THE INTERNATIONAL REVIEW OF RESEARCH IN OPEN AND DISTANCE LEARNING

Drawing Dynamic Geometry Figures Online with Natural Language for Junior High School Geometry

Wing-Kwong Wong,[1] **Sheng-Kai Yin,**[2] and **Chang-Zhe Yang**[1]
[1]National Yunlin University of Science & Technology, Taiwan, R.O.C.
[2]Mingdao University, Taiwan, R.O.C.

Abstract

This paper presents a tool for drawing dynamic geometric figures by understanding the texts of geometry problems. With the tool, teachers and students can construct dynamic geometric figures on a web page by inputting a geometry problem in natural language. First we need to build the knowledge base for understanding geometry problems. With the help of the knowledge base engine InfoMap, geometric concepts are extracted from an input text. The concepts are then used to output a multistep JavaSketchpad script, which constructs the dynamic geometry figure on a web page. Finally, the system outputs the script as an HTML document that can be visualized and read with an internet browser. Furthermore, a preliminary evaluation of the tool showed that it produced correct dynamic geometric figures for over 90% of problems from textbooks. With such high accuracy, the system produced by this study can support distance learning for geometry students as well as distance learning in producing geometry content for instructors.

Keywords: Natural language understanding; dynamic geometry; JavaSketchpad; geometry education; knowledge construction

Introduction

One approach to distance learning is to put learning content on web pages and ask learners, who may be in different geographical locations, to engage in some learning activities while browsing the pages. However, learning content takes a lot of time and human resources to produce, especially when the content involves geometric figures. Thus, producers of learning content would be very happy if they could use a tool to automatically convert an input text of domain concepts, such as a geometry problem, into a target output format such as some programming code to draw geometric figures on web pages. This article proposes a methodology of how to convert input texts of domain concepts into some target output format. The methodology is illustrated by the construction of an actual system in converting input texts of geometry problems at a junior high school level into JavaSketchpad scripts to display dynamic geometry (DG) figures on a web page.

The National Council of Teachers of Mathematics (NCTM) has published two important documents on K-12 mathematics curriculum: *Curriculum and Evaluation Standards for School Mathematics* (1989) and *Principles and Standards for School Mathematics* (2000). The latter focused more on the skill of writing formal proofs of geometry (Knuth, 2002). Furthermore, mathematicians and educators agree that writing geometry proofs involves important skills that are difficult to learn (Koedinger, 1998; Whiteley, 1999).

Using geometry software increases the motivation of students in learning geometry. Some popular programs include Geometer's Sketchpad (GSP, http://www.keypress.com/sketchpad/), Cabri Geometry II (http://www.cabri.com/v2/pages/en/index.php), Geometry Expert (Chou et al., 1996), and Cinderella's Café (http://www.cinderella.de/tiki-index.php). These programs share one common focus on dynamic geometry figures. In a dynamic geometry figure, students can drag a geometry object such as a vertex of a triangle and change the figure dynamically while preserving the geometric invariants, which are the consequences of the given conditions. Thus these programs are commonly used to demonstrate geometry theorems.

Dynamic geometry can also be used by students to discover conjectures about a figure in some given conditions. Students can explore the variations of a dynamic geometry figure and try to discover conjectures about the figure on their own (Jones, 2000). In a study, students are given work sheets to fill in measurements of some properties of geometric objects, and they write down conjectures they discover in a dynamic figure (Furinghetti & Paola, 2003). There are two common problems in such learning activities. First, teachers and students have to learn to use the DG software and the learning process can be difficult. Second, there is no detailed record on how students manipulate the dynamic figures in order to support the findings of the study on these activities. Both problems make it difficult to evaluate the learning effectiveness of the students in using a dynamic geometry system.

To address the above two problems, we built an online dynamic geometry system that can "understand" geometry problems. With this system, the teachers and students can construct dynamic geometric figures on a web page for their learning activities, such as making geometry conjectures and proving a theorem. Furthermore, this tool can help researchers to design user interfaces that are capable of recording the data of students' interactions with the dynamic geometric figures. Based on the data, researchers can get more insights on how students make inferences from their interactions with the dynamic figures.

In addition to benefitting students, the online dynamic geometry system can be a valuable tool for instructors who struggle to produce the learning content of dynamic geometry. Specifically, two types of instructors can use the system in ways that suit their needs. The first type is instructors who do not want to work with JavaSketchpad (JSP). They can directly use the web pages of learning content generated by the system or enhance those pages with other tools such as Adobe's Dreamweaver. The second type is instructors who want to learn JSP. They can input a problem text of their choice to the system and get a web page of JSP with the geometry content. Then they can study how the JSP code produces the dynamic geometry content they pick. This can be a good way to learn JSP by example. How effective this tool is for instructors and learners of geometry really depends on the accuracy of the system in producing correct geometry figures. Thus, an objective of this study is to evaluate the accuracy of the system empirically.

This system uses InfoMap (Hsu et al., 2001), which is a knowledge engineering tool for understanding natural language. InfoMap analyzes a geometry text by extracting geometric concepts and converting them into JavaSketchpad commands for drawing a dynamic figure. Then the system embeds the script in a web page, which can show the dynamic geometry figure on an internet browser. The article is organized as follows. The literature on attempts to solve the problem of learning JSP is first reviewed. Based on our previous study (Wong et al., 2007), a methodology and the core technologies of the system are presented. We also empirically tested the system with geometry problems from three main reference books for junior high schools in Taiwan. At the end of the article, some empirical results and conclusive remarks are reported, and future work is suggested.

Roles of DGE and Instructional Design in Geometry Learning

In research literature, there is yet no satisfactory explanation of how experimentation with dynamic geometry can help the acquisition of skills for formal theorem proving. Researchers think there is a big gap between the experience of dynamic geometry and the learning of formal proof production. Some studies also found that object dragging in a dynamic geometry environment (DGE) can reduce the gap between dynamic geometry experimentation and the generation of theorem proving ideas by learners. As a result, many researchers design some activities in a DGE and study what types of learning result from such activities and the nature of the learning process (e.g., Leung & Lopez-Real, 2003; Hoyles & Healy, 1999; Furinghetti & Paoloa, 2003; Christou et. al., 2004).

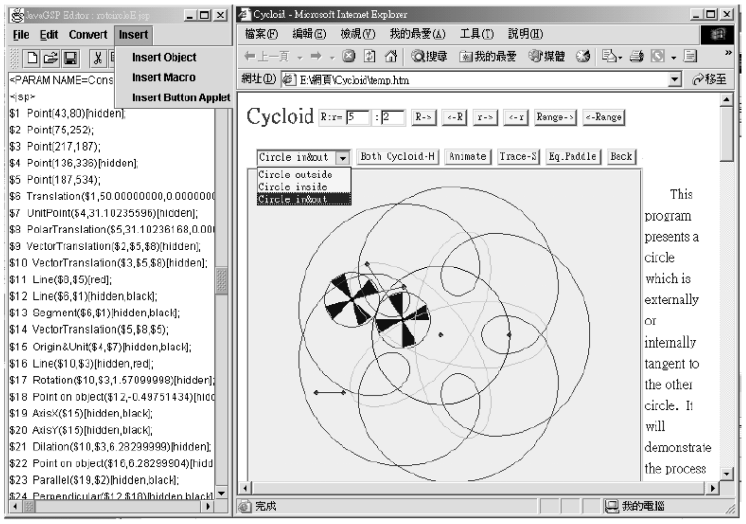
Some studies indicate that when students explored conjectures in a DGE, they could explain the formal proof they wrote based on their experiences in the exploration (e.g., Holys & Healy, 1999). Furthermore, students would strengthen their beliefs in the geometry conjectures they made from their observation of the changes of dynamic figures in a DGE (de Villiers, 1996, 2003). In a DGE, students can drag geometric components and take measurements of geometric objects in a dynamic figure. Then they can notice the variance and invariance of conditions in a dynamic figure, deepening their understanding of geometry theorems (Laborde et al., 2006).

The ultimate goal of geometry learning is to design a DGE that helps students learn the skills of theorem proving in geometry. But current DGEs are not designed for this purpose since they do not provide any tools with instructional strategies for theorem proving. Although a DGE is used in math classes in some schools, many math teachers and researchers think that it is difficult to design instructional materials for a DGE, and students might spend too much time in exploring without achieving the final learning objective. Therefore in the design of a DGE with specific teaching objectives, educators have to pay special attention to the design of interfaces for instructor to author guiding instructions and for learners to proceed to the final learning objective through guided exploration. Thus, we need to develop a system which can support the user interface design of a DGE for specific learning purposes.

Problems Using a DGE in Class

When a DGE is used in some learning activities, students need both basic geometry knowledge and the knowledge to work in a DGE. Sometimes, they need to add some geometric components in a dynamic figure. This can be an obstacle to some students, reducing the effectiveness of learning in a DGE (Talmon & Yerushalmy, 2004). Despite the potential benefits of using a DGE in geometry classes, a DGE is not available to some schools in Taiwan. We found that in four junior high and high schools, located in urban areas of Yunlin County, teachers have never used a DGE in their classes. We believe many other schools have similar experiences. Three reasons are suggested for not using a DGE in school. First, using a DGE might not contribute directly to the performance of students in public examinations for entering the next level of schooling. Second, funds are difficult to get for computer equipment. Third, too much time is required for teachers and students in learning how to construct dynamic figures in a DGE.

In order to address the third problem, some tools are developed to train users to learn to write JavaSketchpad (JSP, http://www.keypress.com/sketchpad/java_gsp/) script. On the web page http://www.mathematik.uni-bielefeld.de/~lisken/jsp/, Lisken provides a tool, jsp.awk, on which an author can write JavaSketchpad code. When coding is done, the tool embeds the code in an HTML file, which the author can view to check the dynamic figure with a browser. On the web page http://home.wxs.nl/%7ehklein/jspgenerator/jspgenerator.htm, Klein provides a similar tool called JSPGenerator, which is an authoring tool written in Javascript to generate JavaSketchpad files. After an author finishes writing Javascript code on JSPGenerator, he can choose to view the dynamic figure generated by the code in the same window just below the code. This previewing step is an improvement over jsp.awk. In Taiwan, Lin (2006) offers a similar tool called JavaGSP Editor, with which learners can write JavaSketchpad scripts and use these scripts in an online learning environment (Figure 1). The tool provides fancy graphical interface components such as buttons and menus for designing more interactive web pages. The above three tools simplify the task of producing dynamic figures on web pages. However, these tools still require users to learn JavaSketchpad's programming syntax and semantics, which can put off many instructors and students in high school.


JavaGSP Editor : rotcircleE.jsp
File Edit Convert Insert
Insert Object
Insert Macro
Insert Button Applet
Cycloid - Microsoft Internet Explorer
網址(D) E:\網頁\Cycloid\temp.htm
Cycloid R:r= 5 : 2 R-> <-R r-> <-r Range-> <-Range
Circle in&out Both Cycloid-H Animate Trace-S Eq.Paddle Back
Circle outside
Circle inside
Circle in&out
完成
我的電腦

```
<PARAM NAME=Cons
<jsp>
$1  Point(43,80)[hidden];
$2  Point(75,252);
$3  Point(217,187);
$4  Point(136,336)[hidden];
$5  Point(187,534);
$6  Translation($1,50.00000000,0.0000000
$7  UnitPoint($4,31.10235596)[hidden];
$8  PolarTranslation($5,31.10236168,0.00
$9  VectorTranslation($2,$5,$8)[hidden];
$10 VectorTranslation($3,$5,$8)[hidden];
$11 Line($8,$5)[red];
$12 Line($6,$1)[hidden,black];
$13 Segment($6,$1)[hidden,black];
$14 VectorTranslation($5,$8,$5);
$15 Origin&Unit($4,$7)[hidden,black];
$16 Line($10,$3)[hidden,red];
$17 Rotation($10,$3,1.57099998)[hidden];
$18 Point on object($12,-0.49751434)[hidd
$19 AxisX($15)[hidden,black];
$20 AxisY($15)[hidden,black];
$21 Dilation($10,$3,6.28299999)[hidden];
$22 Point on object($16,6.28299904)[hidd
$23 Parallel($19,$2)[hidden,black];
$24 Perpendicular($12,$18)[hidden,black]
```

This program presents a circle which is externally or internally tangent to the other circle. It will demonstrate the process

Figure 1. JavaSketchpad Editor (the left is Editor, viewer in the right (Lin, 2006)).

To make a DGE more accessible to teachers and students, we propose a system that can draw dynamic geometry figures by understanding texts of geometry problems. In this way, teachers and students are not required to spend so much time in constructing figures in a DGE. Moreover, if the system is available on a Web site, then there is no need to install any expensive commercial software in school, making a DGE more accessible to schools in poor school districts.

Natural Language Understanding for Computer-Assisted Learning

Natural language understanding is a challenging problem in the research of artificial intelligence. Some missions of research on natural language understanding are to analyze and comprehend human language and answer questions about given texts. With the progress of the research of natural language understanding, these technologies have been applied to various fields, including semantic web, Chinese speech processing, machine translation, concept modeling, knowledge engineering, and computer-assisted learning. In a study, users can use natural language instead of formal commands to work in the UNIX system (Lees & Cowie, 1996). In a study by Li & Chen (1988), an expert system was constructed for helping users learn concepts of computer science. In studies by Wong et al. (2007, 2008), a knowledge model of geometric concepts was used to understand word problems of geometry proofs.

Lees and Cowie (1996) proposed an enquiry system for training students to learn UNIX commands. The system provides a natural language interface for learners so that they can learn UNIX commands by themselves. After a learner inputs a sentence or a UNIX command, the system will parse the sentence with a chart parsing algorithm and generate a UNIX command as a response. The system checks with the learner in a dialog and executes the command if that is what the learner really wants to do with the input sentence. Li and Chen (1988) proposed a Chinese enquiry system about fundamental knowledge of computers. The system uses a linguist string parser to understand the question inputted by a learner, and then outputs an appropriate answer.

In Lu et al. (2005), a model is proposed to simulate the procedural knowledge of basic arithmetic operations. The model helps teachers design an appropriate curriculum and teaching strategy from the records of procedure that students used to solve arithmetic problems. The model is used in an intelligent tutoring system that can accumulate and reproduce the knowledge from teachers and students and help teachers build a good learning map for students. Furthermore, a student model, which is built from a collection of students' errors, can contribute to the design of more suitable teaching tactics.

In Wong et al. (2007), a LIM-G (learners' initiated model for geometry) system is used to understand geometry word problems and help elementary school students comprehend geometry word problems, which are about the area or circumference of various shapes. After a student inputs a geometry problem to LIM-G, the system understands the geometry problem using prebuilt geometry knowledge and constructs a figure for the problem. Geometry word problems from five textbooks published by five major publishers in Taiwan were used to evaluate the performance of LIM-G and about 85% of the problems were comprehended correctly.

In LIM-G, a cognitive knowledge base is constructed with an ontology-based knowledge engineering tool called InfoMap (Hsu et al., 2001), whose knowledge base includes generic template nodes for problem classes, problem-concept, lexical knowledge, lexicon, and so on. Using a template matching mechanism, InfoMap can extract the attributes and values of concepts in a given problem. After reviewing LIM-G and other similar studies, Mukherjee and Garain (2008a) point out that the methodology of natural language understanding technology and its application to understanding geometric problems in mathematics is mature for developing real applications. In our study, InfoMap is used to understand an input problem of geometry proof by extracting the geometric objects, resulting in a JSP script that draws the figure of the problem as a dynamic figure. The next section provides a methodology that other researchers can follow to develop similar systems for their learning areas.

Methodology

Applying the model of LIM-G (Wong et al., 2007), Mukherjee and Garain (2008b) implemented a tool for the automatic conversion of any input text about science and engineering into a concept map. In Mukherjee and Garain (2011), a knowledge base called GeometryNet was used in interpreting the geometric meaning of an input text to draw the corresponding diagram. These studies indicate the feasibility of a general approach in converting any input text about some subject domain, which is intended for some educational context, into a target output format such as JSP and concept map.

By generalizing and extending the model of LIM-G, we suggest a five-step methodology that researchers can adopt in automatic construction of figures of learning content in any subject domain (Figure 2). The first step is to construct a knowledge base (for example, to add geometric concepts and templates of geometric problems into InfoMap). The second is to adopt the basic mechanism of text understanding as shown in this article. Sometimes, the researchers have to apply heuristics to improve the accuracy of text understanding. The third step is to analyze the output format (e.g., JavaSketchpad script and HTML). The fourth is to design learning activities in which learners use the content. The last step is to evaluate the learning effectiveness with empirical experiments. Based on the empirical results, the developer may need to add more concepts and heuristics for the first two steps. Then the cycle can be repeated until the researchers are satisfied. Following this methodology, we develop a system to convert texts of geometry problems into dynamic geometry figures embedded in web pages. The system architecture is described next.

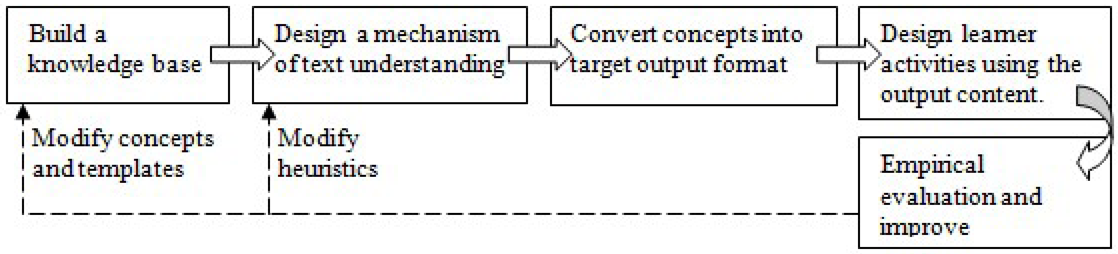

Build a knowledge base
Design a mechanism of text understanding
Convert concepts into target output format
Design learner activities using the output content.
Empirical evaluation and improve
Modify concepts and templates
Modify heuristics

Figure 2. A five-step methodology for converting a text problem into a figure.

System Architecture

This section describes the overall architecture and user-interface of the system for generating dynamic geometry figures from input geometry problems. There are two main components in this system (Figure 3). The first component is the knowledge engine InfoMap. When a user enters a geometry problem in natural language, InfoMap analyzes the problem and extracts the attributes and values of the geometric concepts in the problem. This information is sent to the second component of the system, which is a script generator. This component generates a JSP script that draws a dynamic geometric figure of the problem embedded in an HTML document, which can be loaded by any web browser to display the dynamic geometry figure.

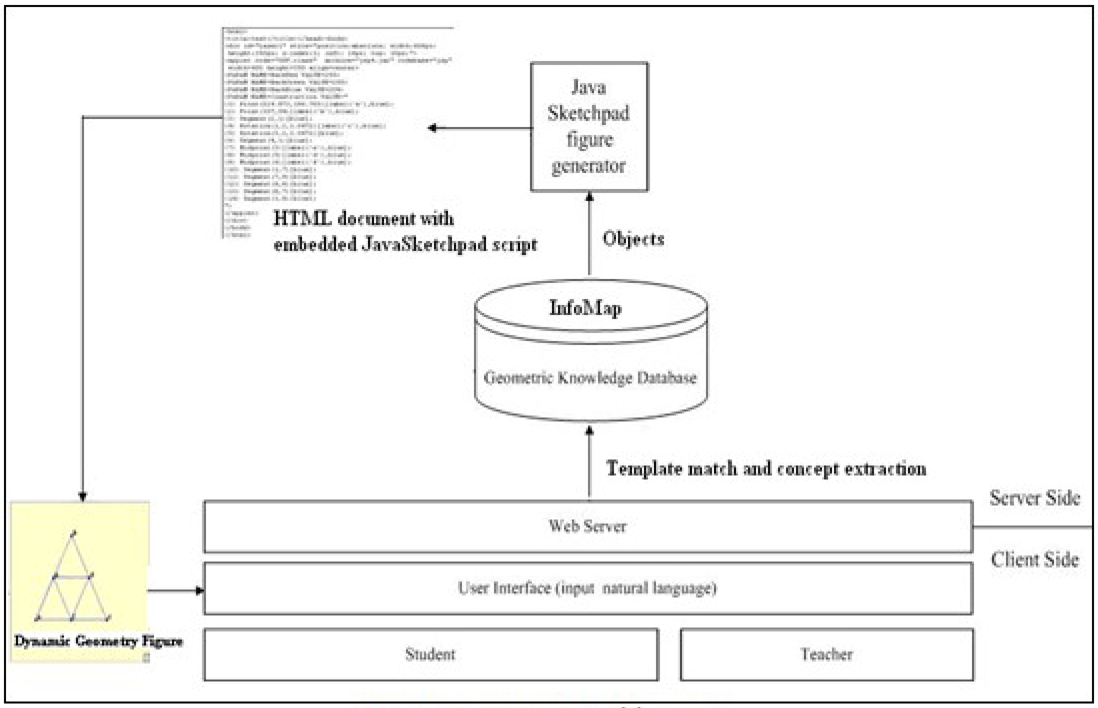
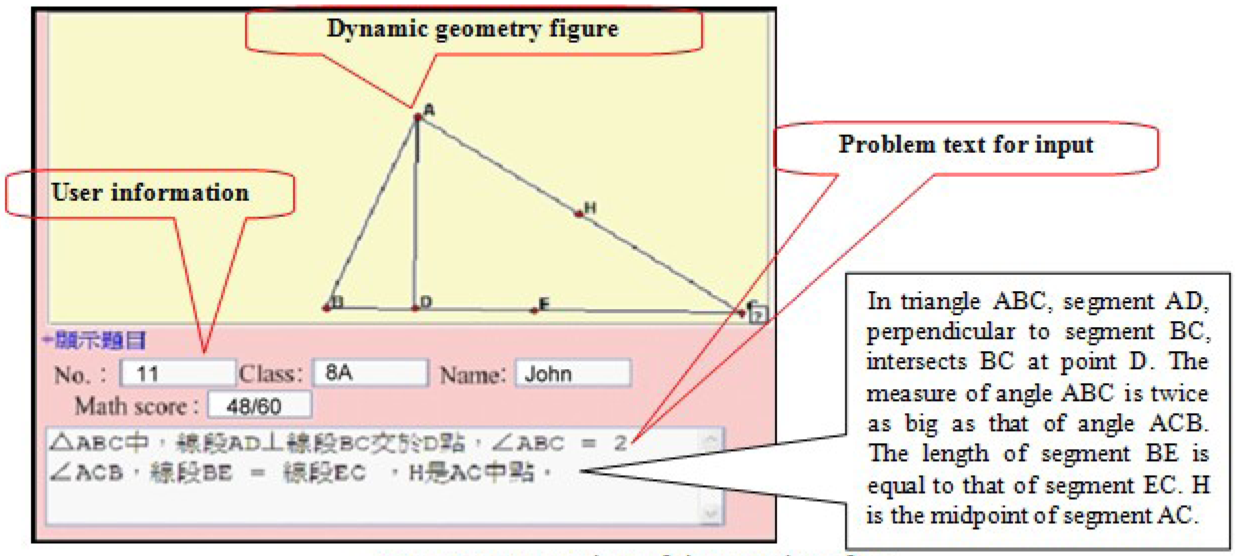

Java Sketchpad figure generator
HTML document with embedded JavaSketchpad script
Objects
InfoMap
Geometric Knowledge Database
Template match and concept extraction
Server Side
Client Side
Web Server
User Interface (input natural language)
Dynamic Geometry Figure
Student
Teacher

Figure 3. System architecture.

Figure 4 shows a snapshot of the user-interface in a web browser. A user can input the geometry problem in Chinese and simple mathematical symbols in the text area at the bottom. The canvas at the top displays the dynamic figure based on the system's analysis of the input problem. The figure is drawn with a JavaSketchpad script embedded in an HTML document.



No. : 11 Class: 8A Name: John
Math score : 48/60

△ABC中，線段AD⊥線段BC交於D點，∠ABC ＝ 2∠ACB，線段BE ＝ 線段EC ，H是AC中點，

In triangle ABC, segment AD, perpendicular to segment BC, intersects BC at point D. The measure of angle ABC is twice as big as that of angle ACB. The length of segment BE is equal to that of segment EC. H is the midpoint of segment AC.

Figure 4. A snapshot of the user-interface.

InfoMap

InfoMap is a knowledge engineering tool provided by the Intelligent Agent System Lab, Institute of Information Science, Academia Sinica. InfoMap is an ontology-based system for knowledge representation and template matching (Hsu et al., 2001). InfoMap works as an agent by understanding texts in any domain and can answer questions about them if the needed domain knowledge is provided. In order to use the tool for this study, we must first build the basic knowledge for geometry problems.

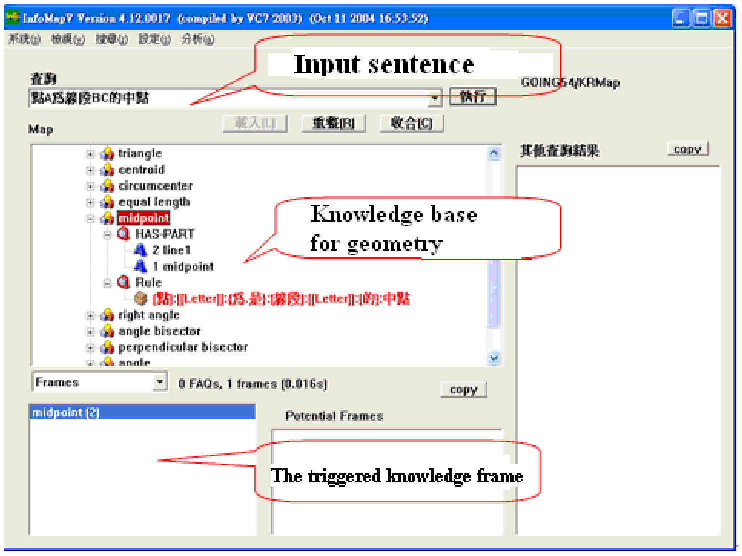


Figure 5. Inquiring a knowledge base of InfoMap.

In a knowledge base in InfoMap, nodes represent geometry concepts and each template of a node specifies the syntax of a sentence that involves the concept of the node. Templates are matched to input sentences in order to extract the concepts from the sentences. Figure 5 shows a tool for getting information from the knowledge base of InfoMap. Before the system can parse an input sentence, the system must load a knowledge base of geometry concepts in InfoMap. A user first inputs a sentence in the text area at the top. Then InfoMap triggers the templates that match the input problem, the parent nodes of the templates and the referenced nodes, extracting these nodes and their contents. Users can see the details of the knowledge base and the triggered nodes, which are colored in red (Figure 5).

Knowledge Base of Geometric Concepts

Before InfoMap can perform the understanding task, we need to build a knowledge base of geometry concepts first. Figure 6 shows part of the knowledge base, which includes many concepts (e.g., midpoint, pedal point, intersection, triangle, isosceles triangle, regular triangle, parallelogram, parallel line, point on line). In this study, we have built more than 50 nodes of geometric concepts. We can always add more concepts when needed.

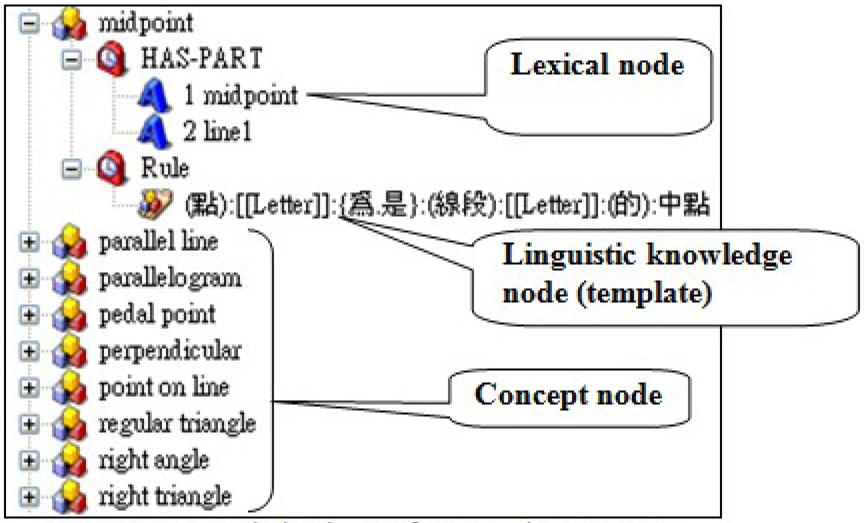


Figure 6. Knowledge base of geometric concepts.

In a knowledge base, a concept node is also a knowledge frame, which includes a rule node and two attribute nodes. The rule node generally includes multiple templates, which describe the syntax of possible sentences about the concept. The HAS-PART node specifies the component nodes that make up the concept. The component nodes can store the concepts and their names which are extracted from an input sentence. For example, the content of midpoint has two components, midpoint and line1 (Figure 6). The former component refers to the midpoint of the latter component line1.

Take midpoint as an example, when the user inputs a Chinese sentence meaning "Point A is the midpoint of segment BC" or "A is the midpoint of BC". The sentence is matched against the InfoMap template of midpoint, and the node "midpoint" is triggered. InfoMap will extract the component concepts of midpoint A and segment BC and then label the component "midpoint" as "A" and component "line1" as "BC". Table 1 shows the input sentence and its matched result. The templates in InfoMap are quite flexible in matching synonyms, extraneous words, and optional words. The construction of the templates is a time-consuming knowledge engineering task.

Table 1

Result of Concept Matching for Midpoint

Input sentence	Concept node	Lexical node
"Point A is the midpoint of segment BC" or "A is the midpoint of BC"	midpoint	1 midpoint=A 2 line1=BC

Consider the template for matching sentences about midpoint for example. The template for an equivalent English sentence is "(Point) [[Letter]] is (the) midpoint (of) (segment) [[Letter]]". So the sentences "Point A is the midpoint of segment BC" (sentence 1) and "A is the midpoint of BC" (sentence 2) both match the template of the "midpoint" concept. The matched results are shown in Tables 2 and 3. The label "NULL" means the word is missing.

Table 2

Result of Template Matching for Sentence 1

Geometry proof description	Point	A	is	the	midpoint	of	segment	BC
Template	(Point)	[Letter]	is	(the)	midpoint	of	segment	[[Letter]]

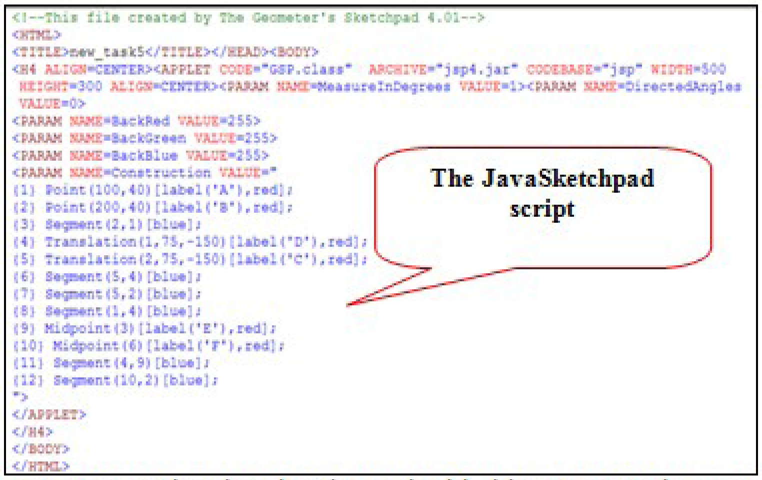
Table 3

Result of Template Matching for Sentence 2

Geometry proof description	NULL	A	is	the	midpoint	of	NULL	BC
Template	(Point)	[Letter]	is	(the)	midpoint	of	segment	[[Letter]]

JavaSketchpad

JavaSketchpad (JSP, http://www.keypress.com/sketchpad/java_gsp/) is a computer program with which authors publish dynamic geometry figures as a Java applet embedded in an HTML file so that users can interact with the figures with a web browser. The Geometer's Sketchpad (GSP), on which JSP is based, is a DGE that can run on personal computers. Instructors can publish interactive, dynamic geometry content in learning activities so that students can participate over the Internet. GSP supports the web solution by publishing an HTML document embedding a Java applet containing the JavaSketchpad script into the document between the tags <body> and </body> with ordinal label {1} to {12} (Figure 7). The Java applet displays a dynamic geometry figure with the JSP code on a browser (Figure 8).



```
<!--This file created by The Geometer's Sketchpad 4.01-->
<HTML>
<TITLE>new_task5</TITLE></HEAD><BODY>
<H4 ALIGN=CENTER><APPLET CODE="GSP.class"  ARCHIVE="jsp4.jar" CODEBASE="jsp" WIDTH=500
 HEIGHT=300 ALIGN=CENTER><PARAM NAME=MeasureInDegrees VALUE=1><PARAM NAME=DirectedAngles
 VALUE=0>
<PARAM NAME=BackRed VALUE=255>
<PARAM NAME=BackGreen VALUE=255>
<PARAM NAME=BackBlue VALUE=255>
<PARAM NAME=Construction VALUE="
{1} Point(100,40)[label('A'),red];
{2} Point(200,40)[label('B'),red];
{3} Segment(2,1)[blue];
{4} Translation(1,75,-150)[label('D'),red];
{5} Translation(2,75,-150)[label('C'),red];
{6} Segment(5,4)[blue];
{7} Segment(5,2)[blue];
{8} Segment(1,4)[blue];
{9} Midpoint(3)[label('E'),red];
{10} Midpoint(6)[label('F'),red];
{11} Segment(4,9)[blue];
{12} Segment(10,2)[blue];
">
</APPLET>
</H4>
</BODY>
</HTML>
```

Figure 7. JavaSketchpad script embedded in a HTML document.

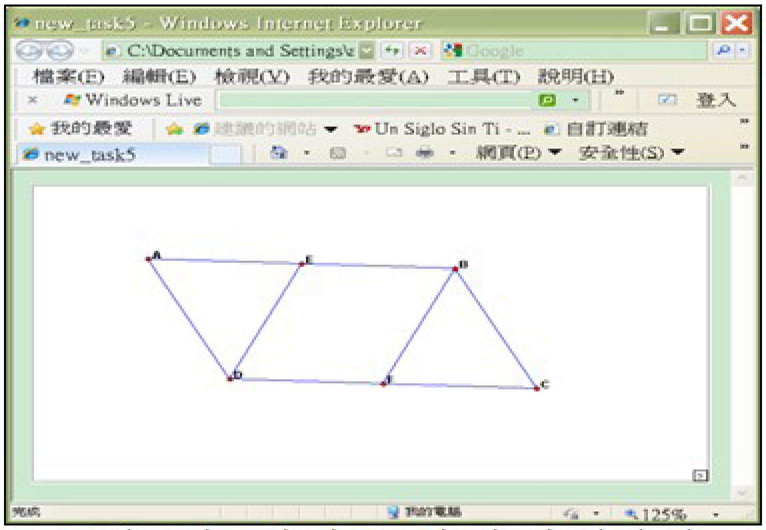

new_task5 - Windows Internet Explorer
C:\Documents and Settings\
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
Windows Live
登入
我的最愛
Un Siglo Sin Ti - ...
自訂連結
new_task5
網頁(P)
安全性(S)
A
E
B
D
F
C
我的電腦
125%

Figure 8. Figure drawn by the JavaSketchpad script in Figure 7.

Text Understanding

There are two different methods for parsing an input geometry problem text. The first method parses all sentences of a text at a time (Wong et al., 2007). This method is not flexible as it requires templates that cover all possible combinations of the sentences in the text. A better method is to parse one sentence at a time and then integrate the results for all sentences. We call this method sequential parses, in contrast to the single parse of an entire text in Wong et al. (2008). Consider the following problem: "Consider parallelogram ABCD. The point E is the midpoint of segment AB. F is the midpoint of segment CD. Prove the length of segment DE is equal to the length of segment FB."

This text is segmented into four sentences. Each sentence is matched in InfoMap and the concepts of the sentences are extracted and mapped to JavaSketchpad statements, which are then generated. One problem occurs if the referenced object is not explicitly mentioned in the preceding sentences. For example, the second sentence mentions segment AB, which is not mentioned in the first sentence explicitly. Rather, the segment AB is implied by the parallelogram ABCD mentioned in the first sentence. The next section explains how this problem is solved by the creation of geometric objects with JavaSketchpad statements.

Generation of a JavaSketchpad Script

The system needs to map the concepts of a sentence into one or more JavaSketchpad (JSP) statements to draw the concepts. The example which was described in the previous section, *parallelogram ABCD,* is mapped to the JSP statements in Figure 7. Since there is no parallelogram statement in JSP, the parallelogram is drawn by generating two points A and B, translating them with same displacement to get points C and D. Then the connection of each pair of points produces four segments AB, BC, CD, and DA. It is important to generate the segments AB and CD from the first sentence of the problem as both segments are referenced later by the second and third sentences respectively. Then, point E is created as the midpoint of segment AB in the second sentence, and point F is created as the midpoint of segment CD in the third sentence. In the last sentence of the problem, point E is referenced implicitly by the segment DE and point F is referenced implicitly by the segment FB. The final JSP script generated from the input text is shown in Figure 7 and the corresponding dynamic geometry figure is shown in Figure 8.

JavaSketchpad cannot draw some basic geometric objects directly, such as equilateral triangle, isosceles triangle, trapezoid, parallelogram, angle bisector, and arc. JavaSketchpad also cannot use some functions of GSP, such as step-by-step button, function graph, point of nonbasic geometric object, iteration, and the text area for input/output. To address this problem, we provide the scripts of some concepts with compass and straightedge constructions. These additional functions help to increase the number of problems that the system can handle. The following is an example of how to construct an angle bisector in JavaSketchpad:

```
Function Make-AngleBisector(String angle(BAC)) return String

Circle middle-Circle = Draw-JSPCircle(Get-Vertex(A));

Point intersect1(D) = JSP-Intersect(line1(AB), middle-Circle);

Point intersect2(E) = JSP-Intersect(line2(AC), middle-Circle);

Circle intersect1-Circle(X)= Draw-JSPCircle(intersect1(D), line3(AD));

Circle intersect2-Circle(Y)= Draw-JSPCircle(intersect2(E), line3(AD));

Point hiddenNode(H)= JSP-Intersect(intersect1-Circle(X), intersect2-Circle(Y));

Segment angle-bisector = JSP-Segment(Get-Vertex(A), hiddenNode(H));

return angle-bisector
```

Figure 9 shows the details of drawing an angle bisector. First, a circle is drawn with center A, intersecting AB at point D and AC at point E. Then a circle is drawn with center D and radius AD while another circle is drawn with center E and radius AE. These two circles intersect at point H as well as at A. Then AH bisects angle BAC.

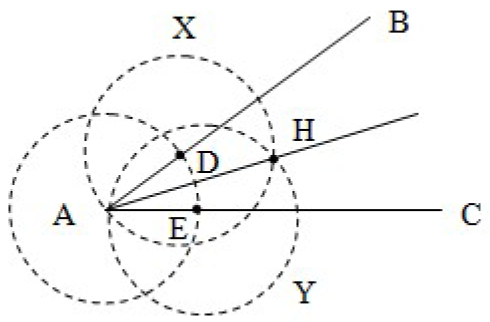


Figure 9. To construct an angle bisector by compass and straightedge.

Heuristic to Increase Correctness Rate: Reverse Understanding

A previous section shows JavaSketchpad has some limitations in constructing geometric objects. For example, the condition of two line segments equal in length or that of two angles equal in measure occurs in many geometry problems but these conditions cannot be constructed by JSP. Consider a typical problem: “Consider quadrilateral ABCD, line segments AB and CD are equal in length and so are line segments AD and BC. Prove the quadrilateral ABCD is a parallelogram.” In this example, the system will draw a quadrilateral ABCD. The system also extracts two segments AB and CD with equal length as well as two segments AD and BC. But there is no way of specifying the condition of two segments equal in length in JavaSketchpad.

In order to solve problems of this kind, we have analyzed geometry problems in junior high schools in Taiwan. About 10% of problems describe the necessary and sufficient conditions of a theorem and then ask students to prove the theorem. With a forward approach, the system will usually fail to draw the correct figure since the necessary and sufficient conditions are often complicated and the system may produce erroneous statements in JavaSketchpad. Fortunately, if the system skips the necessary and sufficient conditions and analyzes the goal condition directly, the system can draw the correct figure, resulting in a 10% increase of correctness.

This heuristic is called reverse understanding. Consider again an earlier example about a parallelogram. Figure 10 shows the process of reverse understanding. In processing the first condition of AB and CD with equal length, the system fails to find any statement to specify the condition. Then the system also fails to find any instruction to specify the second condition of AD and BC with equal length. Finally, the system finds that parallelogram ABCD is the goal to be proved. Since the system has a script to draw a parallelogram, this goal condition can be specified as a JavaSketchpad script. Moreover, this goal condition entails the first and second condition. This reverse understanding is effective in drawing the figures of some problems, despite the failure of the default forward understanding method.

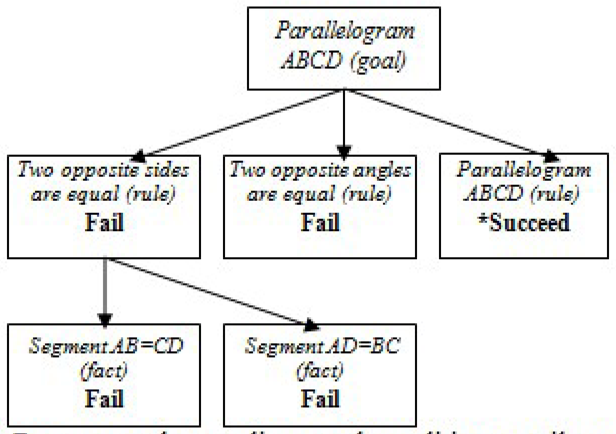

Parallelogram ABCD (goal)
Two opposite sides are equal (rule) Fail
Two opposite angles are equal (rule) Fail
Parallelogram ABCD (rule) *Succeed
Segment AB=CD (fact) Fail
Segment AD=BC (fact) Fail

Figure 10. Reverse understanding: goal condition entails earlier conditions.

Evaluation Results and Discussion

In order to evaluate the correctness rate of this drawing system, we tested the system with problems from textbooks by three publishers, namely NanYi, ChienHong, and KangHsuan. In order to keep the knowledge base within a reasonable size for this study, we chose problems involving only quadrilaterals or triangles. In geometry textbooks for junior high school, there are many geometry problems about quadrilaterals and triangles. In the selected textbooks, there were 61 geometry problems, including 34 on quadrilaterals and 27 on triangles. If a problem involving a circle is inputted to the system, the system will not understand the problem correctly because the circle concept is not included in the knowledge base. This is a restriction of concept coverage by the system.

After analyzing all the geometric problems in natural language, the system produced JavaSketchpad scripts and constructed dynamic figures as web pages. Then the correctness of the produced dynamic figures was judged manually. Table 4 summarizes the evaluation results of the system. The correctness rates were 92% and 89% for 34 quadrilateral problems and 24 triangle problems respectively. The overall correctness rate was 90%.

Table 4

Empirical Results of Correctness

Type	Quadrilaterals	Triangles	Total
Correct	31	24	55
Incorrect	3	3	6
Total	34	27	61
Rate of correctness	92%	89%	90%

Before adopting text analysis with sequential parses, prebuilt scripts, and reverse understanding, the system achieved a 77% correctness rate of figure drawing with the method of Wong et al. (2008). In this study, the proposed methods increased the correctness rate from 77% to 90%. After analyzing the geometry problems whose figures cannot be generated from the input texts, we found two reasons to account for the 10% failure rate. First, JavaSketchpad can not specify two line segments with lengths in a given ratio. Second, for texts accompanied by figures, some texts do not explicitly state conditions that are obvious in the figures. As a result, there is no way to draw correct figures by processing the texts with missing information.

Consider the input text "Given equilateral triangles ADE and ABC. Prove line segments BD and CE are equal" accompanied by the figure in Figure 11. While the figure shows that points B, C, D are collinear, this condition is not stated in the input text. Naturally, the system cannot draw a correct figure by analyzing the text alone. In order to increase the correctness rate, we can prebuild more concept scripts to make up for the limited vocabulary of JavaSketchpad. To solve the missing information problem, we must rewrite the input text by adding the missing conditions from the accompanied figure.

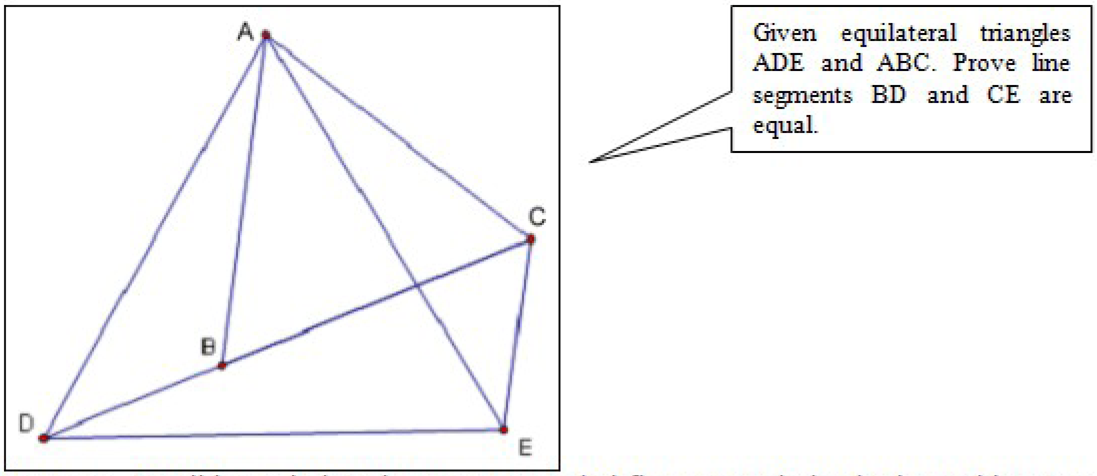

A
C
B
D
E
Given equilateral triangles ADE and ABC. Prove line segments BD and CE are equal.

Figure 11. Conditions obvious in an accompanied figure are missing in the problem text.

Conclusion

Building upon the results of previous studies (Wong et al., 2007; Mukherjee & Garain, 2008a, 2008b, 2011), we propose a methodology of converting input texts about domain concepts into a target output format such as JavaSketchpad and concept map. The methodology is illustrated by the construction of a system that automatically produces dynamic geometry figures from input geometry problems. A dynamic geometry environment such as Geometer's Sketchpad is recognized as a tool with great potential educational value. In a DGE, students can observe invariant conditions, among other changing conditions, under given premises. Unfortunately, it can be difficult for instructors and students to use tools in a DGE to construct dynamic figures. We propose to address this problem by drawing dynamic figures automatically from input problem texts. A system was built for this purpose, using a knowledge base of basic geometric concepts and a knowledge inference engine, InfoMap, to translate problem texts into JavaSketchpad scripts. A JavaSketchpad script embedded in an HTML document can be viewed by a browser on the Internet. Empirical experiments indicated that about 90% of problems from textbooks for junior high school could be analyzed correctly to produce dynamic figures. With such high accuracy, real learning activities can be designed to use the learning content generated by the system.

Mukherjee and Garain (2008a) indicate that the method of natural language understanding in Wong et al. (2007, 2008) is a mature technology that can analyze simple text problems of areas and perimeters for elementary schools. By improving the original method, this study increased the correctness rate from 77% to 90%. The most significant improvement in the new system is the approach of text analysis with integration of parsing results from sequential sentences, while the original method parsed an entire text as one very long sentence. This was possible because the text problems of areas and perimeters in elementary school were simpler than geometry proof problems, and the complicated rule templates of the knowledge base for these simpler problems could still be constructed manually. In addition, the heuristics of reverse understanding and prebuilt scripts of geometric concepts derived with compass and straightedge construction also contribute to the increase in correctness rate.

Future Work

The dynamic geometry figures generated by our system were used in two follow-up empirical studies. In one study, students were asked to make conjectures they could find in some dynamic geometry figures produced by our system (Figure 12). In another study, students were asked to prove theorems with the resources of the corresponding dynamic geometry figures (Figure 13). The empirical results of these studies would indicate the utility value of our system in generating practical and usable geometry content for educational purposes.

At the beginning of this article, we also claim that the system can support instructors who want to learn JSP in constructing dynamic geometry content. In another follow-up study, we could design a distance learning environment for learners of JSP. Activities of writing

JSP codes would be designed for these learners, who could then solve their problems by inputting their hand-picked geometry problems as natural language texts and receive corresponding JSP codes from the system. Empirical experiments are needed to evaluate the effectiveness of this approach based on a learner's initiative.

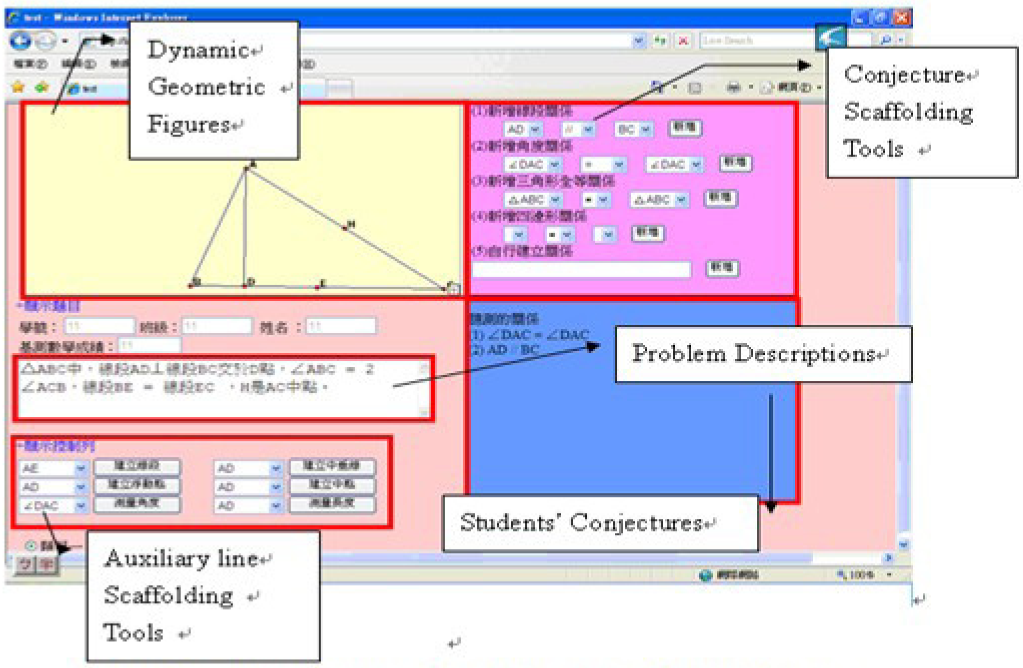

Dynamic Geometric Figures
Conjecture Scaffolding Tools
Problem Descriptions
Students' Conjectures
Auxiliary line Scaffolding Tools

Figure 12. A user interface for making geometric conjectures.

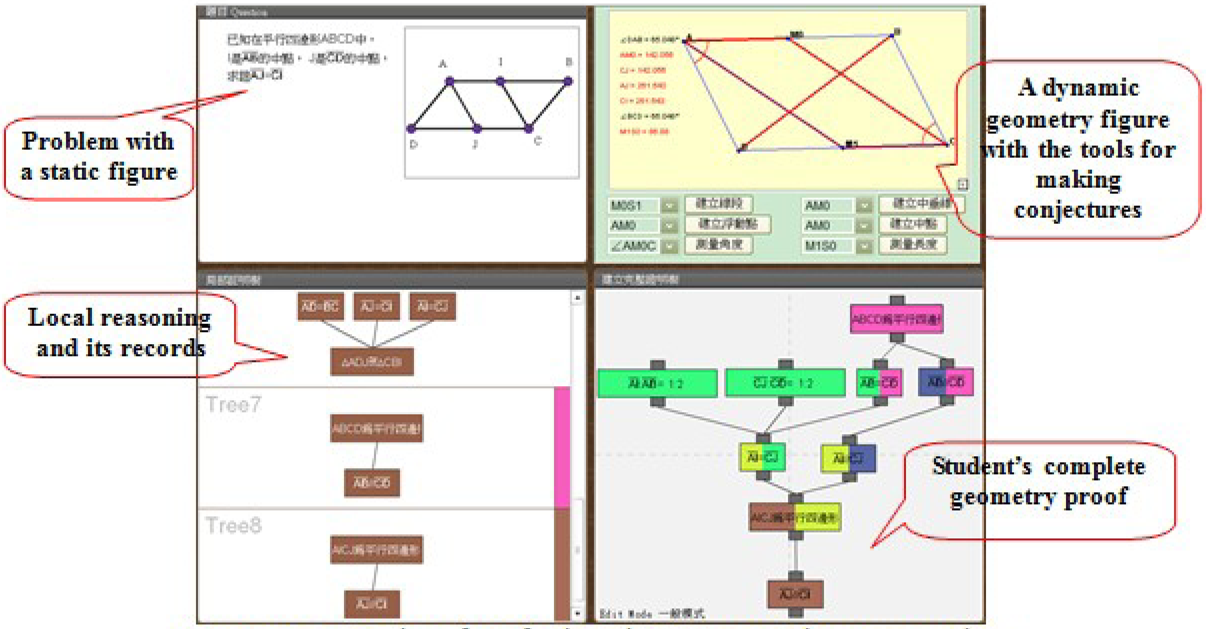

Problem with a static figure
A dynamic geometry figure with the tools for making conjectures
Local reasoning and its records
Student's complete geometry proof
Tree7
Tree8

Figure 13. A user interface for learning geometry theorem proving.

Acknowledgements

This study is supported by the National Science Council, Taiwan (NSC 98-2511-S-224-005-MY3 and NSC 98-2511-S-224-004-MY2). InfoMap is provided by the Institute of Information Science, Academia Sinica, Taiwan.

References

Chou S. C., Gao X. S., & Zhang J. Z. (1996). An introduction to Geometry Expert. *Lecture Notes in Computer Science (Proceedings of International Conference on Automated Deduction New Brunswick, CADE-13)*, *1104*, 235-239.

Christou, C., Mousoulides, N., Pittalis, M., & Pitta-Pantazi, D. (2004). Proofs through exploration in dynamic geometry environments. *Proceedings of the 28th Conference of the International Group for the Psychology of Mathematics Education*, *2*, pp. 215–222.

De Villiers, M. (1996). *Some adventures in Euclidean geometry*. University of Durban-Westville (now University of KwaZulu-Natal), South Africa.

De Villiers, M. (2003). *Rethinking proof with Geometer's Sketchpad 4*. Emeryville: Key Curriculum Press, USA.

Furinghetti, F. & Paola, D. (2003). To produce conjectures and to prove them within a dynamic geometry environment: A case study. In N.A. Pateman, B.J. Dougherty, & J. T. Zilliox (Eds.) *Proceedings of the joint meeting PME 27 and PMENA*, *2* (pp. 397-404).

Hoyles, C., & Healy, L. (1999). Linking informal argumentation with formal proof through computer-integrated teaching experiences. In Zaslavsky (Ed.) *Proceedings of the 23nd Conference of the International Group for the Psychology of Mathematics Education* (pp. 105-112). Haifa, Israel.

Hsu, W. L., Wu, S. H., & Chen, Y. S. (2001). Event identification based on the information map – INFOMAP. *Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference* (pp. 1661-1672). Tucson, Arizona, USA.

Jones, K. (2000). Providing a foundation for deductive reasoning: Students' interpretations when using dynamic geometry software and their evolving mathematical explanations. *Educational Studies in Mathematics*, *44*(1-2), 55-85.

Knuth, Eric J. (2002). Teachers'conceptions of proof in the context of secondary school mathematics. *Journal of Mathematics Teacher Education*, *5*(1), 61-88.

Koedinger, K. R. (1998). Conjecturing and argumentation in high-school geometry students. In R. Lehrer & D. Chazan (Eds.), *Designing learning environments for developing understanding of geometry and space* (pp. 319-347). Mahwah, NJ: Lawrence Erlbaum Associates.

Laborde, C., Kynigos, C., Hollebrands, K., & Strasser, R. (2006). Teaching and learning geometry with technology. In A. Gutierrez & P. Boero (Eds.), *Handbook of research* *on the psychology of mathematics education: Past, present and future* (pp. 275-304). Rotterdam, The Netherlands: Sense publishers.

Lees, B., & Cowie, J. (1996). Applying natural language technology to the learning of operating systems functions. *Proceedings of the 1st Conference on Integrating Technology into Computer Science Education (ITiCSE)* (pp. 11-13). NY, US.

Leung, A., & Lopez-Real, F. (2003). Properties of tangential and cyclic polygons: An application of circulant matrices. *International Journal of Mathematical Education in Science and Technology*, *34*(6), 859-870.

Li, P. Y., & Chen, J. D. (1988). A computer training tool using chinese natural language. *Proceedings of the 1st International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, *2* (pp. 899-904).

Lin, P. P. (2006). JavaSketchpad editor used in the design of mathematical teaching and learning aids. *Science Education Monthly (In Chinese)*, *290*, 48-57.

Lu, C. H., Wu, S. H., Tu, L. Y., & Hsu, W. L. (2005). Ontological support in modeling learners' problem solving process. *Educational Technology & Society*, *8*(4), 64-74.

Mukherjee, A., & Garain, U. (2008a). A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, *29*(2), 93-122.

Mukherjee, A., & Garain, U. (2008b). Automatic diagram drawing based on natural language text understanding. *Lecture Notes in Computer Science*, *5223/2008*, 398-400.

Mukherjee, A., & Garain, U. (2011). Intelligent tutoring of school level geometry using automatic text to diagram conversion utility. *Proceedings of the 2nd International Conference on Emerging Applications of Information Technology (EAIT)* (pp. 45-48).

National Council of Teachers of Mathematics (1989). *Curriculum and evaluation standards for school mathematics*. Reston, VA: NCTM.

National Council of Teachers of Mathematics (2000). *Principles and standards for school mathematics*. Reston, VA: NCTM.

Talmon, V. & Yerushalmy, M. (2004). Understanding dynamic behavior: Parent-Child relations in dynamic geometry environments. *Educational Studies in Mathematics*, *57*, 91-119.

Whiteley, W. (1999). The decline and rise of geometry in 20th century North America. In J.

G. McLoughlin (Ed.) *Canadian mathematics education study group* (pp. 7-30). St. John's, NF: Memorial University of Newfoundland.

Wong, W. K., Hsu, S. C., Wu, S. H., & Hsu, W. L. (2007). LIM-G: Learner-initiating instruction model based on cognitive knowledge for geometry word problem comprehension. *Computers and Education Journal*, *48*(4), 582-601.

Wong, W. K., Li, Y. H., Hsu, S. C., Yin, S. K., & Yang, H. H. (2008). An authoring tool for preparing online theorems and proofs with a dynamic geometry environment. *Proceedings of International Conference of Computers in Education* (pp.159-164). Taiwan, Taipei.