Automated Generation of Geometry Questions for High School Mathematics

Rahul Singhal[1], Martin Henz[1] and Kevin McGee[2]
[1]*School of Computing, National University of Singapore, Singapore, Singapore*
[2]*Department of Communications and New Media, National University of Singapore, Singapore, Singapore*
{*rahulsinghal, mckevin*}*@nus.edu.sg, henz@comp.nus.edu.sg*

Keywords: Automated Deduction, Graph-based Knowledge Representation.

Abstract: We describe a framework that combines a combinatorial approach, pattern matching and automated deduction to generate and solve geometry problems for high school mathematics. Such a system would help teachers to quickly generate large numbers of questions on a geometry topic. Students can explore and revise specific topics covered in classes and textbooks based on generated questions. The system can act as a personalized instructor - it can generate problems that meet users specific weaknesses. This system may also help standardize tests such as GMAT and SAT. Our novel methodology uses (i) a combinatorial approach for generating geometric figures (ii) a pattern matching approach for generating questions and (iii) automated deduction to generate new questions and solutions. By combining these methods, we are able to generate questions involving finding or proving relationships between geometric objects based on a specification of the geometry objects, concepts and theorems to be covered by the questions. Experimental results show that a large number of questions can be generated in a short time. We have tested our generated questions on an existing geometry question solving software JGEX, verifying the validity of the generated questions.

1 INTRODUCTION

Geometry, the study of space and spatial relationships, is an important and essential branch of the mathematics curriculum at all grade levels. The study of geometry develops logical reasoning and deductive thinking, which helps us expand both mentally and mathematically. Children who develop a strong sense of spatial relationships and who master the concepts and language of geometry are better prepared to learn number and measurement ideas, as well as other advanced mathematical topics (National Council of Teachers of Mathematics, 1989).

Euclidean geometry is a branch of mathematics which deals with the study of plane and solid figures on the basis of axioms and theorems employed by the Greek mathematician Euclid. It is important to understand Euclidean geometry when studying a course because geometry does not follow any set pattern. In Euclidean geometry, one can only learn the axioms and results proven from these axioms. The student must apply these axioms with no set pattern or list of steps for solving such questions. Therefore, a question may have (possibly infinitely) many solutions. To practice the required problem solving skills, students require a large number of different types of geometry questions on various concepts. Generally, textbooks and online sites provide a limited predefined number of questions for each topic. Once practiced, these questions lose their purpose of enhancing student thinking. The tedious and error-prone task of generating high-quality questions challenges the resources of teachers. Hence, there is a need for software which assists both teachers and students to generate geometry questions and solutions.

The software can also act as a personalized instructor. It can generate questions that cover the required topic and meet the required level of student proficiency. Apart from helping users, the framework of generating questions has scientific contributions to other research areas, such as Intelligent tutor systems (ITS) and Massive Online Open Courses (MOOC).

Various research has been performed in automated deduction of theorems at high school level in the geometry domain, although none with the goal of automatic question generation. Instead, they mainly demand users to generate the question with the help of tools. In addition, they mainly focus on solving and assessment of the questions. Our survey shows that the currently available geometry systems, such as

JGEX, Geogebra, Cinderella and Sketchpad, are not able to automatically generate questions of user specified geometry topics.

The aim of this paper is to develop a framework that can be used to generate geometry questions based on specific inputs, such as geometry objects and theorems to be involved in their solution. For a given set of geometry objects, the algorithm can generate a large set of questions along with their solutions. The solutions will involve user desired theorems directly or indirectly. Hence the framework can generate questions to test the theorem on various geometry objects and concepts.

Our framework can generate questions involving algebraic computations for a solution. Currently, we restrict the relationships between quantitative entities to linear ones. Our framework has a predefined database of theorems and concepts which can be used for generating questions. Given a set of user-selected objects, our system can generate all possible questions using an existing database of concepts and theorems.

The main contributions of this paper are as follows:

1. Our geometry question generator combines the complementary strengths of a combinatorial approach, pattern matching and deductive reasoning. It can generate geometry questions which were not possible previously.
2. A substantial evaluation is provided that demonstrates the effectiveness of our generator. It can generate various categories of the questions covered in the textbooks and questions asked in SAT and GMAT.
3. A knowledge representation is described for geometry objects and predefined theorems. This representation helps in applying theorem information within the generated questions.

2 RELATED WORK

In this section, we provide a general review of related works. Computational research in the geometry domain started in the 19th century. However, lack of question generating in geometry in the literature required a principled ab-initio approach in our work.

Researchers mainly focused on proving geometry theorems. Broadly, geometric theorems are proven using algebraic or non-algebraic methods. Algebraic methods such as Wu's method and Gröbner bases (GB) (Chou et al., 1994) generate algebraic equations from the given facts and relations. They involve coordinate geometry and algebraic formulas to find new theorems. Hence, the proofs generated by these methods are out of scope of high school mathematics (Chou and Gao, 2001). Therefore, we are mainly interested in the non-algebraic methods. Non-algebraic methods for automatic discovery and proof of geometry theorems can be further divided into three approaches: coordinate-free methods, formal logic methods and search methods.

2.1 Coordinate-free Methods

These methods are applicable to constructive geometry statements of equality. Various methods have been proposed under this category, such as the area method (Narboux, 2010; Chou et al., 2011), the full-angle method (Chou et al., 2000; Wilson and Fleuriot, 2005), the complex number method, the vector method for Euclid plane geometry (Chou et al., 2000), the volume method for Euclidean solid geometry (Chou and Gao, 2001) and the argument method for non-Euclidean geometry (Chou and Gao, 2001). The area method was further improved and developed into a computerized algorithm (Chou et al., 2000; Chou and Gao, 2001). These methods can only be applied in constructive geometry, which is outside the scope of this paper.

2.2 Formal Logic Methods

Theorems in Tarski's geometry were proven using Interactive Theorem Prover (ITP) (McCharen et al., 1976), albeit limited to several trivial theorems. ITP is an interactive theorem prover based on the resolution principle, which generates resolution style proofs that resemble traditional proofs. In 1989, Quaife continued the work of McCharen with Otter (Quaife, 1989). Otter is an automated theorem prover based on resolution. A series of tactics such as Hyper-resolution, UR-resolution, paramodulation, support set and clause weight can improve the resolution efficiency. In 2003, Meikle and Fleuriot developed Hilbert's geometry with the theorem prover Isabelle/Isar (Meikle and Fleuriot, 2003), an interactive and/or semi-automated theorem prover. The greatest disadvantage of formal logic methods is their low reasoning efficiency (Jiang and Zhang, 2012), caused by combinatorial explosion of their search space.

2.3 Search Methods

Fundamentally, the search method is used in a rule-based expert system, which includes a rule database,

a fact database and a reasoning engine. The inference rules stored in the rule database include axioms, theorems, lemmas, formula, definitions and algebraic operation rules in geometry. Geometric facts stored in the fact database include geometry predicates such as angle bisector, equidistant points, parallel lines and perpendicular lines. The reasoning engine deduces new geometric facts by applying inference rules to the facts database. There are three ways of performing deduction search, namely forward chaining (Zhang et al., 1996; Wilson and Fleuriot, 2005), backward chaining (Wilson and Fleuriot, 2005), and bidirectional chaining (Coelho and Pereira, 1986).

Forward chaining starts from the hypotheses and rules and works towards the conclusion, while backward chaining starts from the conclusion and works towards the hypotheses through rules, and the bidirectional chaining proceeds in both direction simultaneously. In 1975, Nevins proved geometry theorems via bidirectional chaining (Nevins, 1975), and in 1986, Coelho and Pereira developed a prover GEOM based on bidirectional chaining (Coelho and Pereira, 1986). Unlike Nevins, GEOM implemented backward chaining in the reasoning process and only used forward chaining to search congruent triangles hidden in the diagram. In 1995, Chou, Gao, and Zhang described an integration of a deductive database into the search methods (Chou et al., 2000; Gao et al., 1998; Gao et al., 1998; Chou and Gao, 2001). The deductive database method can find a fixpoint for a given geometry diagram, containing all properties of the geometry diagrams that can be deduced using a fixed set of geometry rules. They effectively controlled the size of the facts database with structural deductive database techniques.

Each search method has different advantages and disadvantages. Forward chaining is always feasible, but does not have an explicit reasoning goal. Backward chaining has an explicit reasoning goal, but sometimes lacks feasibility. The bidirectional chaining method is feasible and has an explicit reasoning goal, but is hard to implement. We are using forward chaining in our framework as it is most suited for generating previously unknown quantities.

3 GEOMETRY QUESTION SPECIFICATION

Mathematically a geometry question Q generated by our system can be represented by a quintuple (Object O, Concept C, Theorem T, Relationship R, Query type qt) where:

- O ∈ (lines, triangles, square, circle, ...)

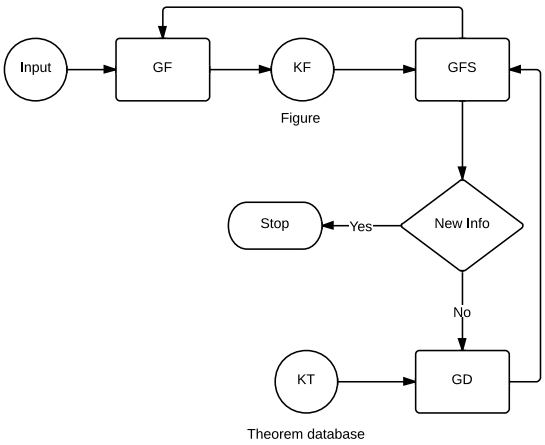


Figure 1: Connection of these components and knowledge representations.

- C ∈ (perpendicular, parallel, midpoint, angle-bisector, circumcircle...)
- T ∈ (Pythagorean theorem, similarity theorem, various triangle-theorems, ...)
- R ∈ (syntactic, quantitative)
- qt ∈ (syntactic, quantitative)

In order to generate geometry questions, the user has to provide a set of geometry objects O such as triangles, squares, etc., and a set of concepts C which the user wants the generated question to cover. Optionally the user may select a set of theorems T to be tested by the question. The relationship R can be either syntactic such as perpendicular, parallel, etc., or quantitative such as the length of an object, the ratio of two quantities etc. The query type qt is the type of generated question that can be asked to find the hidden relationship which can be calculated from the given information.

4 FRAMEWORK

Our framework comprises three major components along with the knowledge databases used for storing input, geometry figures and a set of predefined theorems. Figure 1 shows the connection of these components. The input consists of geometric objects, concepts and theorems selected by the user. The input is fed into the first component, *Generating Figure (GF)*. This component is used for generation of geometric figures from the input. Each figure constitutes a diagrammatic schema (DS) (Greeno et al., 1979) and a set of unknown variables representing the relationship between geometric objects. The geometric figure is passed to the second component, *Generating*

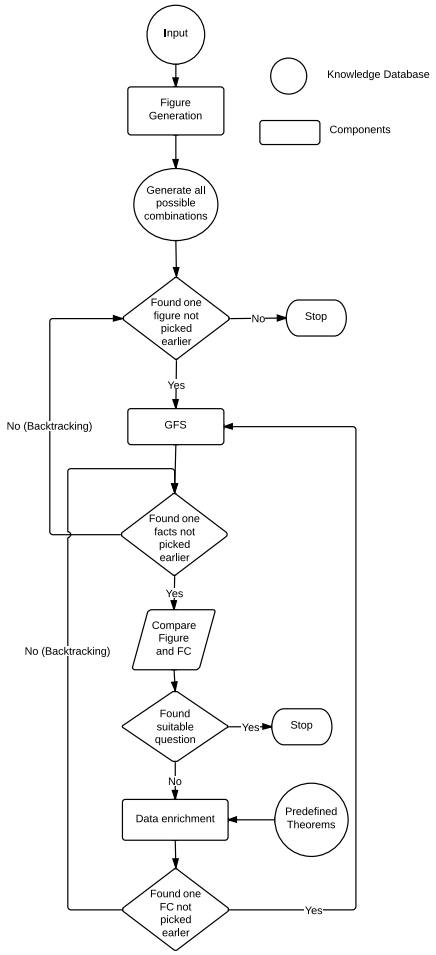


Figure 2: Flow diagram of the algorithm for generating questions.

Facts and Solutions (GFS). This component is used to find values of the unknown variables representing possible relationships to be asked by the generated question. GFS makes use of the predefined knowledge database of axioms. It results in the formation of a *configuration* (Cfg) containing known values for some relationships between its objects. New information refers to the generation of suitable questions. A question is considered suitable if it covers the essential information such as a new fact and a proper reasoning for the generated fact. Currently, the decision of suitability is taken manually by the user.If the suitability conditions for the generated configuration (Cfg) are not met then configuration is fed into the last component, *Generating data for the figure (GD)*. GD assigns values to unknown variables of relationships. Repeated processing by GD makes the questions generated from Cfg easier and easier, because the values assigned by GD appear as given facts in the generated questions. GD makes use of a predefined set of theorems and makes sure that the assignment results in successful generation of geometry questions. FC generated from this component is again passed to GFS component and this loop continues until a question is found which meets suitability condition.

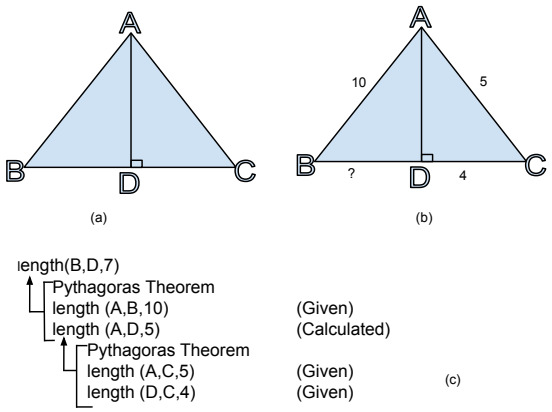


Figure 3: (a) The figure is generated by using *triangle* and *perpendicular* as the geometry objects using the GF component. (b) The data is generated for (a) using the GD component. (c) The new fact and its derivation using the GFS component is generated from the figure shown in (b).

Along with these components, two knowledge databases are used in the framework, namely the Knowledge database for the generated figures *(KF)* and the knowledge database for predefined theorems *(KT)*. KF is generated dynamically during generation of questions while KT is fixed and can be modified only after the algorithm terminates. KF is generated and modified by the GF function, and GD requires both KF and KT for generating data for the figure. GFS has a set of axioms but no access to any knowledge databases.

4.1 Algorithm

Figure 2 represents the flowchart describing the algorithm for generating geometric questions. Algorithm 1 describes the flowchart in further detail. Here, we shall explain Algorithm 1 with the help of an example. Figure 3 shows the step by step execution of the algorithm. We select the following input in our example.

- Object: triangle and line segment
- Concept: perpendicular
- Theorem: Pythagorean Theorem
- Number of questions: 1

In the next subsections, we will describe each knowledge database in detail, followed by the three components and their interaction with these databases.

Data: User selects object(s), concept(s), theorem(s) and the number of questions to be generated.
Result: Question with single or multiple solutions.

1. Generate all possible figures consisting of geometry objects using GF function from the given input.
2. Find one figure which has not been picked earlier. If found, go to next step else terminate.
3. Save this figure using KF knowledge database.
4. Assign values to variables of figure obtained in second step from the predefined knowledge database of axioms through GFS function. Configurations (Cfg) are generated from this step.
5. Find one Cfg not picked earlier. If found, goto next step. If not found, backtracking to step 2.
6. Compare Cfg with the previously stored figure.
7. If the comparison gives the desired suitability then the Cfg is declared as a generated question and the algorithm stops. If the conditions are not met or more number of questions are required, go to the next step.
8. Configuration Cfg obtained from GFS is fed into the third component to assign more unknown variables. It makes use of KT, a predefined database of theorems.
9. From the configurations obtained from GD, find a new configuration Cfg which has not been chosen earlier. If found, goto step 3. If all configurations have been chosen earlier, goto step 5.

Algorithm 1: Algorithm for generating geometry questions.

4.2 Knowledge Database for the Figure/ Figure Configuration (KF)

KF contains the question figure and configurations using a graph-based knowledge representation. The

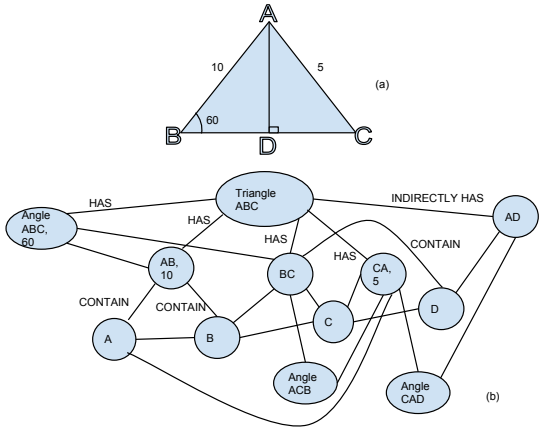


Figure 4: (a)The geometry figure and (b) Partially drawn knowledge representation of (a).

nodes of our graph represent the geometric objects. Two nodes can have multiple labeled arcs between them, representing multiple relationships. In addition, an arc can be bidirectional or unidirectional depending on the relationship between the nodes. Following the example of the previous section, Figure 4(a) shows the geometric figure: a triangle ABC is given and AD is perpendicular to BC. Figure 4b shows its partial representation in a graph format where we focus on the most important relations. Relationships such as "equal angle" and "equal length", which require multiple objects/nodes, are represented by nodes, e.g. , the relationship *"∠ABC"* in Figure 4(b). In addition, nodes of our graph store the value of quantitative relationships. For example, the node named *" Angle ABC "* stores the value of angle and is connected to the two nodes representing sides AB and BC.

KF is generated dynamically from the geometry objects selected by the user. KF is initialized by the GF component and later modification is done according to Algorithm 1 defined in Section 4.1. Modification involves creation and deletion of nodes and arcs between them.

4.3 Knowledge Database for Predefined Theorems (KT)

KT is a knowledge database which contains the predefined database of theorems. KT is used for assigning values to the unknown variables of configurations generated by the GF component. KT uses a graph-based representation similar to KF. However the difference lies in the timing of generation. Unlike KF, KT is generated before the system can generate questions and remains unchanged during the algorithm execution. In order to use KT in the algorithm, some

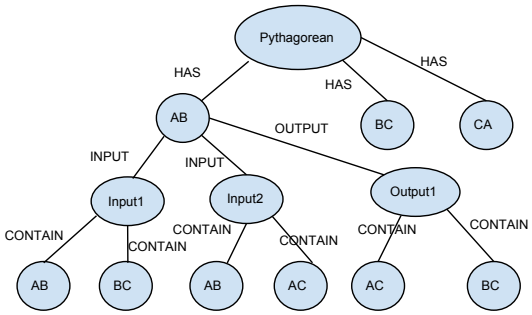


Figure 5: Representation used for generating data for the given figure.

properties of theorems need to be known. Each theorem can be applied to a particular geometry configuration. Each theorem has certain inputs and outputs corresponding to inputs in the geometry configuration. Hence usage of a theorem requires assigning the variables of input and/or output. In KT, the information of input and output for each theorem is stored along with the geometric configuration in which the theorem can be applied. Nodes represent objects whereas arcs represent the input and output relationships. Input and output are decided offline before the execution of the algorithm and later used for assigning unknown variables to get a useful question. To illustrate the process, the Pythagorean Theorem is taken as an example in Figure 5. Given a triangle ABC, where $\angle ABC$ is 90°, Figure 5 partially shows its usage of KT in generating data. The node representing the side AB has three arcs, two of which represent input and one of which represents an output relationship. To use the side AB as an input, the length of the sides AB and BC or AC needs to be assigned. On the other hand, to use side AB as an output, the length of sides AC and BC needs to be given. By giving this as input or output, we can be sure that the Pythagorean Theorem will give a consistent result.

Now we are ready to describe the three components which will use these knowledge databases to generate questions and answers.

4.4 Generating Figure from the User Input (GF)

This is the first step executed by Algorithm 1 described in Section 4.1. This component generates a figure through the combination of a predefined number of ways to combine geometric objects. Currently, we are focusing on triangles and line segments. Hence our algorithm includes combinations in which various triangles and lines can intersect. Furthermore, we are currently limiting our algorithm to the intersection of two triangles. Adding more objects may

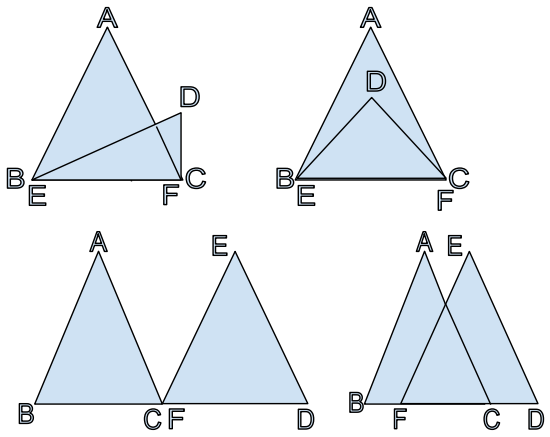


Figure 6: Some predefined ways two triangle can intersect.

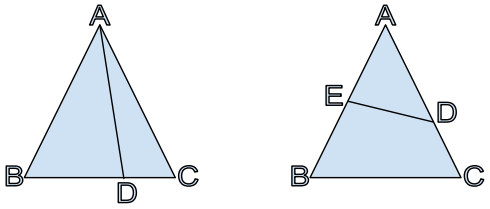


Figure 7: Some predefined ways a triangle and a line can intersect.

make the figure more complex, possibly leading to unusual questions that are not commonly found in textbooks. However there is no limitation on the intersection of lines and triangles. Figure 6 shows some of the possible ways of intersection of two triangles, which have been taken into consideration while generating questions. Intersection of two triangles includes various cases such as two triangles sharing common sides, common part of a side, common vertex. Figure 7 shows the two ways in which a line can intersect with a triangle. Figure 8 describes the flowchart for figure generation from the user-desired input. It includes three functions for adding various items such as objects, concepts and theorems. These functions will add their respective items into the generated DS. For instance, addObject will add a new object without removing the old one. Figure 9 shows an example of adding a new object. Figure 10 shows the same functions but for concept involvement in the DS. Similar functions are used for adding theorems, not discussed here.

The algorithm starts looking for DS in our knowledge database, comprising of the user-desired input. If DS is not found, then a DS is generated using the addObject function, possibly resulting in several DS. From the collection, a DS is picked and moved to the next level of adding concepts to it. If each DS has been picked, our algorithm terminates. The same approach is applied for concept addition in an existing

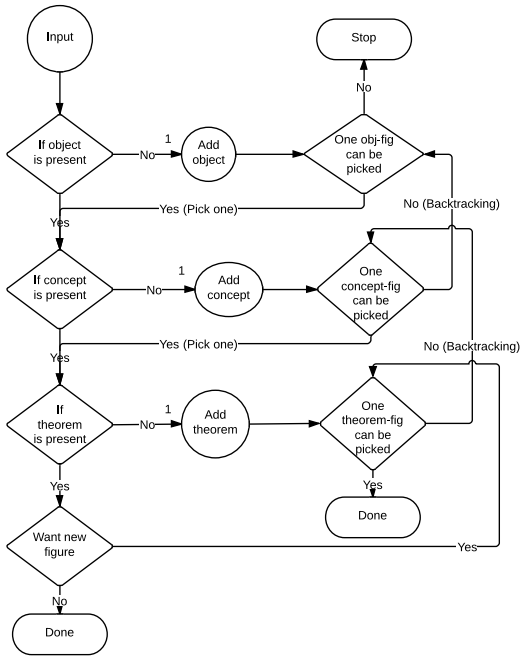


Figure 8: Working of GF component.

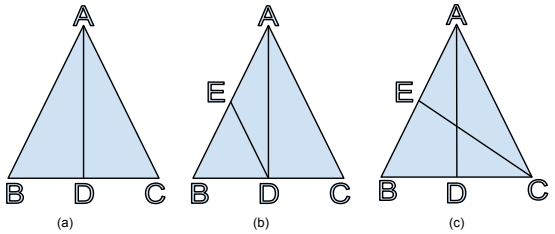


Figure 9: (a) Original figure; (b, c) figures resulting from calling addObject function which adds one more line to the configuration.

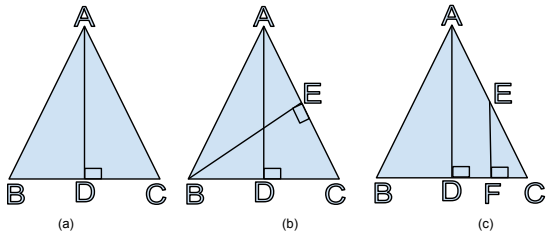


Figure 10: (a) Original figure; (b, c) figures resulting from calling addConcept function which adds one more concept to the configuration.

DS. However, in case all DS formed after concept addition have been picked, backtracking is performed and a new DS is picked from the object-figures. A similar approach is taken when all theorem-figures have been picked, where backtracking leads to choosing a new concept-figure from the previously generated collection.

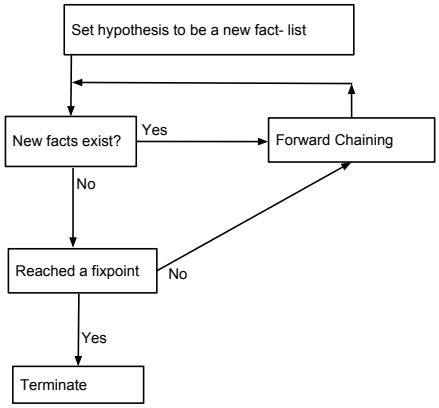


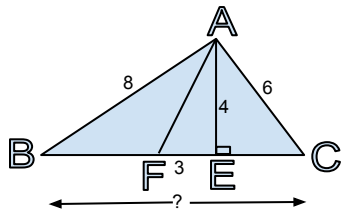
Figure 11: Represents the approach used by GFS component in finding new facts.

4.5 Generating Facts and Solutions from the Configuration (GFS)

This component is responsible for finding the values of unknown variables of the generated figure/configuration from the other two components. This component acts as question generator and solver. The unknown variables whose values have been found represent the generated questions. The steps that leads to finding the unknown variables represent the solution. There can be many ways for finding the values of the unknown variables. In such cases, this component shows all solutions. For generating new facts, it uses a predefined database of theorems. The difference between the theorem database of this component and the previous two components lies in the representation of theorems. In GFS, each theorem is represented in the form of an axiom, while GQ and GF represent theorems in the form of KT and KF, respectively. In addition, GS includes a few basic algebraic theorems, such as $(a+b)^2 = a^2 + b^2 + 2ab$, which are not present in KT and KF. Such algebraic theorems are used inside the axiom system for solving and generating questions which involve algebra.

School Mathematics Study Group (SMSG) axiomatic system is used, which is a combination of Birkhoff's and Hilbert's axiom (Francis, 2002) system. Figure 11 shows the flow diagram of this approach. A fixpoint is a state in which no new facts—in our case, no new assignment of unknown variables—can be deduced from the given facts (Chou et al., 2011). For reaching a fixpoint, forward chaining is used to infer new facts from given facts and rules. The rules used by our system are described in Prolog as follows:

$$Q(x,n_s) : -P_1(x,n_1)...P_k(x,n_k) \qquad (1)$$



(a)

length(B,C,10)
Apollonius Theorem
length (A, B, 8) (Given)
length (A, C, 6) (Given)
length (A, F, 5) (Calculated)
Pythagoras Theorem
length (A, E, 4) (Given)
length (F, E, 3) (Given)

(b)

Figure 12: (a)A question and its solution(b) involving measurement of geometry objects which can be solved with our rules.

which means

$$\forall x[(P_1(x,n_1) \wedge ... \wedge P_k(x,n_k)) \Rightarrow Q(x,n_s)] \quad (2)$$

where $n_1,..,n_k$ and n_s represent numeric values. This representation results in the storage of the integer/real value of the geometry object in last argument of the predicate. With the help of new rules, we can now represent the facts related to dimensions of the geometric objects. Hence this representation of rules can generate geometric questions, which involve measurements of the geometric objects.

In addition, we have added predefined algebraic functions which can be used in rules. One instance of our new rule is shown in Equation 3. The predicate *sq* denotes the square function, which computes the square of a and stores its value in b.

$$sq(a,b) \quad (3)$$

Figure 12 shows an example which can be solved by our rules. Given a triangle ABC, $AE \perp BC$, AB = 8, AC = 6, AE = 4, FE = 3. *our goal is to find the length of BC*. Figure 12(b) provides a solution of this question with the help of our new rules. In addition, these rules help to represent the basic theorems which involve measurement values such as the Pythagorean theorem, the Triangle Inequality theorem and the side/angle ratio theorem in a triangle. Hence it is possible to generate geometric questions based on the above mentioned theorems. For instance, the Pythagorean theorem can be represented in the following way:

$$diff(X,C,0) : -Triangle(a,b,c), sq(a,A), sq(b,B), sq(c,C), sum(A,B,X). \quad (4)$$

A trivial question which requires the Pythagorean theorem for solving would be as follows. *Given a right angle triangle and any two sides of the triangle, the goal is to find the third side.*

Similarly, the Triangle Inequality Theorem can be represented in the following way:

$$lessthan(X,0) : -Triangle(a,b,c), length(a,A), length(b,B), length(c,C), sum(A,B,Z), diff(Z,C,X). \quad (5)$$

A trivial question from this theorem would be as follows. *Given the length of any two sides of a triangle, our goal is to find the maximum value of the third side*.

4.6 Generating New Configurations from an Existing Configuration (GD)

Figure 13 explains the algorithm in the form of a flowchart. It starts with searching for user-selected theorems. In case the user has not chosen any theorem, a theorem is selected from the predefined knowledge database of theorems. However, our algorithm terminates when all theorems have been selected once for a given configuration. After the theorem selection, pattern matching on the theorem figure is performed. If a matching pattern is found, a set of input and/or output values are assigned to the chosen pattern. Assignment is done with the help of a predefined set of theorems. In case the pattern is not found, a new theorem is chosen from KT and the whole process is repeated until we get the desired configuration.

Depending on the requirements, several theorems can be involved in a given configuration. In addition, it is possible to use a single theorem multiple times. The method requires the concept of using the output of one theorem as the input of another one. In this way we can have all possible combinations of predefined theorems. Figure 14 shows an instance of assigning unknown variables. Figure 14b represents the new data assigned to Figure 14a. It is generated with the help of applying Pythagoras Theorem in triangle ABE. The output is changed from BE to AE. Output is considered the length of side calculated using the Pythagoras Theorem from a given input. The input and output for the Pythagoras Theorem is taken using KT. Figure 14c is generated by adding more data in Figure 14a. New data is assigned by applying the

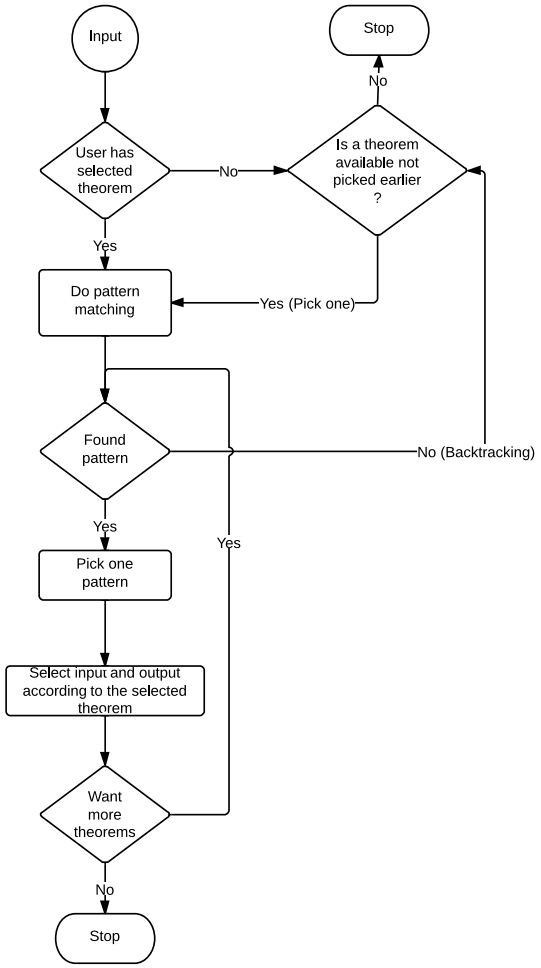


Figure 13: Working of GD algorithm.

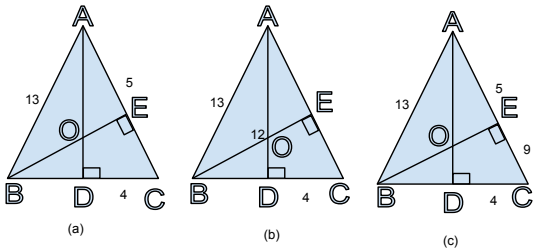


Figure 14: (a) A figure and data is assigned to few geometry objects. (b) Output of calling the changeData function. (c) Result obtained by calling the addData function on (a).

Pythagoras Theorem in the triangle ADC. The theorem requires two sides as input to generate one output. Hence length of side EC is given.

In the above algorithm, it may happen that redundant data is provided in the figure. Redundant data refers to the values assigned to properties of some of the geometric objects which are not required and can be derived from the previously given data and predefined theorems. Figure 15 explains this situation with the help of an example. In this figure, one data item is redundant and any one of the length values can be removed, since, by applying the similarity concept, the third length can be derived from the other two lengths. This type of reasoning may lead to an increase of the difficulty level of the generated questions. This reasoning is added with the help of GFS, which is described in Section 4.5.

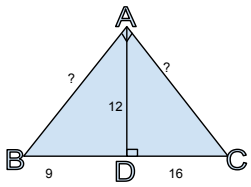


Figure 15: An example in which redundant data is provided in the given figure.

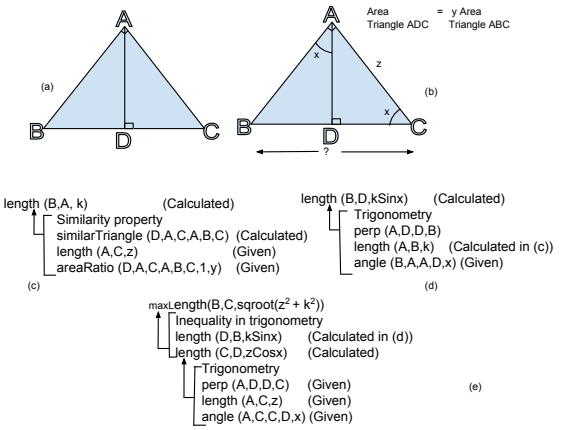


Figure 16: (a) The figure is generated by using *triangle* and *perpendicular* as the geometry objects using the GF component. (b) The data is generated for (a) using the GD component such that the two triangles ADC and ADB become similar. (c, d, e) The new fact and its derivation using the GFS component is generated from the figure shown in (b).

Figure 13 contains non-deterministic decisions at various steps. The non-deterministic selection of options can be avoided by adding more heuristics such as fixing the number and types of theorems to be involved.

5 FURTHER EXAMPLES

This section explains our framework through more examples covering various concepts and geometric theorems.

5.1 First Example

The input given in this question generation task is as follows

- Object: triangle and line segment
- Concept: perpendicular
- Theorem: similarity, trigonometry and Inequality
- Number of questions: 1

The input given in the first example covers the concept of *perpendicular lines*. The generated questions are required to make use of theorems involving *similarity of triangles, trigonometry and inequality*. Figure 16 shows the step by step execution of the algorithm for the given input. Figure 16a generates a configuration involving *triangle and perpendicular* geometry objects using the GF component. This figure is generated by non-deterministically selecting a way of generating perpendicular lines in a triangle from our predefined database.

Figure 16b is generated using GD component with the help of KT in the figure 16a. There are various ways of making two triangles similar. Firstly, this component performs pattern matching to find the suitable ways of making two triangles similar in the existing figure. Later, non-deterministically, it picks one way of making triangles similar and generates data accordingly. Now GD component assigns data using the trigonometry rules stored in the database. Currently, the database is limited to the basic trigonometry rules related to the sides and angles of a triangle. For example, "$\sin 2\theta = 2\sin\theta\cos\theta$" trigonometry rule does not includes any sides and angles in a triangle. Hence, it is not included in our database.

Figure 16c shows the new facts and the their reasoning generated using the GFS component. The new fact is the new question and the reasoning is the solution of the generated question.

5.2 Second Example

The input given in this question generation task is as follows

- Object: triangle and line segment
- Concept: parallel line
- Theorem: similarity and ratio of length of sides
- Number of questions: 1

The input given in the second example covers the concept of *parallel line*. The generated questions are required to make use of theorems involving *similarity and ratio of length of sides* of a triangle. Figure 17 shows the step by step execution of the algorithm for

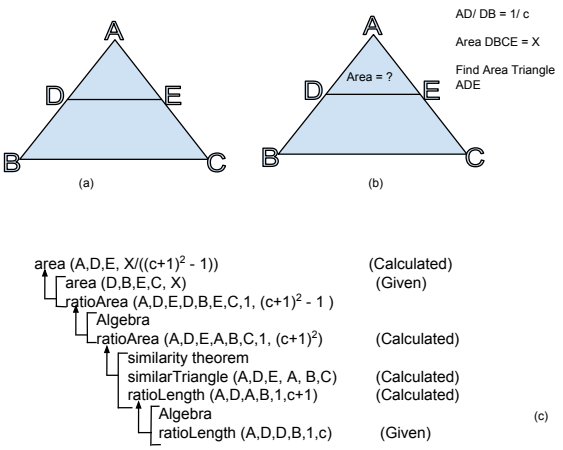


Figure 17: (a) The figure is generated by using *triangle* and *parallel line* as the geometry objects using the GF component. (b) The data involving ratio of length of sides is generated for (a) using the GD component. (c) The new fact and its derivation using the GFS component is generated from the figure shown in (b).

the given input. Figure 17a generates a configuration involving *triangle and parallel line* geometry objects using the GF component. Before generating similarity of two triangles, it finds similar triangles in the existing configuration. It can be seen that the existing configurations has similar triangles ADE and ABC. Hence, there is no need to generate similar triangles. Application of similarity theorem results in the generation of ratios of sides of length of two triangles. Figure 17b is generated using GD component with the help of KT knowledge database. Figure 17c shows the new facts and the their reasoning generated using the GFS component. The new fact is the new question and the reasoning is the solution of the generated question.

5.3 Third Example

The input given in this question generation task is as follows

- Object: triangle and line segment
- Concept: perpendicular and angle bisector
- Theorem: Sum of angle in a triangle
- Number of questions: 1

The input given in the third example covers the concept of *angle-bisector*. The generated questions are required to make use of theorems about sum of angles in a triangle. Figure 18 shows the step by step execution of the algorithm for the given input. Figure 18a generates a configuration involving *triangle and perpendicular* geometry objects using the GF

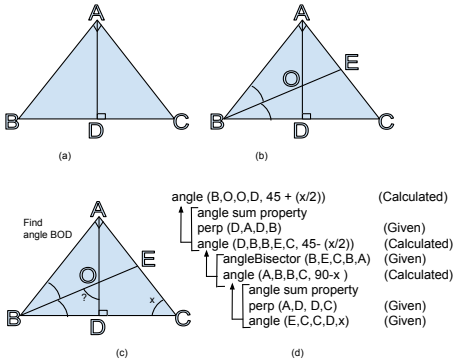


Figure 18: (a) The figure is generated by using *triangle* and *perpendicular* as the geometry objects using the GF component. (b) *Angle-bisector* is added to (a) using the GF component.(c) The data involving angles in a triangle is generated for (b) using the GD component. (d) The new fact and its derivation using the GFS component is generated from the figure shown in (c).

component. Figure 18b generates a new configuration from (a) involving *angle-bisector* geometry object. This figure is generated by non-deterministically selecting a way of generating angle-bisector in a triangle. The decision of generating perpendicular before angle-bisector is taken randomly. Figure 18c is generated using GD component with the help of KT knowledge database. There are many triangles in the existing configuration. Hence, there are various ways of assigning data to satisfy angle-sum property in these triangles. GD component picks one triangle and generates the data accordingly. Figure 18d shows the new facts and the their reasoning generated using the GFS component. The new fact is the new question and the reasoning is the solution of the generated question.

6 IMPLEMENTATION

Each component of our tool is implemented independently, using state-of-the-art libraries and systems. Various programming languages are used in the system. C++ is used for performing calculations and Python is used for implementation of the algorithms used in GF and GD components. The algorithm in GFS component is implemented using Constraint Handling Rules (CHR) (Frühwirth and Raiser, 2011). CHR are used for generating new facts from the axioms and the given facts. In our implementation, we use the CHR library provided by K.U.Leuven, on top of SWI-Prolog (Schrijvers and Demoen, 2004). The theorems used in GFS component are manually converted in the format used by CHR library. For implementing knowledge representation KF and KT, the graph database Neo4j (Vicknair and Macias, 2010) is

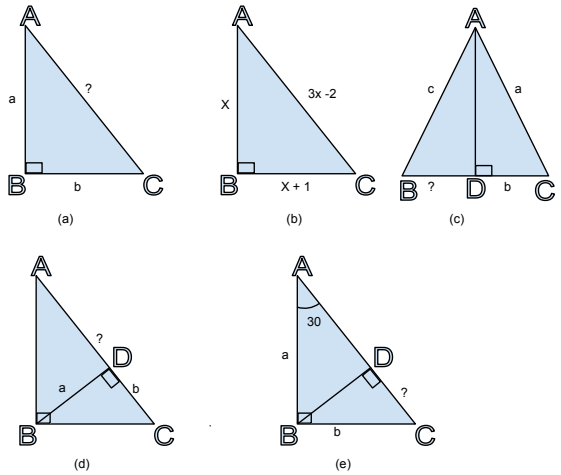


Figure 19: Generated questions based on "triangle", "perpendicular" and "Pythagorean Theorem" as input. Questions in figure(a-c)can be solved using Pythagorean Theorem only. However, questions of figure (d) requires similarity and figure (e) requires trigonometry for finding the unknown value.

used. KF knowledge graph is used and modified by all the components and finally represents the question. Our knowledge databases such KT and predefined ways of intersection of geometric objects are manually generated and stored before the questions generation.

6.1 Experimental Results

The system can generate geometry questions using the framework described in Section 4. Currently, our knowledge database of objects contains line segments and triangles. In addition, we have a predefined set of more than 100 theorems. The generated questions cover five categories, e.g. similarity. Figure 19 shows various questions generated by our system on selecting "*triangle*" as object, "*perpendicular*" as concept and "*Pythagorean Theorem*" as a theorem to be covered. The generated questions are tested using the existing geometry solver tool JGEX (Gao and Lin, 2004). For testing in JGEX, the figure configuration is drawn manually by the user and the system is asked to prove/find a certain relationship. The tool is able to prove/solve all the questions generated by our system. Comparing the solutions generated by JGEX with the solutions generated by our system, we found interesting differences that may stem from different representations of geometric knowledge and reasoning techniques and that deserve further investigation.

7 CONCLUSION

In this paper, we provide a framework for the automatic generation and solving of questions for high school mathematics, specifically in the geometry domain. Our system is able to quickly generate large numbers of questions on specific topics. Such a system will help teachers reduce the time and effort spent on the tedious and error-prone task of generating questions. Our work aims to develop an automated geometry question generation system that uses a deductive approach for finding the relations between mathematical concepts and for generating and proving these conjectures about concepts.

Future work can be carried in various directions. An experiment needs to be performed in which the generated problems would be placed in front of teachers and let them guess which problem is from a textbook and which one was generated from our algorithm. Other major work would be generating questions according to the required difficulty level. Another improvement would be the addition of knowledge by the user. Lastly, a GUI should be developed in order to make it usable for teachers, students and testers.

REFERENCES

Chou, S. C. and Gao, X. S. (2001). *Handbook-of-Automated-Reasoning*. Elsevier and MIT Press.

Chou, S. C., Gao, X. S., and Zhang, J. Z. (1994). Machine proofs in geometry: Automated production of readable proofs for geometry theorems.

Chou, S. C., Gao, X. S., and Zhang, J. Z. (2000). A deductive database approach to automated geometry theorem proving and discovering. *J. Autom. Reason.*, 25(3):219–246.

Chou, S. C., Gao, X. S., and Zhang, J. Z. (2011). Area method and automated reasoning in affine geometries.

Coelho, H. and Pereira, L. (1986). Automated reasoning in geometry theorem proving with prolog. *Journal of Automated Reasoning*, 2(4):329–390.

Francis, G. (2002). Axiomatic systems for geometry.

Frühwirth, T. and Raiser, F., editors (March 2011). *Constraint Handling Rules: Compilation, Execution, and Analysis*.

Gao, X.-S. and Lin, Q. (2004). Mmp/geometer asoftware package for automated geometric reasoning. In Winkler, F., editor, *Automated Deduction in Geometry*, volume 2930 of *Lecture Notes in Computer Science*, pages 44–66. Springer Berlin Heidelberg.

Gao, X. S., Zhu, C. C., and Huang, Y. (1998). Building dynamic mathematical models with geometry expert, i. geometric transformations, functions and plane curves. In *Proceedings of the Third Asian Technology Conference in Mathematics*.

Greeno, G., J., Magone, and Maria E. Chaiklin, S. (1979). Theory of constructions and set in problem solving. *Memory and Cognition*, 7:445–461.

Jiang, J. and Zhang, J. (2012). A review and prospect of readable machine proofs for geometry theorems. *Journal of Systems Science and Complexity*, 25:802–820.

McCharen, J. D., Overbeek, R. A., and Wos, L. A. (1976). Problems and experiments for and with automated theorem-proving programs. *IEEE Trans. Comput.*, 25(8):773–782.

Meikle, L. and Fleuriot, J. (2003). , formalizing hilberts grundlagen in isabelle/isar. In *Theorem Proving in Higher Order Logics*.

Narboux, J. (2010). The area method: a recapitulation. *Journal of Automated Reasoning*, 48:489–532.

National Council of Teachers of Mathematics (1989). Curriculum and evaluation standards for school mathematics.

Nevins, A. (1975). Plane geometry theorem proving using forward chaining. *Artificial Intelligence*, 6(1):1–23.

Quaife, A. (1989). Automated development of tarskis geometry. *Journal of Automated Reasoning*, 5:97–118.

Schrijvers, T. and Demoen, B. (2004). The k.u.leuven chr system: Implementation and application.

Vicknair, C. and Macias, M. (2010). *A Comparison of a Graph Database and a Relational Database*.

Wilson, S. and Fleuriot, J. D. (2005). Combining dynamic geometry, automated geometry theorem proving and diagrammatic proofs. In *European joint conference on Theory And Practice of Software - ETAPS*.

Zhang, J. Z., Gao, X. S., and Chou, S. C. (1996). The geometric information search system by forward reasoning. *Chinese Journal of Computers*, 19(10):721727.