current node. The function *expand()*, guided by the known conditions of the current node, checks the list of applicable theorems and extends new nodes.

Algorithm 1 Forward search

Input: *tree*: a tree with the known problem conditions as the root node.

Output: *theorem_seqs*: list of theorem sequence for problem solving.

Initialize an list *theorem_seqs*

node \leftarrow *tree.get_expandable()*

while *node* is not None **do**

solved \leftarrow *node.apply_theorem()*

if *solved* **then**

node.state \leftarrow SOLVED

theorem_seqs \leftarrow *node.get_theorem_seqs()*

break

end if

node.state \leftarrow EXPANDED

l \leftarrow *node.expand()*

if *l* = 0 **then**

node.state \leftarrow FAILED

end if

node \leftarrow *tree.get_expandable()*

end while

Backward search (BW), on the other hand, begins with the problem-solving goal, expands it into multiple sub-goals, and repeats this process until all sub-goals are resolved. The search process involves the construction of a search tree, with nodes representing subgoals, hypernodes representing sets of subgoals, and edges representing theorems, as illustrated in Fig. 3. The description of the backward search algorithm is provided in Alg. 2. The function *get_expandable()* traverses the search tree based on pre-defined strategies, returning hypernodes with the EXPANDABLE state. The function *node.check()* updates the state of the superNode based on the known problem conditions, while the function *super_node.check()* updates its own state based on the states of its nodes. The function *expand()* extends the current goal into several subgoals based on the list of theorems. The function *update()* propagates the state update from child nodes to parent nodes, starting

from the leaves and progressing up to the root. The function `get_theorem_seqs()` provides a list of theorems applied from the current node to the root node.

Algorithm 2 Backward search

Input: *super_tree*: a super tree with the problem goal as the root super node.

Output: *theorem_seqs*: list, theorem sequence for problem solving.

Initialize an list *theorem_seqs*

super_node \leftarrow *super_tree.get_expandable()*

while *super_node* is not *None* **do**

for *i* = 1 to *len*(*super_node.nodes*) **do**

node \leftarrow *super_node.nodes*[*i*]

node.check()

if *node.state* is SOLVED **then**

 continue

else if *node.state* is FAILED **then**

 break

else

l \leftarrow *node.expand()*

if *l* = 0 **then**

node.state \leftarrow FAILED

 break

else

node.state \leftarrow EXPANDED

end if

end if

end for

super_node.check()

super_tree.update()

if *super_tree.solved* **then**

theorem_seqs \leftarrow *super_tree.get_theorem_seqs()*

 break

end if

super_node \leftarrow *super_tree.get_expandable()*

end while

4. Datasets

Most of the existing datasets for GPS suffer from the following issues: 1.Limited data volume or non-open source availability. 2.Lack of annotations or incomplete and low-quality annotations. 3.Absence of formalization theory support, resulting in incoherent and inconsistent formal systems. 4.Low scalability, Defining new predicates and theorems require solver's code modifications. 5.Lower difficulty level of the problems. To address the aforementioned issues, we have annotated formalgeo7k and formalgeo-imo.

Our data is collected from various sources, including Geometry3k [11], GeoQA [12], GeoQA+ [50], and online resources. We carefully curated, classified, deduplicated, and standardized the problem statements. The creation of the formalgeo7k involved 16 trained master's students over a period of around 13 weeks. The creation of the formalgeo-imo involved 4 trained master's students over a period of around 1 week. Excluding the time spent on collaboration and dataset allocation, annotating datasets took approximately 1000 person-hours.

formalgeo7k comprises 6981 geometric problems that are accompanied by natural language descriptions, geometric diagrams, formal language annotations, and solution theorem sequence annotations. A annotated problem is illustrated in in Fig. 1 (omit the theorem annotations). The problem-solving process can be represented as a hypertree with conditions as hypernodes and theorems as hyperedges. The solution theorem sequence is a path from the root node (known conditions) to a leaf node (the problem-solving

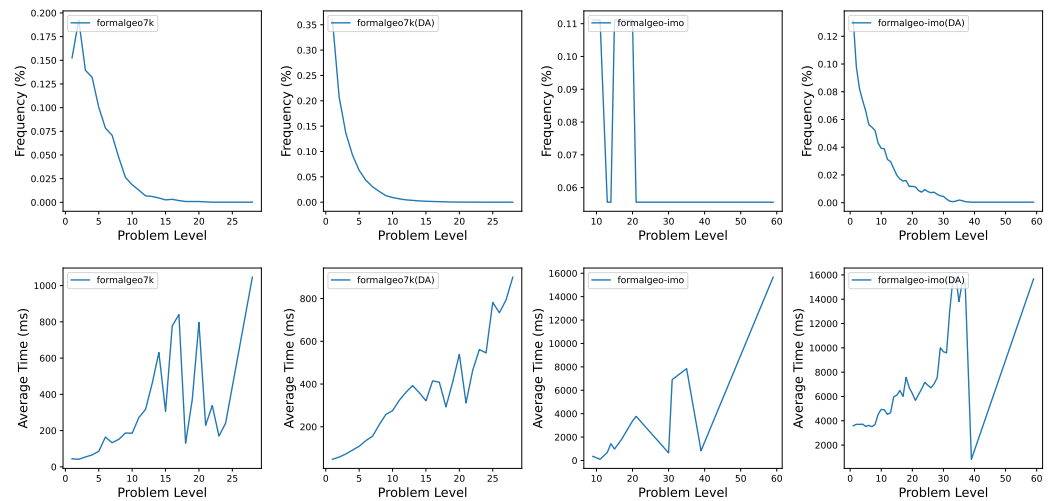


Figure 4. Distribution of problem (the top 4). Average Time of interactive verification (the bottom 4). DA represents data augmentation.

objective). By selecting any intermediate node along this path as the problem-solving objective, we can generate new problems, allowing us to expand the problem number to 133,818. formalgeo-imo is constructed with the same standards but with more challenging problem difficulty.

We use the length of theorem sequences required for problem solving as a rough metric for assessing problem difficulty. All annotated and expanded problems have been verified by the solver, and their average solution times varying with problem difficulty also show in the Fig. 4. The number of questions with a difficulty level of 15 or higher in formalgeo7k is quite small, leading to significant fluctuations. After data augmentation, datasets exhibit a larger scale of data and a smoother difficulty curve. In general, more challenging problems require longer solving time.

5. Experiments

We conducted experiments on the formalgeo7k, comparing different search methods and strategies in terms of problem-solving success rate, solution time, and the number of steps required for problem-solving.

Forward search (FW) starts from the known conditions of the problem and continuously apply theorems to derive new conditions until the goal is achieved. Backward search (BW), on the other hand, begins with the problem-solving goal, expands it into multiple sub-goals, and repeats this process until all sub-goals are resolved.

The search-based methods construct a search tree during the problem-solving process. We have the flexibility to choose various strategies to traverse the search tree and reach the goal. Breadth-first search (BFS) begins by expanding the top-level nodes of the search tree and then proceeds layer by layer into the depth. Depth-first search (DFS) recursively selects nodes from the search tree from shallow to deep and continues this process. Random search (RS) randomly selects an expandable node at each stage of expansion. Beam search (BS) selects k nodes in each stage of expansion and can be viewed as a trade-off between BFS and RS.

We conducted experiments on 2 Intel i9-10900X, 1 AMD Ryzen 9 5900X, and 1 AMD Ryzen 9 7950X, running the search algorithms using multiple processes while maintaining a CPU utilization rate of 80%. The maximum search depth was set to 15, and the beam size was set to 20. The total duration of the experiments was approximately 3 days. When the timeout for each problem was 300 seconds, the best success rate for problem-solving was approximately 30%. When the timeout for each problem was increased to 600 seconds, the specific results are as follows.

Table 1. An overview of search results.

method	strategy	result (%)		
		solved	unsolved	timeout
FW	BFS	38.86	7.42	53.72
FW	DFS	36.16	9.80	54.05
FW	RS	39.71	9.07	51.22
FW	BS	25.28	38.72	36.00
BW	BFS	35.44	2.68	61.88
BW	DFS	33.73	2.42	63.84
BW	RS	34.05	2.65	63.30
BW	BS	34.39	12.86	52.74

Table 2. Results of success rates.

method	strategy	success rates (%)						
		total	l_1	l_2	l_3	l_4	l_5	l_6
FW	BFS	38.86	59.95	38.62	28.55	17.35	8.63	3.77
FW	DFS	36.16	55.75	40.04	22.94	12.38	7.03	4.11
FW	RS	39.71	59.24	40.04	33.68	16.38	5.43	4.79
FW	BS	25.28	46.12	22.60	13.47	5.83	2.88	0.34
BW	BFS	35.44	67.22	33.72	11.15	6.67	6.07	1.03
BW	DFS	33.73	65.93	30.82	8.90	6.55	5.11	0.68
BW	RS	34.05	66.64	31.66	8.66	5.83	4.47	0.68
BW	BS	34.39	67.10	31.35	9.46	6.31	5.75	1.03

An overview of search-based automated problem-solving results is presented in Tab. 1. The highest problem-solving success rate was achieved by forward random search, reaching 39.708%. Most of the remaining problems were due to timeouts. As timeout settings are extended and computational resources increase, the proportion of timeout problems is expected to decrease. The number of unsolved problems using beam search was significantly higher compared to other strategies. This is because when selecting k branches, beam search occasionally discards the correct branch. Other contributing factors may include code bugs, equation solving timeouts, and the omission of theorems related to trigonometric.

In accordance with the length of the theorems required for problem-solving, we roughly categorize the difficulty of the questions into 6 levels, denoted as l_1 ($length \leq 2$), l_2 ($3 \leq length \leq 4$), l_3 ($5 \leq length \leq 6$), l_4 ($7 \leq length \leq 8$), l_5 ($9 \leq length \leq 10$), l_6 ($length \geq 11$), with corresponding problem numbers of 2407, 1898, 1247, 824, 313 and 292. The success rates for solving geometric problems of varying difficulty are presented in Tab. 2. As problem difficulty increases, the success rate of problem-solving rapidly declines. This phenomenon can be attributed to the fact that search-based problem-solving methods exhibit exponential growth in solving time as the length of the theorem sequence increases, often resulting in timeouts before achieving the goal. For problems of lower difficulty, backward search demonstrate a relatively higher success rate, while forward search outperforms in the case of more challenging problems.

The efficiency of the problem-solving algorithm can be measured by the search time and step. The experimental results of search-based automated problem-solving algorithms on the formalgeo7k are presented in Fig. 5.

In terms of average search time, backward search is slightly better than forward search overall. For solved problems, the search time is roughly proportional to the difficulty of the problems when problems are of low difficulty. However, as the difficulty increases, the search time for both forward search and backward search decreases. On the one hand, this is because there are very few successfully solved high-difficulty problems, leading to significant statistical errors. On the other hand, when dividing the difficulty of problems, we only consider the length of the solution theorem sequence but do not consider the time required for each theorem execution. The solved high-difficulty problems are precisely

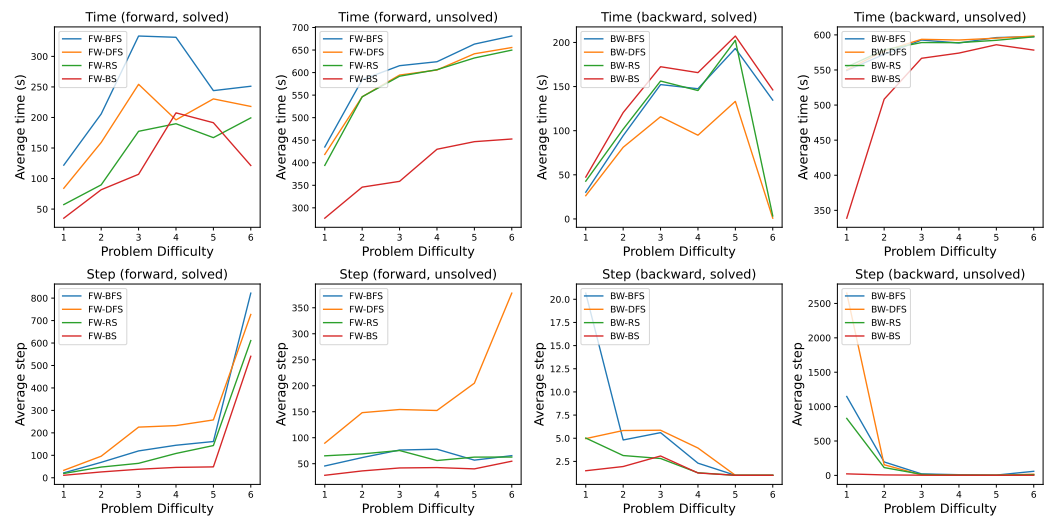


Figure 5. Average search time (the top 4). Average search step (the bottom 4).

those that require less solution time. For unsolved problems, the search time is roughly proportional to the difficulty of the problems.

Comparing different search strategies, it can be observed that in the forward search, BFS has a slightly lower success rate compared to the RS, but it takes the most time. BS has the lowest success rate but the least time consumption. For forward search, RS is the optimal strategy as it has the highest success rate and only slightly higher time consumption than BS that has the lowest success rate. In backward search, BFS is the optimal strategy, with the highest success rate and only slightly higher time consumption than DFS.

We observe a significant difference in the solution time of the BS strategy in backward search for solved and unsolved problems. This difference may be due to the characteristics of the backward search, where even if possible solution branches were discarded in previous steps, they may be reconstructed in later search steps. Therefore, as for BS of backward search, discarding potential solution branches does not lead to solution failure but takes longer search time.

Regarding the search step, forward search statistics are based on the number of nodes, while backward search statistics are based on the number of super nodes. Hence, they cannot be directly compared. The search step length in forward search is positively correlated with the difficulty of problems, while in backward search, it is negatively correlated with problem difficulty. The results of backward search are counterintuitive, and this could be because, for higher difficulty problems, the super nodes in backward search may contain more nodes, leading to increased time spent traversing a single super node and a reduction in the total number of traversed super nodes. Additionally, it can be observed that the search step length for unsolved problems in backward search is significantly higher than the average step length for solved problems. This is because, compared to forward search, backward search is less likely to halt, and it continues searching even if it misses a potential solution branch.

Comparing different strategies, DFS has the highest search step length, BS has the lowest search step length, and RS and BFS strategies have approximately the same average step length. For forward search, RS strategy is still the optimal strategy because it has the highest success rate and its search step length is only slightly higher than BS. Backward search does not exhibit a significantly superior strategy.

6. Conclusion

Based on the Geometry Formalization Theory and the FormalGeo geometric formal system, we construct FGPS, which can serve as an interactive assistant for verifying problem-solving processes and as an automated problem solver that utilizes a variable

search-based methods and strategies. Moreover, we annotate GPS datasets formalgeo7k and formalgeo-imo, which contains 6,981 geometry problems with detailed formal language descriptions and solutions. Experiments have demonstrated the correctness and efficiency of FGPS. In the future, we plan to further refine the formal system to annotate geometry problems at the IMO-level. We also plan to apply deep learning techniques to search tree pruning for the automatic solving of IMO-level geometric problems.

Author Contributions: Conceptualization, X.Z. and T.L.; Methodology, X.Z., N.Z., Y.H., J.Z. and T.L.; Software, X.Z.; Validation, X.Z., C.Q. and Y.L.; Writing—original draft preparation, X.Z.; Writing—review and editing, X.Z., Z.Z. and T.L.; Supervision, T.L.; Funding acquisition, T.L.. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by NSFC Grant.12071282.

Data Availability Statement: The project is available at [here](#).

Acknowledgments: We thank the reviewers for their helpful comments.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Daniel, S.; Leonardo, d.M.; Kevin, B.; Reid, B.; Percy, L.; Sarah, L.; Freek, W. IMO Grand Challenge, 2019.
2. XTXMarkets. Artificial Intelligence Mathematical Olympiad Prize (AIMO Prize), 2023.
3. Littman, M.L.; Ajunwa, I.; Berger, G.; Boutilier, C.; Currie, M.; Doshi-Velez, F.; Hadfield, G.; Horowitz, M.C.; Isbell, C.; Kitano, H.; et al. Gathering strength, gathering storms: The one hundred year study on artificial intelligence (AI100) 2021 study panel report. *arXiv preprint arXiv:2210.15767* **2022**.
4. Jingzhong, Z.; Yongbin, L. Automatic theorem proving for three decades. *Journal of Systems Science and Mathematical Sciences* **2009**, *29*, 1155.
5. Gelernter, H.L. Realization of a geometry theorem proving machine. In Proceedings of the IFIP congress, 1959, pp. 273–281.
6. Nevins, A.J. Plane geometry theorem proving using forward chaining. *Artificial Intelligence* **1975**, *6*, 1–23.
7. Chou, S.C.; Gao, X.S.; Zhang, J.Z. A deductive database approach to automated geometry theorem proving and discovering. *Journal of Automated Reasoning* **2000**, *25*, 219–246.
8. Wu, W.T. On the decision problem and the mechanization of theorem proving in elementary geometry. *Scientia Sinica* **1978**, *21*, 157–179.
9. Zhang, J.Z.; Chou, S.C.; Gao, X.S. Automated production of traditional proofs for theorems in Euclidean geometry I. The Hilbert intersection point theorems. *Annals of Mathematics and Artificial Intelligence* **1995**, *13*, 109–137.
10. Peng, S.; Fu, D.; Liang, Y.; Gao, L.; Tang, Z. GeoDRL: A Self-Learning Framework for Geometry Problem Solving using Reinforcement Learning in Deductive Reasoning. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2023, 2023, pp. 13468–13480.
11. Lu, P.; Gong, R.; Jiang, S.; Qiu, L.; Huang, S.; Liang, X.; Zhu, S.C. Inter-GPS: Interpretable Geometry Problem Solving with Formal Language and Symbolic Reasoning. In Proceedings of the 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), 2021, pp. 6774–6786.
12. Chen, J.; Tang, J.; Qin, J.; Liang, X.; Liu, L.; Xing, E.; Lin, L. GeoQA: A Geometric Question Answering Benchmark Towards Multimodal Numerical Reasoning. In Proceedings of the Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, 2021, pp. 513–523.
13. Trinh, T.H.; Wu, Y.; Le, Q.V.; He, H.; Luong, T. Solving olympiad geometry without human demonstrations. *Nature* **2024**, *625*, 476–482.
14. Zhang, X.; Zhu, N.; He, Y.; Zou, J.; Huang, Q.; Jin, X.; Guo, Y.; Mao, C.; Zhu, Z.; Yue, D.; et al. FormalGeo: The First Step Toward Human-like IMO-level Geometric Automated Reasoning. *arXiv preprint arXiv:2310.18021* **2023**.
15. Gan, W.; Yu, X. Automatic understanding and formalization of natural language geometry problems using syntax-semantics models. *International Journal of Innovative Computing, Information and Control* **2018**, *14*, 83–98.
16. Rao, Y.; Xie, L.; Guan, H.; Li, J.; Zhou, Q. A Method for expanding predicates and rules in automated geometry reasoning system. *Mathematics* **2022**, *10*, 1177.
17. Sachan, M.; Dubey, K.; Xing, E. From textbooks to knowledge: A case study in harvesting axiomatic knowledge from textbooks to solve geometry problems. In Proceedings of the Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 773–784.
18. Sachan, M.; Dubey, A.; Hovy, E.H.; Mitchell, T.M.; Roth, D.; Xing, E.P. Discourse in multimedia: A case study in extracting geometry knowledge from textbooks. *Computational Linguistics* **2020**, *45*, 627–665.

19. Yu, W.; Wang, M.; Wang, X.; Zhou, X.; Zha, Y.; Zhang, Y.; Miao, S.; Liu, J. Geore: A relation extraction dataset for chinese geometry problems. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021) Workshop on Math AI for Education (MATHAI4ED), 2021. 483–485
20. Huang, L.; Yu, X.; He, B. A Novel Geometry Problem Understanding Method based on Uniform Vectorized Syntax-Semantics Model. In Proceedings of the 2022 International Conference on Intelligent Education and Intelligent Research (IEIR). IEEE, 2022, pp. 78–85. 486–488
21. Zhou, W.; Xu, R.; Guan, H.; Zhao, J.; Rao, Y. Research on Geometry Problem Text Understanding Based on Bidirectional LSTM-CRF. In Proceedings of the 2022 9th International Conference on Digital Home (ICDH). IEEE, 2022, pp. 121–127. 489–490
22. Seo, M.J.; Hajishirzi, H.; Farhadi, A.; Etzioni, O. Diagram understanding in geometry questions. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2014, Vol. 28. 491–492
23. Zhang, M.L.; Yin, F.; Hao, Y.H.; Liu, C.L. Plane Geometry Diagram Parsing. In Proceedings of the Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22; Raedt, L.D., Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2022, pp. 1636–1643. Main Track, <https://doi.org/10.24963/ijcai.2022/228>. 493–495
24. Wong, M.F.; Qi, X.; Tan, C.W. EuclidNet: Deep Visual Reasoning for Constructible Problems in Geometry. In Proceedings of the 2nd MATH-AI Workshop at NeurIPS'22: Toward Human-Level Mathematical Reasoning, 2022. 496–497
25. Yu, X.; Wang, M.; Gan, W.; He, B.; Ye, N. A framework for solving explicit arithmetic word problems and proving plane geometry theorems. *International Journal of Pattern Recognition and Artificial Intelligence* **2019**, *33*, 1940005. 498–499
26. Gan, W.; Yu, X.; Zhang, T.; Wang, M. Automatically proving plane geometry theorems stated by text and diagram. *International Journal of Pattern Recognition and Artificial Intelligence* **2019**, *33*, 1940003. 500–501
27. Kovács, Z.; Yu, J.H. Automated Discovery of Geometrical Theorems in GeoGebra. *arXiv preprint arXiv:2202.04627* **2022**. 502
28. Seo, M.; Hajishirzi, H.; Farhadi, A.; Etzioni, O.; Malcolm, C. Solving geometry problems: Combining text and diagram interpretation. In Proceedings of the Proceedings of the 2015 conference on empirical methods in natural language processing, 2015, pp. 1466–1476. 503–505
29. Zhong, X.; Fu, H.; Yu, Y.; Liu, Y. Interactive learning environment based on knowledge network of geometry problems. In Proceedings of the 2015 10th International Conference on Computer Science & Education (ICCSE). IEEE, 2015, pp. 53–58. 506–507
30. Alvin, C.; Gulwani, S.; Majumdar, R.; Mukhopadhyay, S. Synthesis of geometry proof problems. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2014, Vol. 28. 508–509
31. Alvin, C.; Gulwani, S.; Majumdar, R.; Mukhopadhyay, S. Synthesis of solutions for shaded area geometry problems. In Proceedings of the The Thirtieth International Flairs Conference, 2017. 510–511
32. Sachan, M.; Xing, E. Learning to solve geometry problems from natural language demonstrations in textbooks. In Proceedings of the Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (SEM 2017), 2017, pp. 251–261. 512–513
33. Yu, X.; Gan, W.; Wang, M. Understanding explicit arithmetic word problems and explicit plane geometry problems using syntax-semantics models. In Proceedings of the 2017 International Conference on Asian Language Processing (IALP). IEEE, 2017, pp. 247–251. 514–516
34. Buchberger, B. Applications of Gröbner bases in non-linear computational geometry. *Mathematical aspects of scientific software* **1988**, pp. 59–87. 517–518
35. Yang, L.; Zhang, J.; Li, C. A prover for parallel numerical verification of a class of constructive geometry theorems. In Proceedings of the Proc. IWMM, 1992, Vol. 92, pp. 244–250. 519–520
36. Gao, X.S.; Chou, S.C. On the dimension of an arbitrary ascending chain. *CHINESE SCIENCE BULLETIN-ENGLISH EDITION* **1993**, *38*, 799–799. 521–522
37. Collins, G.E. Quantifier elimination for real closed fields by cylindrical algebraic decomposition—preliminary report. *ACM SIGSAM Bulletin* **1974**, *8*, 80–90. 523–524
38. Lu, Y. Practical automated reasoning on inequalities: Generic programs for inequality proving and discovering. *Proceedings of the Third Asian Technology Conference in Mathematics. Tsukuba, Japan* **1998**, pp. 24–35. 525–526
39. Wang, D. GEOTHER: A geometry theorem prover. In Proceedings of the Automated Deduction—CADE-13: 13th International Conference on Automated Deduction New Brunswick, NJ, USA, July 30–August 3, 1996 Proceedings. Springer, 2005, pp. 166–170. 527–528
40. Chou, S.C.; Gao, X.S.; Zhang, J.Z. Automated geometry theorem proving by vector calculation. In Proceedings of the Proceedings of the 1993 international symposium on Symbolic and algebraic computation, 1993, pp. 284–291. 529–530
41. Chou, S.C.; Gao, X.S.; Zhang, J.Z. A collection of 110 geometry theorems and their machine produced proofs using full-angles. *Washington State University, Washington* **1994**. 531–532
42. Li, H. Symbolic computation in the homogeneous geometric model with Clifford algebra. In Proceedings of the Proceedings of the 2004 international symposium on Symbolic and algebraic computation, 2004, pp. 221–228. 533–534
43. Wilson, S.; Fleuriet, J.D. Geometry Explorer: A tool for generating diagrammatic full-angle method proofs. In Proceedings of the Automated deduction in geometry: extended abstracts, 2006, pp. 144–150. 535–536
44. Chou, S.C.; Gao, X.S.; Zhang, J.Z. An introduction to geometry expert. In Proceedings of the CADE, 1996, Vol. 1104, pp. 235–239. 537
45. Ye, Z.; Chou, S.C.; Gao, X.S. An introduction to java geometry expert. In Proceedings of the Automated Deduction in Geometry: 7th International Workshop, ADG 2008, Shanghai, China, September 22–24, 2008. Revised Papers 7. Springer, 2011, pp. 189–195. 538–539
46. Chou, S.C.; Gao, X.S.; Zhang, J.Z. Automated production of traditional proofs in solid geometry. *Journal of Automated Reasoning* **1995**, *14*, 257–291. 540–541

47. Yang, L.; Gao, X.S.; Chou, S.C.; Zhang, J.Z. Automated production of readable proofs for theorems in non-Euclidean geometries. In Proceedings of the Automated Deduction in Geometry: International Workshop on Automated Deduction in Geometry Toulouse, France, September 27–29, 1996 Selected Papers 1. Springer, 1997, pp. 171–188. 542
48. Tsai, S.h.; Liang, C.C.; Wang, H.M.; Su, K.Y. Sequence to General Tree: Knowledge-Guided Geometry Word Problem Solving. In Proceedings of the Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2021, pp. 964–972. 543
49. Hao, Y.; Zhang, M.; Yin, F.; Huang, L.L. PGDP5K: A Diagram Parsing Dataset for Plane Geometry Problems. In Proceedings of the 2022 26th International Conference on Pattern Recognition (ICPR). IEEE, 2022, pp. 1763–1769. 544
50. Cao, J.; Xiao, J. An Augmented Benchmark Dataset for Geometric Question Answering through Dual Parallel Text Encoding. In Proceedings of the Proceedings of the 29th International Conference on Computational Linguistics, 2022, pp. 1511–1520. 545
51. Chen, J.; Li, T.; Qin, J.; Lu, P.; Lin, L.; Chen, C.; Liang, X. UniGeo: Unifying Geometry Logical Reasoning via Reformulating Mathematical Expression. *arXiv preprint arXiv:2212.02746* 2022. 546
52. Zhang, M.L.; Yin, F.; Liu, C.L. A Multi-Modal Neural Geometric Solver with Textual Clauses Parsed from Diagram. *arXiv preprint arXiv:2302.11097* 2023. 547
53. Jian, P.; Guo, F.; Wang, Y.; Li, Y. Solving Geometry Problems via Feature Learning and Contrastive Learning of Multimodal Data. *CMES-COMPUTER MODELING IN ENGINEERING & SCIENCES* 2023, 136, 1707–1728. 548
54. Ning, M.; Wang, Q.F.; Huang, K.; Huang, X. A Symbolic Character-Aware Model for Solving Geometry Problems, 2023, [arXiv:cs.CV/2308.02823]. 549
55. Zhang, M.L.; Li, Z.Z.; Yin, F.; Liu, C.L. LANS: A Layout-Aware Neural Solver for Plane Geometry Problem. *arXiv preprint arXiv:2311.16476* 2023. 550

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content. 551