Inter-GPS: Interpretable Geometry Problem Solving with Formal Language and Symbolic Reasoning

Pan Lu[1*], **Ran Gong**[1*], **Shibiao Jiang**[2*], **Liang Qiu**[1], **Siyuan Huang**[1],
Xiaodan Liang[3], **Song-Chun Zhu**[1,4,5,6]
[1]Center for Vision, Cognition, Learning and Autonomy, UCLA
[2]College of Computer Science and Technology, Zhejiang University
[3]School of Intelligent Systems Engineering, Sun Yat-sen University
[4]Beijing Institute of General Artificial Intelligence, [5]Peking University, [6]Tsinghua University
panlu@cs.ucla.edu, {nikepupu, liangqiu, huangsiyuan}@ucla.edu,
{jiangshibiao1999, xdliang328}@gmail.com, sczhu@stat.ucla.edu

Abstract

Geometry problem solving has attracted much attention in the NLP community recently. The task is challenging as it requires abstract problem understanding and symbolic reasoning with axiomatic knowledge. However, current datasets are either small in scale or not publicly available. Thus, we construct a new large-scale benchmark, Geometry3K, consisting of 3,002 geometry problems with dense annotation in formal language. We further propose a novel geometry solving approach with formal language and symbolic reasoning, called *Interpretable Geometry Problem Solver* (Inter-GPS). Inter-GPS first parses the problem text and diagram into formal language automatically via rule-based text parsing and neural object detecting, respectively. Unlike implicit learning in existing methods, Inter-GPS incorporates theorem knowledge as conditional rules and performs symbolic reasoning step by step. Also, a theorem predictor is designed to infer the theorem application sequence fed to the symbolic solver for the more efficient and reasonable searching path. Extensive experiments on the Geometry3K and GEOS datasets demonstrate that Inter-GPS achieves significant improvements over existing methods. [1]

1 Introduction

Geometry problem solving is a long-standing challenging task in artificial intelligence and has been gaining more attention in the NLP community recently (Seo et al., 2014; Hopkins et al., 2019; Sachan et al., 2020). Solving geometry problems is an essential subject in high-school education for the development of students' abstract thinking. As an example shown in Figure 1, given problem text in natural language and a corresponding diagram, one needs to identify the geometric relations, apply theorem knowledge, and conduct algebraic calculations to derive the numerical value of the answer.

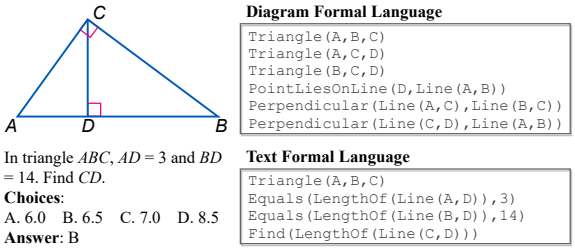


Figure 1: A data example in Geometry3K dataset. Each data is annotated with formal language descriptions.

Psychologists and educators believe that solving geometric problems requires high-level thinking abilities of symbolic abstraction and logical reasoning (Chinnappan, 1998; Nur and Nurvitasari, 2017). However, if algorithms take the raw problem content, it might encounter challenges to understand the abstract semantics and perform human-like cognitive reasoning for inferring the answer in the geometry domain. A formal language is composed of words from a well-formed alphabet based on a specific set of rules and is commonly used in the fields of linguistics and mathematics. Therefore, our proposed geometry solver parses the problem inputs into formal language descriptions (see examples in Figure 1) before solving the problems.

To translate the problem text and diagrams to formal descriptions, existing methods (Seo et al., 2015; Sachan et al., 2017; Sachan and Xing, 2017) highly depend on human annotations like symbols in diagrams as the intermediate results. Also, these methods fail to provide the explicit reasoning processes when predicting the answer. For example, (Seo et al., 2015) simplifies the problem solving task to an optimization problem to pick one that satisfies all constraints from choice candidates. Fur-

[*]Equal contribution.
[1]The project with code and data is available at https://lupantech.github.io/inter-gps.

thermore, most current datasets are either small in scale or not publicly available (Seo et al., 2015; Sachan and Xing, 2017), which further hinders the research of geometry problem solving.

To overcome these challenges, we first construct a new large-scale benchmark, called Geometry3K, to assess algorithms' performance of geometry problem solving. The Geometry3K dataset consists of 3,002 multi-choice problems as well as covers diverse geometric shapes and problem goals. In contrast with existing work, we also annotate each problem text and diagram with unified structural descriptions in formal language.

This paper further presents a novel geometry solving approach with formal language and symbolic reasoning, called *Interpretable Geometry Problem Solver* (Inter-GPS). Inter-GPS (Figure 4) develops an automatic parser that translates the problem text via template rules and parses diagrams by a neural object detector into formal language, respectively. In contrast to parameter learning, Inter-GPS formulates the geometry solving task as problem goal searching, and incorporates theorem knowledge as conditional rules to perform symbolic reasoning step by step. It demonstrates an interpretable way to tackle the task. Also, we design a theorem predictor to infer the possible theorem application sequence in Inter-GPS for the efficient and reasonable searching path. Extensive experiments on the Geometry3K and GEOS datasets show Inter-GPS achieves large improvements over existing methods.

Our contributions are three-fold: (1) we introduce a large-scale diverse benchmark of geometry problem solving, Geometry3K, which is densely annotated with formal language; (2) we develop an automatic problem parser to translate the problem text and diagram into formal language; (3) we propose a novel interpretable problem solver that applies symbolic reasoning to infer the answer.

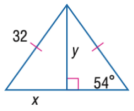
2 Related Work

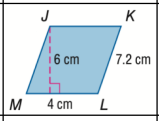
Datasets for Geometry Problem Solving. Several datasets for geometry problems have been released in recent years. These include GEOS (Seo et al., 2015), GEOS++ (Sachan et al., 2017), GeoShader (Alvin et al., 2017) and GEOS-OS (Sachan and Xing, 2017) datasets. However, these datasets are relatively small in scale and contain limited problem types. For example, there are only 102 shaded area problems in GeoShader and 186 problems in GEOS. While GEOS++ and GEOS-OS contain more data of 1,406 and 2,235 problems, respectively, they have not been publicly available yet. Instead, our Geometry3K dataset features 3,002 SAT-style problems collected from two high-school textbooks that cover diverse graph and goal types. Besides, each problem in Geometry3K is annotated with dense descriptions in formal language (defined in Section 3), which makes it particularly suited for symbolic reasoning and interpretable problem solving. In order to promote follow-up work in the geometry domain, we release the dataset and evaluation baselines.

Approaches for Geometry Problem Solving. Due to the sparsity of appropriate data, most early works on automated geometry systems focus on geometry theorem proving (Wen-Tsun, 1986; Chou et al., 1996; Yu et al., 2019; Gan et al., 2019), problem synthesis (Alvin et al., 2014), diagram parsing (Seo et al., 2014), as well as problem formalization (Gan and Yu, 2018). (Seo et al., 2015) attempt using computer vision and natural language processing techniques to solve geometry problems with problem understanding. However, the system does not perform explicit reasoning with axiomatic knowledge as it reduces the task to an optimization problem to see which choice can satisfy all constraints. Some recent efforts (Sachan et al., 2017, 2020) have been made to incorporate theorem knowledge into problem solving. They feed geometry axioms written as horn clause rules and declarations from the diagram and text parser into logical programs in prolog style to solve the problem. However, these methods fail to provide human-readable solving steps. And parameter learning on horn clause rules and built-in solvers leads to an uncontrollable search process. In contrast, our proposed Inter-GPS implements explicit symbolic reasoning to infer the answer without the help of candidate answers in an interpretable way.

Interpretable Math Problem Solving. Due to the intrinsic requirements of symbolic understanding and logical reasoning, interpretability of solvers plays an essential role in geometry problem solving. While the interpretability of geometry problem solvers is rarely explored, some pioneering work has been proposed in the general math problem solving domain. Broadly there are two main lines of achieving interpretable solving steps for math problems. The first generates intermediate structural results of equation templates (Huang et al.,

Problem Text	Diagram	Choices	Text Literals	Diagram Literals

Find y. Round to the nearest tenth.		A. 18.8
		B. 23.2
		C. 25.9
		D. 44.0
		Answer: C
			`Find(y)`	`Equals(LengthOf(Line(A,B)),32)`
				`Equals(LengthOf(Line(B,D)),y)`
				`Equals(MeasureOf(Angle(A,C,B)),54)`
				`Equals(LengthOf(Line(A,D)),x)`
				`PointLiesOnLine(D,Line(A,C))`
				`Perpendicular(Line(B,D),Line(C,D))`
				`Equals(LengthOf(Line(A,B)),LengthOf(Line(B,C)))`

Find the perimeter of $\parallelogram$ JKLM.		A. 11.2
		B. 22.4
		C. 24
		D. 44.8
		Answer: B
			`Find(PerimeterOf(Parallelogram(J,K,L,M)))`	`Equals(LengthOf(Line(L,K)),7.2)`
				`Equals(LengthOf(Line(M,L)),4)`
				`Equals(LengthOf(Line(E,J)),6)`
				`PointLiesOnLine(E,Line(M,L))`
				`Perpendicular(Line(J,E),Line(E,L))`

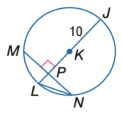
In $\odot$ K, MN = 16 and m $\widehat$ MN = 98. Find the measure of LN. Round to the nearest hundredth.		A. 6.93
		B. 7.50
		C. 8.94
		D. 10.00
		Answer: C
			`Circle(K)`
			`Equals(LengthOf(Line(M,N)),16)`
			`Equals(MeasureOf(Arc(M,N)),98)`
			`Find(LengthOf(Line(L,N)))`
				`Equals(LengthOf(Line(J,K)),10)`
				`Perpendicular(Line(P,K),Line(M,P))`
				`PointLiesOnLine(P,Line(M,N))`
				`PointLiesOnLine(P,Line(L,J))`
				`PointLiesOnLine(P,Line(L,K))`
				`PointLiesOnLine(K,Line(P,J))`
				`PointLiesOnLine(K,Line(L,J))`
				`PointLiesOnCircle(M,Circle(K))`
				`PointLiesOnCircle(J,Circle(K))`
				`PointLiesOnCircle(N,Circle(K))`
				`PointLiesOnCircle(L,Circle(K))`

Figure 2: More data examples in the Geometry3K dataset.

2017; Wang et al., 2019), operational programs (Amini et al., 2019) and expression trees (Wang et al., 2018; Qin et al., 2020; Hong et al., 2021). The second line of work with a higher level of interpretability translates the math problems into symbolic language and conducts logical reasoning iteratively to predict the final results (Matsuzaki et al., 2017; Roy and Roth, 2018). Furthermore, inspired by work on semantic parsing (Han and Zhu, 2005; Zhu and Mumford, 2006; Tu et al., 2014), we claim structured diagram parsing and joint semantic representations for text and diagrams is critical in interpretable geometry problem solving.

3 Geometry Formal Language

A geometry problem P is defined as a tuple $(t, d, \mathbf{c})$, in which t is the input text, d is the diagram image and $\mathbf{c} = \{c_1, c_2, c_3, c_4\}$ is the multiple-choice candidate set in the format of numerical values. Given the text t and diagram d, an algorithm is required to predict the correct answer $c_i \in \mathbf{c}$. We formally describe the problem in the geometric domain language Ω, a set of literals composed of predicates and arguments. Basic terms used in the geometry problem solver are defined as follows.

Definition 1. A *predicate* is a geometric shape entity, geometric relation, or arithmetic function.

Definition 2. A *literal* is an application of one *predicate* to a set of arguments like variables or constants. A set of *literals* makes up the semantic description from the problem text and diagrams in the formal language space Ω.

Definition 3. A *primitive* is a basic geometric element like a point, a line segment, a circle, or an arc segment extracted from the diagram.

Terms	Examples
predicate	`Line`, `IntersectAt`, `IsMedianOf`
literal	`Find(AreaOf(Triangle(A,B,C)))`

Table 1: Term examples in geometry formal language.

Table 1 lists examples of *predicates* and *literal* templates. There are 91 *predicates* in our defined formal language, and we list them in the Tables 10 to 15 in the Appendix Section.

4 Geometry3K Dataset

4.1 Dataset Collection

Most existing datasets for geometry problem solving are relatively small, contain limited problem types, or not publicly available. For instance, the GEOS dataset (Seo et al., 2015) only contains 186 SAT problems. Although there are 1,406 problems in GEOS++ (Sachan et al., 2017), this dataset has not been released to the public yet. Therefore, we build a new large-scale geometry problem benchmark, called Geometry3K. The data is collected from two popular textbooks for high school students across grades 6-12 by two online digital libraries (McGraw-Hill[2], Geometryonline[3]). Groups of well-trained annotators with undergraduate degrees manually collect each problem with its problem text, geometry diagram, four candidate choices, and correct answer. In order to evaluate the fine-grained performance of geometry solvers, we label each problem data with the corresponding problem goal and geometry shapes.

[2]https://www.mheducation.com/
[3]www.geometryonline.com

Dataset	#qa	#word	#shape	#goal	#var	grade	operator type
GeoShader (Alvin et al., 2017)	102	/	4	1	1	6-10	$\{+, -, \times, \div, \square^2, \sqrt{\square}\}$
GEOS (Seo et al., 2015)	186	4,343	4	3	1	6-10	$\{+, -, \times, \div, \square^2, \sqrt{\square}\}$
GEOS++ (Sachan et al., 2017)	1,406	/	4	3	1	6-10	$\{+, -, \times, \div, \square^2, \sqrt{\square}\}$
GEOS-OS (Sachan and Xing, 2017)	2,235	/	4	3	1	6-10	$\{+, -, \times, \div, \square^2, \sqrt{\square}\}$
Geometry3K (ours)	3,002	36,736	6	4	3	6-12	$\{+, -, \times, \div, \square^2, \sqrt{\square}, \sin, \cos, \tan\}$

Table 2: Comparison of our Geometry3K dataset with existing datasets.

	Total	Train	Val	Test
Questions	3,002	2,101	300	601
Sentences	4,284	2,993	410	881
Words	30,146	20,882	2,995	6,269
Literals (Text)	6,293	4,357	624	1,312
Literals (Diagram)	27,213	19,843	2,377	4,993

Table 3: Basic statistics of our Geometry3K dataset.

Predicates (Text)	%	Predicates (Diagram)	%
`Find`	19.00	`Line`	30.89
`Line`	14.49	`PointLiesOnLine`	16.66
`Equals`	11.83	`Equals`	15.17
`LengthOf`	9.53	`MeasureOf`	10.46
`MeasureOf`	8.97	`LengthOf`	8.69
.....		.....	
`CircumscribedTo`	0.05	`Triangle`	0.03
`SumOf`	0.04	`Quadrilateral`	0.02
`HeightOf`	0.04	`Kite`	0.01
`BaseOf`	0.04	`HeightOf`	0.01
`IsHypotenuseOf`	0.04	`Square`	0.01

Table 4: Most and least frequent predicates of formal descriptions in Geometry3K (frequency >5).

Unlike existing datasets that only collect the problem text and diagrams, we further annotate each data in Geometry3K with dense formal language descriptions that bridge the semantic gap between the textual and visual contents as well as benefit the symbolic problem solver. The annotated formal language is used to train and evaluate our proposed problem parsers. Data examples are illustrated in Figure 2.

4.2 Dataset Statistics

The Geometry3K dataset consists of 3,002 problems and is divided into the train, validation, and test sets with the ratio of 0.7:0.1:0.2, as shown in Table 3. Figure 3 illustrates the question distribution by the number of sentence words. The long tail in the distribution requires the geometry solvers to understand the rich semantics in the textual content.

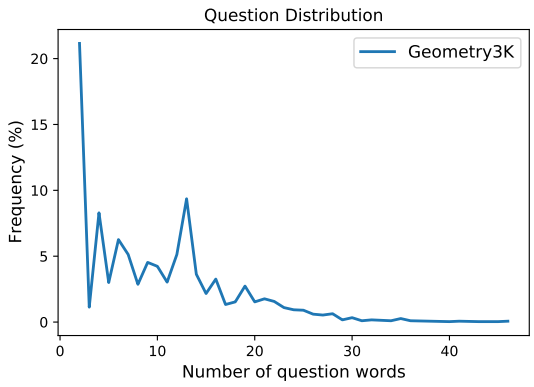


Figure 3: Question length distribution of Geometry3K.

There are 6,293 literals for the problem text and 27,213 literals for the diagrams in Geometry3K, respectively. We list the most and least frequent predicates with a frequency greater than 5 in Table 4. It is shown that the predicates for the problem text are more evenly distributed than those for diagrams. This is mainly because the problem text describes diverse geometric shapes, attributes, and relations while diagrams display the basic properties of points, lines, and arcs.

4.3 Comparisons with Existing Datasets

To the best of our knowledge, currently, it is the largest geometry problem dataset. We summarize the Geometry3K dataset's main statistics and a comparison of existing datasets in Table 2. In addition to four elementary shapes (lines, triangles, regular quadrilaterals, and circles) mentioned in that GEOS dataset, Geometry3K contains irregular quadrilaterals and other polygons. Besides, in Geometry3K, there are more unknown variables and operator types that may require equation solving to find the goal of the problem. Note that 80.5% of problems are solvable without the associated diagram in the GEOS dataset. By contrast, less than 1% of the problems in our Geometry3K dataset could be solved when the problem diagram is not provided. In general, the statistics and comparisons above show Geometry3K is challenging for geometry problem solvers.

4.4 Human Performance

As an intellectual task, it is necessary to know the human performance for geometry problems. We push the test-split data of the dataset in the crowd-

sourcing platform, Amazon Mechanical Turk[4]. Each eligible annotator must have obtained a high school or higher degree and is asked to answer 10 problems in 25 minutes. To ensure annotators solving the problem to the best of their ability, they are further asked to spend at least 7 minutes on the problem set and 10 seconds on each problem. We filter out annotators who do not satisfy the requirement. We also ask dozens of graduates majoring in science or engineering to answer these problems to evaluate human experts' performance. Table 5 shows the human performance. Compared to random guess's accuracy of 25%, humans achieve an overall accuracy of 56.9%, and human experts can achieve a good performance of 90.9%.

5 Geometry Problem Parser

Our proposed Inter-GPS takes the problem text and diagrams as inputs and translates them into formal language descriptions automatically via the text parser (Section 5.1) and the diagram parser (Section 5.2), respectively.

5.1 Text Parser

Given the word sequence of the problem text T, the text parser needs to translate it into a set of literals L_t, a sequence composed of predicates and variables. Recently, deep neural networks have achieved promising performances in sequence-to-sequence (Seq2Seq) learning tasks like machine translation (Sutskever et al., 2014; Vaswani et al., 2017; Devlin et al., 2018). However, semantic parsers using Seq2Seq learning methods are not feasible to generate satisfactory literals in the Geometry3K dataset for two reasons. Firstly, the limited scale of geometry datasets weakens these highly data-driven methods. Secondly, neural semantic parsers tend to bring noises in generated results while geometry solvers with symbolic reasoning are sensitive to such deviations.

Inspired by previous works (Koo et al., 2008; Seo et al., 2015; Bansal et al., 2014) that indicate the rule-based parsing method is able to obtain precise parsing results, we apply this approach with regular expressions to perform text parsing. We also achieve a semantic text parser using BART (Lewis et al., 2020), one of the state-of-the-art sequence learning models for comparison.

[4]https://www.mturk.com/

5.2 Diagram Parser

Diagrams provide complementary geometric information that is not mentioned in the problem text. Previous works (Seo et al., 2014, 2015) require manual annotations to identify symbols in the diagrams and fail to deal with special relational symbols such as *parallel*, *perpendicular*, and *isosceles*. Instead, an automatic diagram parser without human intervention is proposed in this work and is able to detect varied diagram symbols.

The diagram parser first applies Hough Transformation (Shapiro and Stockman, 2001) to extract geometry primitives (points, lines, arcs, and circles), following (Seo et al., 2015). Then the diagram symbols and text regions are extracted through a strong object detector RetinaNet (Lin et al., 2017), and the textual content is further recognized by the optical character recognition tool MathPix[5]. After obtaining the primitive set P and symbol set S, we need to ground each symbol with its associated primitives. (Seo et al., 2015) adapts a greedy approach where each symbol is assigned to the closest primitive without considering its validity. Instead, we formulate the grounding task as an optimization problem with the constraint of geometry relations:

$$\begin{aligned} &\min \sum_{s} dist(s_i, p_j) \times \mathbb{1}\{s_i \text{ assigns to } p_j\} \\ &\text{s.t. } (s_i, p_j) \in \text{Feasibility set } F, \end{aligned} \tag{1}$$

where the $dist$ function measures the Euclidean distance between the symbol s_i and primitive p_j. F defines the geometric constraints for symbol grounding. For example, the *parallel* symbol could only be assigned to two lines with the same slopes and the *perpendicular* symbol is only valid to two orthogonal lines.

6 Geometry Problem Solver

Unlike existing methods (Seo et al., 2015; Sachan et al., 2017; Alvin et al., 2017; Sachan et al., 2020), Inter-GPS achieves the explicit symbolic reasoning with the theorem knowledge base and the human-readable search process, shown in Figure 4.

6.1 Symbolic Geometry Solver

Overall, Inter-GPS takes the relation set $\mathcal{R}$ and the theorem knowledge base set $\mathcal{KB}$ as inputs, and outputs the numeric solution g^* of the problem goal g.

[5]https://mathpix.com/

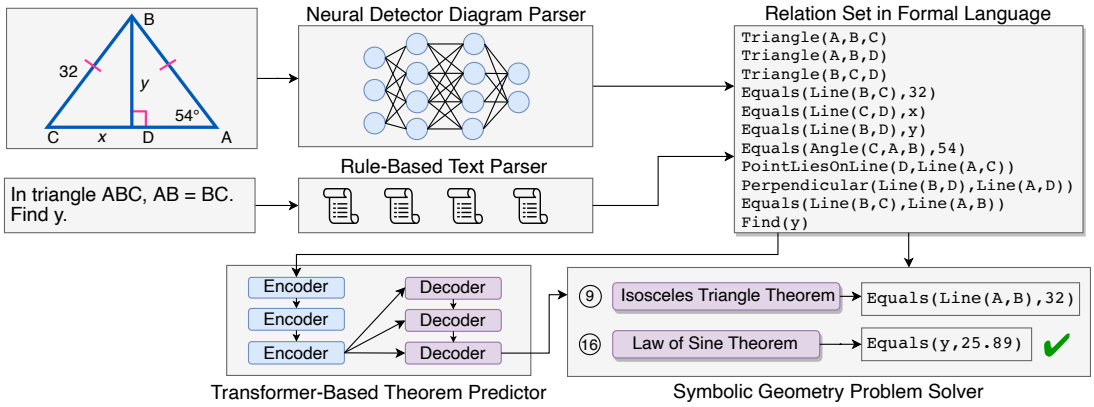


Figure 4: Given the problem diagram and text, our proposed Inter-GPS first parses the inputs into a relation set defined in formal language. Then Inter-GPS applies the theorem sequence predicted by the theorem predictor to perform symbolic reasoning over the relation set to infer the answer. ⑨ and ⑯ denote the theorem orders.

The relation set $\mathcal{R}$ defines geometry attributes and relations in the given problem, and is initialized with literals from the text and diagram parsers. $\mathcal{R}$ is further expanded with literals that are derived from definitions of geometry shapes. For example, a triangle is defined as three connected sides. So if there is a *literal* `Triangle(A,B,C)`, six more *literals* (`Ponit(A)`, `Ponit(B)`, `Ponit(C)`, `Line(A,B)`, `Line(B,C)`, `Line(C,A)`) will be appended to $\mathcal{R}$.

The theorem set $\mathcal{KB}$ is represented as a set of theorems, where each theorem k_i is written as a conditional rule with a premise p and a conclusion q. For the search step t, if the premise p of k_i matches the current relation set $\mathcal{R}_{t-1}$, the relation set is updated according to the conclusion q:

$$\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}, k_i \in \mathcal{KB}. \quad (2)$$

After the application of several theorems, equations between the known values and the unknown problem goal g are established, and g could be solved after solving these equations:

$$g^* \leftarrow \textsc{SolveEquation}(\mathcal{R}_t, g). \quad (3)$$

6.2 Theorem Predictor (TP)

As the geometry problems in Geometry3K are collected from high school textbooks, it might need to apply multiple theorems before the problems are solved. Intuitively, one possible search strategy is to use brute force to enumerate candidates in the theorem set randomly. The random search strategy is inefficient and might lead to problems unsolvable as there might be applications of complicated theorems in the early stage. Therefore, an ideal geometry problem solver can solve the problems using reasonable theorem application sequences. Students with good academic performance can solve a problem with prior knowledge learning from a certain amount of problem solving training. Inspired by this phenomenon, a theorem predictor is proposed to infer the possible theorem application sequence for inference after multiple attempts on the train data. Recent studies (Loos et al., 2017; Balunovic et al., 2018) also suggest that neural guided search can speed up the search process.

There are no annotated theorem application sequences for data in Geometry3K due to tremendous worker labor. Thus, we randomly sample from the theorem set multiple times to generate the application sequences. A generated sequence is regarded as positive if the geometry solver Inter-GPS solves the problem after the application of that sequence. A positive sequence with the minimum length for a problem is seen as pseudo-optimal. Finally, after attempts, we collect 1,501 training samples with the problem and its pseudo-optimal theorem application sequence.

Given the problem formal description $L = \{l_1, ..., l_m\}$, the theorem predictor aims to reconstruct the pseudo-optimal theorem sequence $T = \{t_1, ..., t_n\}$ token by token. We formulate the generation task as a sequence-to-sequence (Seq2Seq) problem and use a transformer-based model (Lewis et al., 2020) to generate theorem sequence tokens. Specifically, the transformer decoder predicts the next theorem order t_i given $T = \{t_1, ..., t_i\}$. The Seq2Seq model is trained to optimize the negative log-likelihood loss:

$$\mathcal{L}_{\text{TP}} = -\sum_{i=1}^{n} \log p_{\text{TP}}\left(t_i \mid t_1, \ldots, t_{i-1}\right), \quad (4)$$

Algorithm 1 Symbolic Geometry Solver

Input Literals $\mathcal{L}$, goal g, knowledge bases $\mathcal{KB}_1, \mathcal{KB}_2$
Output Numeric goal value g^* and theorem application $\mathcal{S}$

function SEARCH($\mathcal{L}$, g, $\mathcal{KB}_1$, $\mathcal{KB}_2$)
  Initialize relation set $\mathcal{R}_0$ with $\mathcal{L}$, $g^* = \emptyset$, $\mathcal{S} = \emptyset$
  $\mathcal{KB}_p \leftarrow$ THEOPREDICTOR($\mathcal{L}$) ▷ Predicted
  for $k_i \in \mathcal{KB}_p$ **do**
    $\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}$
    $\mathcal{S}$.APPEND(k_i)
  end for
  $g^* \leftarrow$ SOLVEEQUATION($\mathcal{R}_t, g$)
  if $g^* \neq \emptyset$ **then**
    return g^* and S
  end if
  while $g^* = \emptyset$ and $\mathcal{R}_t$ is updated **do**
    for $k_i \in \mathcal{KB}_1$ **do** ▷ Lower-order
      $\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}$
      $\mathcal{S}$.APPEND(k_i)
      $g^* \leftarrow$ SOLVEEQUATION($\mathcal{R}_t, g$)
      if $g^* \neq \emptyset$ **then**
        return g^* and S
      end if
    end for
    for $k_i \in \mathcal{KB}_2$ **do** ▷ Higher-order
      $\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}$
      $\mathcal{S}$.APPEND(k_i)
      $g^* \leftarrow$ SOLVEEQUATION($\mathcal{R}_t, g$)
      if $g^* \neq \emptyset$ **then**
        return g^* and S
      end if
    end for
  end while
end function

where p_{TP} is the parametrized conditional distribution in the theorem predictor model.

6.3 Low-first Search Strategy

After the application of the theorem sequence predicted by the theorem predictor, it is likely that Inter-GPS still could not find the problem goal. Generally, humans incline to use simple theorems first when solving math problems to reduce complex calculations. If simple theorems are not tangible, they will turn to more complex theorems. On account of that, we apply an efficient search strategy with heuristics driven by subject knowledge. We categorize theorems into two groups: **lower-order** theorem set $\mathcal{KB}_1$ and **higher-order** theorem set $\mathcal{KB}_2$. The lower-order set $\mathcal{KB}_1$ (e.g, *Triangle Angle-Sum Theorem*, *Congruent Triangle Theorem*) only involves in two simple operations of addition and subtraction, while $\mathcal{KB}_2$ (e.g, *Law of Sines*) requires complex calculations. In each following search step after using predicted theorems, we first enumerate theorems in the lower-order set $\mathcal{KB}_1$ to update the relation set $\mathcal{R}$:

$$\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}, k_i \in \mathcal{KB}_1. \quad (5)$$

If lower-order theorems fail to update $\mathcal{R}$ anymore, higher-order theorems are considered to update $\mathcal{R}$:

$$\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}, k_i \in \mathcal{KB}_2. \quad (6)$$

The search process stops once we find the problem goal g or the search steps reach the maximum steps allowed. The whole search algorithm for Inter-GPS is presented in Algorithm 1.

7 Experiments

7.1 Experimental Settings

Datasets and evaluation metrics. We conduct experiments on the Geometry3K and GEOS (Seo et al., 2015) datasets. The Geometry3K dataset involves 2,101 training data, 300 validation data, and 601 test data, respectively. The GEOS dataset provides 55 official SAT problems for evaluating geometry solvers. Regarding our proposed Inter-GPS model, if the one closest to the found solution among the four choices is exactly the ground truth, the found solution is considered correct. For a fair comparison, if Inter-GPS fails to output the numeric value of the problem goal within allowed steps, it will randomly choose the one from the four candidates. In terms of compared neural network baselines, the predicted answer has a maximum confidence score among choice candidates.

Baselines. We implement several deep neural network baselines for geometry solvers to compare them with our method. By default, these baselines formalize the geometry problem solving task as a classification problem, fed by the text embedding from a sequence encoder and the diagram representation from a visual encoder. *Q-only* only encodes the problem text in the natural language by a bi-directional Gated Recurrent Unit (Bi-GRU) encoder (Cho et al., 2014). *I-only* only encodes the problem diagram by a ResNet-50 encoder (He et al., 2016) as the input. *Q+I* uses Bi-GRU and ResNet-50 to encode the text and diagram, respectively. *RelNet* (Bansal et al., 2017) is implemented for embedding the problem text because it is a strong method for modeling entities and relations. *FiLM* (Perez et al., 2018) is compared as it achieves effective visual reasoning for answering questions about abstract images. *FiLM-BERT* uses the BERT encoder (Devlin et al., 2018) instead of the GRU encoder, and *FiLM-BART* uses the recently proposed BART encoder (Lewis et al., 2020).

Method	All	Angle	Length	Area	Ratio	Line	Triangle	Quad	Circle	Other
Random	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0
Human	56.9	53.7	59.3	57.7	42.9	46.7	53.8	68.7	61.7	58.3
Human Expert	90.9	89.9	92.0	93.9	66.7	95.9	92.2	90.5	89.9	92.3
Q-only	25.3	29.5	21.5	28.3	33.3	21.0	26.0	25.9	25.2	22.2
I-only	27.0	26.2	28.4	24.5	16.7	24.7	26.7	30.1	30.1	25.9
Q+I	26.7	26.2	26.7	28.3	25.0	21.0	28.1	32.2	21.0	25.9
RelNet (Bansal et al., 2017)	29.6	26.2	34.0	20.8	41.7	29.6	33.7	25.2	28.0	25.9
FiLM (Perez et al., 2018)	31.7	28.7	32.7	**39.6**	33.3	33.3	29.2	33.6	30.8	29.6
FiLM-BERT (Devlin et al., 2018)	32.8	32.9	33.3	30.2	25.0	32.1	32.3	32.2	34.3	33.3
FiLM-BART (Lewis et al., 2020)	33.0	32.1	33.0	35.8	50.0	34.6	32.6	37.1	30.1	37.0
Inter-GPS (ours)	**57.5**	**59.1**	**61.7**	30.2	**50.0**	**59.3**	**66.0**	**52.4**	**45.5**	**48.1**
Inter-GPS (GT)	78.3	83.1	77.9	62.3	75.0	86.4	83.3	77.6	61.5	70.4

Table 5: Evaluation results by our proposed method and compared baselines on the Geometry3K dataset.

Method	Acc (%)
GEOS (Seo et al., 2015)	49
GEOS++ (Seo et al., 2015)	49
GEOS-OS (Sachan and Xing, 2017)	52
GEOS++AXIO (Sachan et al., 2017)	55
Inter-GPS (ours)	**67**

Table 6: Evaluation results on the GEOS dataset.

Implementation details. Main hyper-parameters used in the experiments are shown below. For our symbolic solver, a set of 17 geometry theorems is collected to form the knowledge base. For generating positive theorem sequences, each problem is attempted by 100 times with the maximum sequence length of 20. The transformer model used in the theorem predictor has 6 layers, 12 attention heads, and a hidden embedding size of 768. Search steps in Inter-GPS are set up to 100. For the neural solvers, we choose the Adam optimizer and set the learning rate as 0.01, and the maximum epochs are set as 30. Each experiment for Inter-GPS is repeated three times for more precise results.

7.2 Comparisons with Baselines

Table 5 compares the results of symbolic solver Inter-GPS with baselines on our proposed Geometry3K dataset. Apart from the overall accuracy, the results of different problem types are also reported. Benefiting from symbolic reasoning with theorem knowledge, our Inter-GPS obtains an overall accuracy of 57.5%, significantly superior to all neural baselines. Inter-GPS even attains a better accuracy compared to human beings. Inter-GPS with ground truth formal language gains a further improvement of 20.8%. Inter-GPS also obtains state-of-the-art performance over exiting geometry solvers on the GEOS dataset, as shown in Table 6.

7.3 Ablation Study and Discussion.

Search strategies. The overall accuracy and average steps needed for solving problems with different search strategies in Inter-GPS are reported in Table 7. *Predict* refers to the strategy that uses the theorems from the theorem predictor followed by a random theorem sequence. The strategy largely reduces the average steps to 6.5. The final strategy in Inter-GPS applies the predicted theorems first and lower-order theorems in the remain search steps, and gains the best overall accuracy.

Search strategies	Accuracy (%)	# Steps
Random	75.5 $\pm$ 0.2	13.2 $\pm$ 0.1
Low-first	77.3 $\pm$ 0.3	15.1 $\pm$ 0.2
Predict	77.5 $\pm$ 0.1	**6.5** $\pm$ 0.1
Predict+Low-first (final)	**78.3** $\pm$ 0.1	7.1 $\pm$ 0.1

Table 7: Performance of Inter-GPS with different search strategies.

Problem parsers and literal sources. The rule-based text parser achieves an accuracy of 97% while only 67% for the semantic text parser. Table 8 reports the Inter-GPS performance fed with different sources of literals. With literals generated from our problem solver, Inter-GPS achieves an accuracy of 57.5%. The current text parser performs very well as there is only a slight gap between Inter-GPS with generated text literals and ground truth literals. An improvement of 17.5% for Inter-GPS with annotated diagram literals indicates that there is still much space to improve for the diagram parser.

	Diagram w/o	Diagram	Diagram (GT)
Text w/o	25.0 $\pm$ 0.0	46.6 $\pm$ 0.7	58.7 $\pm$ 0.2
Text	25.4 $\pm$ 0.0	57.5 $\pm$ 0.2	75.0 $\pm$ 0.6
Text (GT)	25.4 $\pm$ 0.0	58.0 $\pm$ 1.7	**78.3** $\pm$ 0.1

Table 8: Performance of Inter-GPS with predicted and ground truth (GT) literals.

Searching step distribution. Figure 5 compares correctly solved problem distribution by the average number of search steps in different strategies. Our final Inter-GPS applies the *Predict+Low-first* strategy, with which 65.97% problems are solved in two steps and 70.06% solved in five steps.

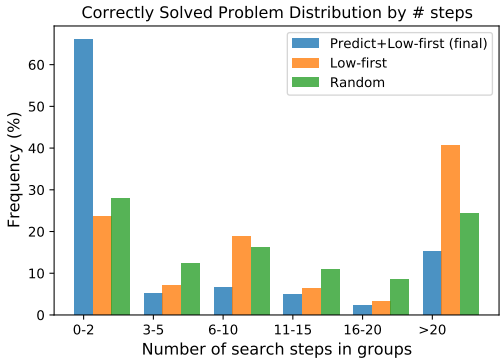


Figure 5: Correctly solved problem distribution by the number of search steps.

Neural geometry solvers. Current neural network baselines for geometry solving fail to achieve satisfactory results in the Geometry3K dataset. It is because there are limited data samples for these neural methods to learn meaningful semantics from the problem inputs. Besides, dense implicit representations might not be suitable for logical reasoning tasks like geometry problem solving. We replace the inputs of problem text and diagram in the *Q+I* baseline with the ground truth textual and visual formal annotations and report the result in Table 9. An improvement of 9.2% indicates the promising potential for neural network models for problem solving if structural representations with rich semantics are learned.

	Diagram (visual)	Diagram (formal)
Text (natural)	26.7	35.3
Text (formal)	34.6	35.9

Table 9: Neural solver performance with different representations of the problem text and diagrams.

Failure cases. Inter-GPS might not find a solution because of inaccurate parsing results and the incomplete theorem set. Figure 6 illustrates some failure examples for Inter-GPS. For example, diagram parsing tends to fail if there are ambiguous annotations or multiple primitives in the diagram. It is difficult for the text parser to handle nested expressions and uncertain references. And the symbolic solver is still not capable of solving complex problems with combined shapes and shaded areas

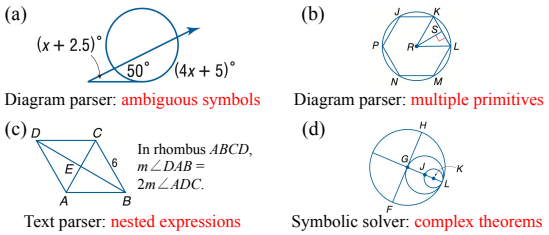


Figure 6: Failure examples for Inter-GPS.

in the diagrams.

Interpretability in Inter-GPS. Inter-GPS provides an interpretable symbolic solver for geometry problem solving. First, Inter-GPS parses the problem contents into a structural representation of formal language. Second, Inter-GPS performs symbolic reasoning to update the geometric relation set explicitly. Last, Inter-GPS applies reasonable theorems sequentially in the search process.

8 Conclusion

Solving geometry problems is one of the most challenging tasks in math question answering. In this paper, we propose a large-scale benchmark, Geometry3K, which consists of 3,002 high-school geometry problems with dense descriptions in formal language. We further propose a novel geometry solving approach, *Interpretable Geometry Problem Solver* (Inter-GPS), which parses the problem as formal language from an automatic parser and performs symbolic reasoning over the theorem knowledge base to infer the answer. Also, a theorem predictor with a low-first search strategy is designed to generate the reasonable theorem application sequence. Experiment results show that Inter-GPS outperforms existing state-of-the-art methods by a large margin. In the future, we plan to extend our work in other math question answering tasks and explore more general symbolic reasoning models.

Acknowledgments

This work was supported by MURI N00014-16-1-2007 and DARPA XAI N66001-17-2-4029. We thank Johnson Zhou and Jiahao Li for collecting part of the data. And we thank the help from Jianheng Tang in baseline implementation.

Ethical Impact

The problems in Geometry3K are collected from online open sources. The work in this paper may inspire the following research in symbolic reasoning and interpretable models and facilitate education.

References

Chris Alvin, Sumit Gulwani, Rupak Majumdar, and Supratik Mukhopadhyay. 2014. Synthesis of geometry proof problems. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Chris Alvin, Sumit Gulwani, Rupak Majumdar, and Supratik Mukhopadhyay. 2017. Synthesis of solutions for shaded area geometry problems. In *The Thirtieth International Flairs Conference*.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Mislav Balunovic, Pavol Bielik, and Martin T Vechev. 2018. Learning to solve smt formulas. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10338–10349.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 809–815.

Trapit Bansal, Arvind Neelakantan, and Andrew McCallum. 2017. Relnet: End-to-end modeling of entities & relations. *arXiv preprint arXiv:1706.07179*.

Mohan Chinnappan. 1998. Schemas and mental models in geometry problem solving. *Educational Studies in Mathematics*, 36(3):201–217.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. 1996. Automated generation of readable proofs with geometric invariants. *Journal of Automated Reasoning*, 17(3):325–347.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT (NAACL)*.

Wenbin Gan and Xinguo Yu. 2018. Automatic understanding and formalization of natural language geometry problems using syntax-semantics models. *International Journal of Innovative Computing, Information and Control*, pages 83–98.

Wenbin Gan, Xinguo Yu, Ting Zhang, and Mingshu Wang. 2019. Automatically proving plane geometry theorems stated by text and diagram. *International Journal of Pattern Recognition and Artificial Intelligence*, 33(07):1940003.

Feng Han and Song-Chun Zhu. 2005. Bottom-up/top-down image parsing by attribute graph grammar. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1778–1785. IEEE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Yining Hong, Qing Li, Daniel Ciao, Siyuan Huang, and Song-Chun Zhu. 2021. Learning by fixing: Solving math word problems with weak supervision. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI-21*.

Mark Hopkins, Ronan Le Bras, Cristian Petrescu-Prahova, Gabriel Stanovsky, Hannaneh Hajishirzi, and Rik Koncel-Kedziorski. 2019. Semeval-2019 task 10: Math question answering. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 893–899.

Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 805–814.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 595–603.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7871–7880.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988.

Sarah Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. 2017. Deep network guided proof search. *arXiv preprint arXiv:1701.06972*.

Takuya Matsuzaki, Takumi Ito, Hidenao Iwane, Hirokazu Anai, and Noriko H Arai. 2017. Semantic parsing of pre-university math problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2131–2141.

Andi Saparuddin Nur and Evy Nurvitasari. 2017. Geometry skill analysis in problem solving reviewed from the difference of cognitive style students junior

high school. *Journal of Educational Science and Technology (EST)*, 3(3):204–210.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 32.

Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. 2020. Semantically-aligned universal tree-structured solver for math word problems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3780–3789.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. *Transactions of the Association for Computational Linguistics (TACL)*, 6:159–172.

Mrinmaya Sachan, Avinava Dubey, Eduard H Hovy, Tom M Mitchell, Dan Roth, and Eric P Xing. 2020. Discourse in multimedia: A case study in extracting geometry knowledge from textbooks. *Computational Linguistics*, 45(4):627–665.

Mrinmaya Sachan, Kumar Dubey, and Eric Xing. 2017. From textbooks to knowledge: A case study in harvesting axiomatic knowledge from textbooks to solve geometry problems. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 773–784.

Mrinmaya Sachan and Eric Xing. 2017. Learning to solve geometry problems from natural language demonstrations in textbooks. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics*, pages 251–261.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram understanding in geometry questions. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 1466–1476.

Linda G Shapiro and George C Stockman. 2001. *Computer vision*. Prentice Hall.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

Kewei Tu, Meng Meng, Mun Wai Lee, Tae Eun Choe, and Song-Chun Zhu. 2014. Joint video and text parsing for understanding events and answering queries. *IEEE MultiMedia*, 21(2):42–70.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30:5998–6008.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018. Translating a math word problem to an expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 7144–7151.

Wu Wen-Tsun. 1986. Basic principles of mechanical theorem proving in elementary geometries. *Journal of Automated Reasoning*, 2(3):221–252.

Xinguo Yu, Mingshu Wang, Wenbin Gan, Bin He, and Nan Ye. 2019. A framework for solving explicit arithmetic word problems and proving plane geometry theorems. *International Journal of Pattern Recognition and Artificial Intelligence*, 33(07):1940005.

Song-Chun Zhu and David Mumford. 2006. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362.

A Appendix

We define 91 *predicates* and their corresponding *literal* templates in the geometry language domain. For development, these *predicates* are categorized into six groups: geometric shapes (Table 10), unary geometric attributes (Table 11), general geometric attributes (Table 12), binary geometric relations (Table 13), A-IsXOf-B-type geometric relations (Table 14), as well as numerical attributes and relations (Table 15). Moreover, `$` in the literal templates denotes the undetermined shape.

#	Predicates	Literal templates
1	Point	Point(A),Point($)
2	Line	Line(A,B),Line(m),Line($)
3	Angle	Angle(A,B,C),Angle(A),Angle(1),Angle($)
4	Triangle	Triangle(A,B,C),Triangle($),Triangle($1,$2,$3)
5	Quadrilateral	Quadrilateral(A,B,C,D),Quadrilateral(1),Quadrilateral($)
6	Parallelogram	Parallelogram(A,B,C,D),Parallelogram(1),Parallelogram($)
7	Square	Square(A,B,C,D),Square(1),Square($)
8	Rectangle	Rectangle(A,B,C,D),Rectangle(1),Rectangle($)
9	Rhombus	Rhombus(A,B,C,D),Rhombus(1),Rhombus($)
10	Trapezoid	Trapezoid(A,B,C,D),Trapezoid(1),Trapezoid($)
11	Kite	Kite(A,B,C,D),Kite(1),Kite($)
12	Polygon	Polygon($)
13	Pentagon	Pentagon(A,B,C,D,E),Pentagon($)
14	Hexagon	Hexagon(A,B,C,D,E,F),Hexagon($)
15	Heptagon	Heptagon(A,B,C,D,E,F,G),Heptagon($)
16	Octagon	Octagon(A,B,C,D,E,F,G,H),Octagon($)
17	Circle	Circle(A),Circle(1),Circle($)
18	Arc	Arc(A,B),Arc(A,B,C),Arc($)
19	Sector	Sector(O,A,B),Sector($)
20	Shape	Shape($)

Table 10: 20 predicates and corresponding literal templates for geometric shapes.

#	Predicates	Literal templates
1	RightAngle	RightAngle(Angle($))
2	Right	Right(Triangle($))
3	Isosceles	Isosceles(Polygon($))
4	Equilateral	Equilateral(Polygon($))
5	Regular	Regular(Polygon($))
6	Red	Red(Shape($))
7	Blue	Blue(Shape($))
8	Green	Green(Shape($))
9	Shaded	Shaded(Shape($))

Table 11: 9 predicates and corresponding literal templates for unary geometric attributes.

#	Predicates	Literal templates
1	AreaOf	AreaOf(A)
2	PerimeterOf	PerimeterOf(A)
3	RadiusOf	RadiusOf(A)
4	DiameterOf	DiameterOf(A)
5	CircumferenceOf	CircumferenceOf(A)
6	AltitudeOf	AltitudeOf(A)
7	HypotenuseOf	HypotenuseOf(A)
8	SideOf	SideOf(A)
9	WidthOf	WidthOf(A)
10	HeightOf	HeightOf(A)
11	LegOf	LegOf(A)
12	BaseOf	BaseOf(A)
13	MedianOf	MedianOf(A)
14	IntersectionOf	IntersectionOf(A,B)
15	MeasureOf	MeasureOf(A)
16	LengthOf	LengthOf(A)
17	ScaleFactorOf	ScaleFactorOf(A,B)

Table 12: 17 predicates and corresponding literal templates for general geometric attributes .

#	Predicates	Literal templates
1	PointLiesOnLine	PointLiesOnLine(Point($),Line($1,$2))
2	PointLiesOnCircle	PointLiesOnCircle(Point($),Circle($))
3	Parallel	Parallel(Line($),Line($))
4	Perpendicular	Perpendicular(Line($),Line($))
5	IntersectAt	IntersectAt(Line($),Line($),Line($),Point($))
6	BisectsAngle	BisectsAngle(Line($),Angle($))
7	Congruent	Congruent(Polygon($),Polygon($))
8	Similar	Similar(Polygon($),Polygon($))
9	Tangent	Tangent(Line($),Circle($))
10	Secant	Secant(Line($),Circle($))
11	CircumscribedTo	CircumscribedTo(Shape($),Shape($))
12	InscribedIn	InscribedIn(Shape($),Shape($))

Table 13: 12 predicates and corresponding literal templates for binary geometric relations.

#	Predicates	Literal templates
1	IsMidpointOf	IsMidpointOf(Point($),Line($))
2	IsCentroidOf	IsCentroidOf(Point($),Shape($))
3	IsIncenterOf	IsIncenterOf(Point($),Shape($))
4	IsRadiusOf	IsRadiusOf(Line($),Circle($))
5	IsDiameterOf	IsDiameterOf(Line($),Circle($))
6	IsMidsegmentOf	IsMidsegmentOf(Line($),Triangle($))
7	IsChordOf	IsChordOf(Line($),Circle($))
8	IsSideOf	IsSideOf(Line($),Polygon($))
9	IsHypotenuseOf	IsHypotenuseOf(Line($),Triangle($))
10	IsPerpendicularBisectorOf	IsPerpendicularBisectorOf(Line($),Triangle($))
11	IsAltitudeOf	IsAltitudeOf(Line($),Triangle($))
12	IsMedianOf	IsMedianOf(Line($),Quadrilateral($))
13	IsBaseOf	IsBaseOf(Line($),Quadrilateral($))
14	IsDiagonalOf	IsDiagonalOf(Line($),Quadrilateral($))
15	IsLegOf	IsLegOf(Line($),Trapezoid($))

Table 14: 15 predicates and corresponding literal templates for A-IsXOf-B-type geometric relations.

#	Predicates	Literal templates
1	SinOf	SinOf(Var)
2	CosOf	CosOf(Var)
3	TanOf	TanOf(Var)
4	CotOf	CotOf(Var)
5	HalfOf	HalfOf(Var)
6	SquareOf	SquareOf(Var)
7	SqrtOf	SqrtOf(Var)
8	RatioOf	RatioOf(Var),RatioOf(Var1,Var2)
9	SumOf	SumOf(Var1,Var2,...)
10	AverageOf	AverageOf(Var1,Var2,...)
11	Add	Add(Var1,Var2,...)
12	Mul	Mul(Var1,Var2,...)
13	Sub	Sub(Var1,Var2,...)
14	Div	Div(Var1,Var2,...)
15	Pow	Pow(Var1,Var2)
16	Equals	Equals(Var1,Var2)
17	Find	Find(Var)
18	UseTheorem	UseTheorem(A_B_C)

Table 15: 18 predicates and corresponding literal templates for numerical attributes and relations.