Automatic Verification of Regular Constructions in Dynamic Geometry Systems

Predrag Janičić[1]* and Pedro Quaresma[2]**

[1] Faculty of Mathematics, University of Belgrade
Studentski trg 16, 11000 Belgrade, Serbia
janicic@matf.bg.ac.yu

[2] Department of Mathematics, University of Coimbra
3001-454 Coimbra, Portugal
pedro@mat.uc.pt

Abstract. We present an application of automatic theorem proving (ATP) in the verification of constructions made with dynamic geometry software (DGS). Given a specification language for geometric constructions, we can use its processor to deal with syntactic errors. The processor can also detect semantic errors — situations when, for a given concrete set of geometrical objects, a construction is not possible. However, dynamic geometry tools do not test if, for a given set of geometrical objects, a construction is geometrically sound, i.e., if it is possible in a general case. Using ATP, we can do this last step by verifying the geometric constructions deductively. We have developed a system for the automatic verification of regular constructions (made within DGSs GCLC and Eukleides), using our ATP system, GCLCprover. This gives a real-world application of ATP in dynamic geometry tools.

1 Introduction

Dynamic geometry software (e.g., *Cinderella*, [24, 26], *Geometer's Sketchpad*, [10, 27] *Cabri*, [14, 25]) visualise geometric objects and link formal, axiomatic nature of geometry (most often — Euclidean) with its standard models (e.g., Cartesian model) and corresponding illustrations. The common experience is that dynamic geometry software significantly helps students to acquire knowledge about geometric objects and, more generally, for acquiring mathematical rigour. However, most (if not all) of these programs use only geometric concepts interpreted via concrete instances in Cartesian plane. Namely, a construction is always associated with concrete fixed points (with concrete Cartesian coordinates). In such environments, some constructions (usually by ruler and compass) are illegal (e.g., if they attempt to use the intersection of two parallel lines), but the question if

* This work was partially supported by Serbian Ministry of Science and Technology grant 144030. Also, partially supported by the programme POSC, by the Centro International de Matemática (CIM), under the programme "Research in Pairs", while visiting Coimbra University under the Coimbra Group Hospitality Scheme.

** This work was partially supported by programme POSC.

such construction is always illegal or it is illegal only for given particular fixed points is left open (if a construction is always possible, we will call it *regular*). Indeed, for answering such question, one has to use deductive reasoning, and not only a semantic check for the special case. Consider one simple example: given (by Cartesian coordinates) three fixed distinct points A, B, C, we can construct a point D as an image of point C in translation $\mathcal{T}_{AB}$; later on, if we try to construct an intersection of lines AC and BD, we will discover that there is no such intersection (since these two lines are parallel). This holds, not for some specific points A, B, C, with D determined as above, but for all triples of points A, B, C. So, in this situation, the user of a geometry tool should get the information that his/her construction is illegal, and moreover, that it is illegal not only for a given special case, but always. In this way, the deductive nature of geometric conjectures and proofs should be linked to the semantic nature of models of geometry and, also, to human intuition and to geometric visualisations.

In the rest of this paper we present our system which addresses the above problem. Our system is implemented within dynamic geometry software GCLC [11] and Eukleides [19, 23] and uses a geometry theorem prover, GCLCprover [12], based on the area method [4, 5]. Our framework, GeoThms [21, 22], is a Web tool that integrates the above components with a repository of theorems related to geometric constructions.

Closely related to our system is Gool — a geometric object-oriented language and a system for geometric computation, reasoning and visualisation [15]. This system focuses on symbolic manipulation of geometric objects (in algebraic form). Regarding handling degeneracy conditions and illegal constructions, there is a *consistency checking* system [16]. When an object is modified, or a new relation among existing object is declared, the system checks if this action is allowed, i.e., if it is consistent with the rest of the construction. This check is reduced to testing if a corresponding algebraic system has solutions in real numbers. Also, related to our system is *Geometry Explorer*, based on the full-angle method [30]. This system provides tight integration of DGS and ATP, and produces human-readable proofs of properties of constructed objects (in LaTeX form). MMP/Geometer also combines features of DGS and ATP, and uses different proving methods, including those generating synthetic, human-readable proofs [8, 9]. There are several other systems that in some degree link DGSs with ATPs: *Geometry Expert* (GEX) [7]; GEOTHER [28, 29]; *Cinderella* [13, 24, 26]; *Discover* [2]; *GeoView* [1], and *GeoGebra* [6]. However, none of these systems incorporates a verification system for constructions which provides arguments in the form of synthetic, readable proofs.

Paper overview. Section 2 briefly discusses geometric constructions, the domain of our system. Section 3 talks about parts of our framework; subsection 3.1 is about dynamic geometry software, especially GCLC and Eukleides; subsection 3.2 is about automated theorem proving in geometry and especially the prover GCLCprover and subsection 3.3 briefly describes the integration of these tools in a Web Geometric Framework. Section 4 presents the verification system and the covered critical constructions. Section 5 presents some examples. Section 6 discusses further work, and in Section 7 we draw final conclusions.

2 Geometric Constructions

For hundreds, or even thousands of years, geometric construction problems have been one of the most attractive parts of geometry and mathematics. A geometric construction is a sequence of specific, primitive construction steps. These primitive construction steps (also called *elementary constructions*) are based on using a *ruler* (or a *straightedge*[1]) and a *compass*, and they are:

– construction (with a *ruler*) of a line such that two given points belong to it;
– construction (with a *ruler*) of a segment connecting two points;
– construction of a point which is an intersection of two lines (if such a point exists);
– construction (with a *compass*) of a circle such that its centre is one given point and such that a second given point belongs to it;
– construction of intersections between a given line and a given circle (if such points exist).

By using this set of primitive constructions, one can define more complex, compound constructions (e.g., construction of a right angle, construction of the midpoint of a line segment, etc.).

The abstract (i.e., formal, axiomatic) nature of geometric objects has to be distinguished from their usual interpretations. A geometric construction is a procedure consisting of abstract steps and it is not a picture. However, for each construction there are its counterparts, its interpretations in the standard Cartesian model.

3 Component Modules of the Automatic Verification System

In this section, we present the building blocks of our automatic verification system for geometric constructions.

3.1 GCLC and Eukleides

GCLC is a tool for teaching and studying mathematics, especially geometry and geometric constructions, and also for storing descriptions of mathematical figures and for producing digital illustrations.[2] GCLC provides support for a

[1] The term “straightedge” is sometimes used instead of “ruler” in order to emphasise there are no markings which could be used to make measurements.

[2] GCLC package is freely available from `www.matf.bg.ac.yu/~janicic/gclc/`. The mirrored version is available from EMIS (The European Mathematical Information Service) `www.emis.de/misc/index.html`. There are versions of GCLC for Windows and for Linux.

range of geometric constructions and isometric transformations. In GCLC there is also support for symbolic expressions, second order curves, parametric curves, control structures, etc. GCLC is based on the idea that constructions are formal procedures, rather than drawings. Thus, in GCLC, producing mathematical illustrations is based on "describing figures" rather than of "drawing figures". All figures are described in this spirit, using the GC language. These descriptions directly reflect the mathematical contents, i.e., the meaning of mathematical objects to be presented, and are easily understandable to mathematicians. WinGCLC is the Windows version of GCLC, with a rich graphical interface and it provides a range of additional functionalities to GCLC. It supports interactive work, animations, traces, "watch window" for monitoring values of selected objects, etc. [11].

Eukleides[3] is an Euclidean geometry drawing language. There are two programs related to it. The first is `eukleides`, a processor for describing geometric figures within a (La)TeX document. It can also convert figures in Eukleides format to EPS format. The second is `xeukleides`, a GUI front-end for creating interactive geometric figures. This program can also be used for editing Eukleides code. Eukleides, like GCLC, has been designed to be close to the traditional language of elementary Euclidean geometry. We have developed a tool `euktogclcprover`, that converts Eukleides files to GCLCprover files, enabling the prover to be used with geometric constructions described within both GCLC and Eukleides.

We have developed a XML-based format (and accompanying tools) for representing geometric constructions and proofs. This format enables a suitable rendering of this contents, and also serves as a convenient exchange format between, not only GCLC, Eukleides, and GCLCprover, but other geometric tools as well.

3.2 GCLCprover, an ATP based on the Area Method

Automated theorem proving in geometry has two major lines of research: synthetic proof style and algebraic proof style (see, for instance, [17] for a survey). Algebraic proof style methods are based on reducing geometry properties to algebraic properties expressed in terms of Cartesian coordinates. These methods are usually very efficient, but the proofs they produce often do not reflect the geometric nature of the problem and, basically, they give only *yes* or *no* conclusions. Synthetic methods attempt to automate traditional geometry proof methods and to produce human-readable proofs.

The area method is a synthetic method providing traditional (not coordinate-based), human-readable proofs [4, 5]. The proofs are expressed in terms of higher-level geometric lemmas and expression simplifications. The main idea of the

[3] Eukleides is available from `http://www.eukleides.org`, There are versions for a number of languages. The second author of this paper is responsible for the Portuguese version of Eukleides: EukleidesPT is available from `http://gentzen.mat.uc.pt/~EukleidesPT/`

method is to express hypotheses of a theorem using a set of constructive statements, each of them introducing a new point, and to express a conclusion by an equality of expressions in some geometric quantities (e.g., the signed area of a triangle), without referring to Cartesian coordinates. The proof is then based on eliminating (in reverse order) the points introduced, using for that purpose a set of appropriate lemmas. After eliminating all introduced points, the current goal becomes an equality between two expressions in quantities over independent points. If it is trivially true, then the original conjecture was proved valid, if it is trivially false, then the conjecture was proved invalid, otherwise, the conjecture has been neither proved nor disproved. In all stages, different simplifications are applied to the current goal. The method does not have branching, which makes it very efficient. The area method is applicable to a wide range of constructions and a wide range of geometric conjectures. For details of the method and correctness proofs for all simplification steps see [20].

We have implemented GCLCprover, a theorem prover that allows formal deductive reasoning about objects constructed with the help of DGSs. The prover is based on the area method. It produces proofs that are human-readable and with an explicit justification for every proof step. The prover can be used in conjunction with other dynamic geometry tools. Apart from the original implementation by its authors [4, 5], we are aware of another two geometry provers based on the area method: one within the Theorema project [3], and one within the system Coq (COQareaMethod) [18].

GCLCprover is tightly integrated with dynamic geometry tools (GCLC and Eukleides). This means that one can use the prover to reason about a a DGS construction (i.e., about objects introduced in it), without changing and adapting it for the deduction process — the user only needs to add the conclusion he/she wants to prove. The geometric constructions made within the DGSs are internally transformed into primitive constructions of the area method, and in some cases, some auxiliary points are introduced.

GCLCprover can prove many complex geometric problems in milliseconds, producing readable proofs (in LaTeX or XML form).

3.3 The Geometric Framework

GeoThms[4] is a framework that links dynamic geometry software (GCLC, Eukleides), geometry theorem provers (GCLCprover), and a repository of geometry problems (geoDB). GeoThms provides a Web workbench in the field of constructive problems in Euclidean geometry. Its tight integration with dynamic geometry tools and automatic theorem provers (currently GCLC, Eukleides, GCLCprover and COQareaMethod) and its repository of theorems, figures, and proofs, give the user the possibility to easily browse through a list of geometric problems, their statements, illustrations and proofs. It also provides an interface to the DGS and ATP components, allowing the interactive use of those pro-

[4] GeoThms is accessible from `http://hilbert.mat.uc.pt/~geothms`

grams and also supporting the automatic verification of regular constructions performed within the DGSs.

4 Integrated Automated Verification System

The system for automated deductive testing whether a construction is regular, is built into the DGSs, GCLC and Eukleides, and uses GCLCprover. While processing the input file (with a description of a geometrical construction), a DGS provides to the built-in theorem prover all construction steps performed (transformed into a suitable form). This system can be switched off or on.

When the main module of GCLC encounters a construction step that cannot be performed (for instance, two identical points do not determine a line), it reports that the step is illegal with respect to a given set of fixed points (at this point, this is only an argument based on semantics, on calculations concerning concrete fixed points), and then it invokes the theorem prover. After that, the prover is ran on the critical conjecture (e.g., it tries to prove that the two points are identical) and, if successful, it reports that the construction step is always illegal/impossible. This is a result of a deduction process based on formal description of constructions, not on coordinates of the concrete points involved.

We point out that the "errors" that our deduction system detects and reports about are substantially different from syntax errors detected by the parsing modules of DGSs. Syntax errors are usually simple, local, must be eliminated from the description of the construction, and are not related to any deeper underlying geometrical knowledge. On the other hand, an illegal construction detected by our system signals the user to reconsider the whole of the construction, and claims that the construction is impossible no matter how the fixed points were selected. From a semantical point of view, we can eliminate some of the errors that our system reports about: for instance, if we use homogeneous coordinates, we could treat intersections of lines uniformly and there would be no exception for parallel lines. However, we don't want our tool to *avoid* errors within constructions, we want to explore the properties that are deeply related to the intended construction and to guide the user through the construction process. This approach reveals properties of Euclidean constructions, therefore it also has an educational role.

Realm. Our automatic verification deductive-check system currently covers the following critical constructions:

– constructing a line given two points (error if the two points are identical);
– constructing an intersection of two lines (error if the two lines are parallel);
– constructing a segment-bisector, given its two endpoints (error if the two points are identical);
– constructing an angle-bisector of the angle determined by three points A, B, C (error if A and B, or C and B are identical);
– calculating an angle determined by three points A, B, C (error if A and B, or C and B are identical);

Geometric objects that are subject to deductive verification have to be made within the DGSs using the following primitives:

- introducing a new point;
- constructing a line given two points;
- determining the intersection of two lines;
- constructing the midpoint of a segment;
- constructing the segment bisector;
- constructing the line passing through a given point, perpendicular to a given line;
- constructing the foot from a point to a given line;
- constructing the line passing through a given point, parallel to a given line;
- constructing the image of a point in a given translation;
- constructing the image of a point in a given scaling transformation;
- selecting a random point on a given line.

which are internally transformed into primitive constructions of the area method. For more details about this transformation see [20].

It is worth pointing out that although GCLC and Eukleides have support for a large number of constructions, only few of them can be illegal. The above list of critical constructions almost exhaust them. The only possible illegal constructions which are not covered by the current version of our system are constructions of intersection points of a circle and a line, and of two circles. Corresponding geometric conjectures cannot be generally handled by the area method and GCLCprover. In our future work, we will consider extending our system by additional deduction methods that can also cover this sort of constructions.

5 Worked Examples

In this section we give several examples for which our system can deductively test if they are regular. There is also one example that is out of the scope of the current version of our system.

5.1 Example 1

Consider the example discussed in Section 1: given three fixed distinct points A, B, C, let us construct a point D as an image of the point C in translation $\mathcal{T}_{AB}$; draw lines AB and CD (denoted p and q) and label all the points. The GCLCcode for this construction and the corresponding illustration, are shown in Figure 1.

If we attempt to construct a point X as the intersection of the lines p and q (by adding the command `intersec X p q` at the end of the code given in Figure 1), we will get the following message:

```
Error 14: Run-time error: Bad definition.
Can not determine intersection. (Line: 18, position: 10)
File not processed.
```

```
point A 20 10
point B 40 25
point C 70 15

translate D A B C
line p A C
line q B D

drawline A B
drawline C D
cmark_lt A
cmark_lt B
cmark_lt C
cmark_lt D
```

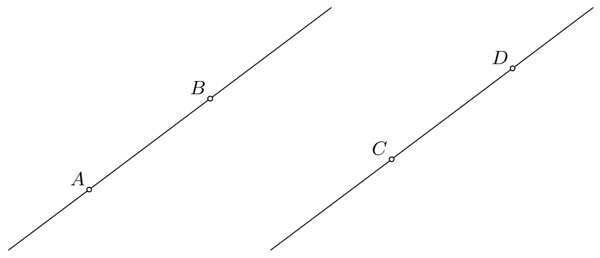

Fig. 1. GCLC code and the corresponding illustration for the example with parallel lines.

This information is semantic-based, it is true for the given particular points A, B, C, i.e., for these three particular points, the lines p and q are parallel. However, if our deductive-check system is turned on, we will also get additional, much deeper information:

```
Deduction check invoked: the property that led to the error is
tested for validity.

Total number of proof steps:          18

Time spent by the prover: 0.001 seconds
The conjecture successfully proved - the critical property always holds.
The prover output is written in the file error-proof.tex.
```

This means that it was proved that lines p and q never intersect, so this construction is always illegal. The proof of this fact is generated by the prover GCLCprover and the proof outline is given in Figure 2. Note that the condition $p \| q$ is equivalent to the condition that the areas of triangles ABD and CBD are equal.

5.2 Example 2

Consider the example given in Figure 3. This example is very similar to the previous one, the only difference is in the way point D is determined. In both cases point D gets the same Cartesian coordinates. However, in the first example, D is determined by a construction based on the points A, B, C. In contrast, in the second example, point D is determined by Cartesian coordinates, independently from the points A, B, C.

This time, it is not possible to deduce that this construction is always illegal:

(1)	$S_{ABD} = S_{CBD}$	,	by the statement
(2)	$(S_{ABC} + (1 \cdot (S_{ABB} + (-1 \cdot S_{ABA})))) = S_{CBD}$	,	by Lemma 29 (point D eliminated)
(3)	$(S_{ABC} + (1 \cdot (0 + (-1 \cdot 0)))) = S_{CBD}$	,	by geometric simplifications
(4)	$S_{ABC} = S_{CBD}$	,	by algebraic simplifications
(5)	$S_{ABC} = (S_{CBC} + (1 \cdot (S_{CBB} + (-1 \cdot S_{CBA}))))$	,	by Lemma 29 (point D eliminated)
(6)	$S_{ABC} = (0 + (1 \cdot (0 + (-1 \cdot (-1 \cdot S_{ABC})))))$	,	by geometric simplifications
(7)	$0 = 0$	,	by algebraic simplifications

Q.E.D.

Fig. 2. Proof of the critical property for example 5.1.

```
point A 20 10
point B 40 25
point C 70 15
point D 90 30

line p A C
line q B D

drawline A B
drawline C D
cmark_lt A
cmark_lt B
cmark_lt C
cmark_lt D

intersec X p q
```

Fig. 3. GCLC code for example with parallel lines and the point D given by Cartesian coordinates.

```
Run-time error: Bad definition. Can not determine intersection.
(Line: 16, position: 10)

Deduction check invoked: the property that led to the error will
be tested for validity.

The conjecture not proved - the critical property does not
always hold.
The prover output is written in the file error-proof.tex.
```

5.3 Example 3

Consider a more elaborate example (see Figure 4): let O_1 and O_2 be the pairwise intersections of the side-bisectors of triangle ABC. These two points are always identical, so the construction of a line p determined by these two points is not possible. When the system encounters this construction step, it invokes the prover which successfully proves that this step is always illegal.

```
point A 30 10
point B 70 10
point C 60 45

med a B C
med b A C
med c B A

intersec O_1 a b
intersec O_2 a c

drawsegment A B
drawsegment A C
drawsegment B C
cmark_b A
cmark_b B
cmark_t C
cmark_lb O_1
cmark_rb O_2

line p O_1 O_2
```

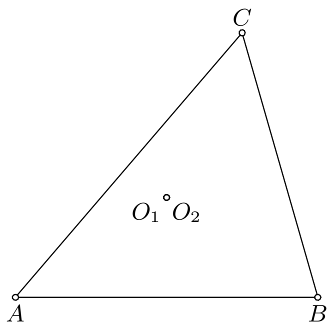

Fig. 4. GCLC code and the corresponding illustration for the example with two identical points.

5.4 Example 4

In the example shown in Figure 5, line t contains point A and is the tangent to circle k. The constructions is as follows: let k be the circle with centre O passing through the point X; let M be the midpoint of the segment OA; let l be the circle with centre M and passing through the point A; let T_1 and T_2 be the intersection points of the circles k and l. Since T_2 belongs to l, it holds that the angle AT_2O is a right angle. Since T_2 belongs to k, it follows that T_2 belongs to the tangent from A to k. Let us denote by t the tangent AT_2 from A to k. Since t is a tangent, its two intersection points with k, P_1 and P_2, are identical. Therefore, P_1 and P_2 do not determine a line, which is relevant for the construction step `line p P_1 P_2`. This is detected by the main construction module (for the given, specific points), but the prover fails to prove it (because of the realm of the area method, see Section 4).

```
point A 20 25
point O 60 25
point X 60 40

circle k O X
midpoint M O A
circle l M A

intersec2 T_1 T_2 k l
line t A T_2
intersec2 P_1 P_2 k t

cmark_t A
cmark_t O
cmark_lb T_2
drawcircle k
drawline t

line p P_1 P_2
```

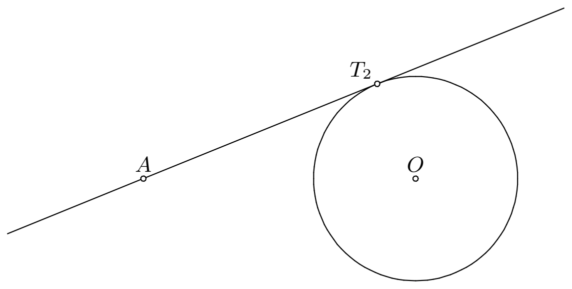

Fig. 5. GCLC code and the corresponding illustration for the example with tangent.

6 Further Work

Our verification system checks if a construction is illegal, i.e., if it is always impossible (no matter how the starting points were selected). While the construction (with its Cartesian counterpart) is being performed, the verification system is invoked, when and if, for a given set of fixed points, a construction step cannot be performed. We can extend our system so it can also invoke deduction checks for all construction steps even if there is no semantic error encountered. This would give verified regular constructions — constructions that can always be performed (provided the fixed points meet some conditions). However, such system would be time consuming (as it would run the theorem prover for each construction step).

We are planning to further improve the underlying deducting module, and to implement other geometry theorem provers, covering constructions that are out of the realm of the current system.

Also, we are planning to develop a support for guided step-by-step constructions. Such a tutoring system would control each user step, both in syntax, semantics, and deductive terms, and would serve as a teaching assistant for studying geometry.

7 Conclusions

In this paper, we presented our system for automatic verification of constructions. It provides a deep argument why a certain construction is not regular and it gives a new power to dynamic geometry tools. The system is used within dynamic geometry tools GCLC and Eukleides, and in the wider context of our publicly available geometric framework GeoThms. The underlying deduction module is based on the area method for Euclidean geometry. For future work, we are planning to implement some other geometry prover, and to further extend the realm of our system. We are also planning to extend the system so it can be used as a tutor for studying geometry.

Acknowledgements: We are grateful to Reinhard Kahle for an inspiring discussion about topics presented in this paper.

References

1. Yves Bertot, Frédérique Guilhot, and Loïc Pottier. Visualizing geometrical statements with GeoView. *Electr. Notes Theor. Comput. Sci.*, 103:49–65, 2004.
2. Francisco Botana and José L. Valcarce. A dynamic-symbolic interface for geometric theorem discovery. *Computers and Education*, 38:21–35, 2002.
3. B. Buchberger, A. Craciun, T. Jebelean, L. Kovacs, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. Theorema: Towards computer-aided mathematical theory exploration. *Journal of Applied Logic*, 4(4):470–504, 2006.
4. Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated production of traditional proofs for constructive geometry theorems. In Moshe Vardi, editor, *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science LICS*, pages 48–56. IEEE Computer Society Press, June 1993.
5. Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated generation of readable proofs with geometric invariants, I. multiple and shortest proof generation. *Journal of Automated Reasoning*, 17:325–347, 1996.
6. Karl Fuchs and Markus Hohenwarter. Combination of dynamic geometry, algebra and calculus in the software system geogebra. In *Computer Algebra Systems and Dynamic Geometry Systems in Mathematics Teaching Conference 2004*, pages 128–133, Pecs, Hungary, 2004.
7. Xiao-Shan Gao. GEX. http://www.mmrc.iss.ac.cn/~xgao/software.html.
8. Xiao-Shan Gao and Qiang Lin. MMP/Geometer - a software package for automated geometric reasoning. *Lecture Notes in Computer Science*, 2930:44–66, 2004.
9. X.S. Gao and Q. Lin. MMP/Geometer. http://www.mmrc.iss.ac.cn/~xgao/software.html.
10. Nicholas Jackiw. *The Geometer's Sketchpad v4.0.* Emeryville: Key Curriculum Press, 2001.
11. Predrag Janičić. GCLC — a tool for constructive euclidean geometry and more than that. *Lecture Notes in Computer Science*, 4151:58–73, 2006.
12. Predrag Janičić and Pedro Quaresma. System Description: GCLCprover + GeoThms. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 145–150. Springer, 2006.

13. Ulrich Kortenkamp and Jürgen Richter-Gebert. Using automatic theorem proving to improve the usability of geometry software. In *Procedings of the Mathematical User-Interfaces Workshop 2004*, 2004.
14. J. M. Laborde and R. Strässer. Cabri-géomètre: A microworld of geometry guided discovery learning. *International reviews on mathematical education- Zentralblatt fuer didaktik der mathematik*, 90(5):171–177, 1990.
15. Tielin Liang and Dongming Wang. Towards a geometric-object-oriented language. In Hoon Hong and Dongming Wang, editors, *Automated Deduction in Geometry*, volume 3763 of *Lecture Notes in Computer Science*, pages 130–155. Springer, 2006.
16. Tielin Liang and Dongming Wang. Geometric constraint handling in Gœl. In *Automated Deduction in Geometry*, pages 66–73, 2006.
17. Noboru Matsuda and Kurt Vanlehn. Gramy: A geometry theorem prover capable of construction. *Journal of Automated Reasoning*, 32:3–33, 2004.
18. Julien Narboux. A decision procedure for geometry in Coq. *Lecture Notes in Computer Science*, 3223:225–240, 2004.
19. Christian Obrecht. Eukleides. http://www.eukleides.org/.
20. Pedro Quaresma and Predrag Janicic. Framework for constructive geometry (based on the area method). Technical Report 2006/001, Centre for Informatics and Systems of the University of Coimbra, 2006.
21. Pedro Quaresma and Predrag Janicic. GeoThms - geometry framework. Technical Report 2006/002, Centre for Informatics and Systems of the University of Coimbra, 2006.
22. Pedro Quaresma and Predrag Janičić. Integrating dynamic geometry software, deduction systems, and theorem repositories. In Jonathan M. Borwein and William M. Farmer, editors, *Mathematical Knowledge Management*, volume 4108 of *Lecture Notes in Artificial Intelligence*, pages 280–294. Springer, 2006.
23. Pedro Quaresma and Ana Pereira. Visualização de construções geométricas. *Gazeta de Matemática*, 151:38–41, Junho 2006.
24. Jürgen Richter-Gebert and Ulrich Kortenkamp. *The Interactive Geometry Software Cinderella*. Springer, 1999.
25. Cabri Web site. http://www.cabri.com.
26. Cinderella Web site. http://www.cinderella.de.
27. Geometer's Sketchpad Web site. http://www.keypress.com/sketchpad/.
28. Dongming Wang. Geother (geometry theorem prover). http://www-calfor.lip6.fr/~wang/GEOTHER/.
29. Dongming Wang. GEOTHER 1.1: Handling and proving geometric theorems automatically. *Lecture Notes in Artificial Intelligence*, 2930:194–215, 2004.
30. Sean Wilson and Jacques Fleuriot. Combining dynamic geometry, automated geometry theorem proving and diagrammatic proofs. In *Proceedings of UITP 2005*, Edinburgh, UK, April 2005.