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 & Montenegro
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 an 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 (i.e., a DGS) to deal with syntactic errors (with respect to the underlying language). 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 don't 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 an ATP, we can do this last step by verifying the geometric construction deductively. We have developed a system for the automatic verification of regular constructions, using our ATP system, GCLCprover, to automatically verify the geometric constructions made with the DGSs GCLC and Eukleides. This gives a real application of ATP in dynamic geometry tools.

1 Introduction

Dynamic geometry software (e.g., *Cinderella*, [5, 26], *Geometer's Sketchpad*, [9, 12] *Cabri*, [4, 17]) 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 help students to acquire knowledge about geometric objects and, more generally, for acquiring mathematical rigour. However, most (if not all) of these programs use only geometry concepts interpreted via concrete instances in Cartesian plane. A construction is always associated with

⋆ 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.

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 intersection of parallel lines), but the question if such construction is always illegal or it is illegal only for given particular fixed points is left open (if a construction is never illegal, i.e., if it is always possible, we will call it *regular*). Indeed, for answering such question, one has to use deductive reasoning, and not only 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 the 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 geometrical conjectures and proofs should be linked to the semantic nature of models of geometry and, also, to human intuition and to geometrical visualisations.

In the rest of this paper we discuss our system that addresses the above problem. Our system is implemented within dynamic geometry systems GCLC [13] and Eukleides [21, 23] and uses the geometry theorem prover GCLCprover [14] based on the area method [6, 7]. Our framework, GeoThms [24, 25], is a web tool that integrates the above components with a repository of theorems related to geometrical constructions.

Closely related to our system is *Geometry Explorer*, based on the full-angle method [28]. 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, 19]. There are several other system that in some degree link DGSs with ATPs: *Geometry Expert* (GEX) [11]; GEOTHER [10, 27]; *Cinderella* [5, 16, 26]; *Discover* [2]; *GeoView* [1], and *GeoGebra* [15]. However, none of these system provides a formal verification system for constructions, accompanied with arguments in the form of synthetic proofs.

Paper overview. Section 2, briefly discusses geometric constructions, the domain of our system; Section 3 talks about parts of our framework, with §3.1 about dynamic geometry software, especially GCLC and Eukleides, §3.2 about automated theorem proving in geometry and especially the prover GCLCprover, based on the area method, §3.3 the integration of all these tools in an 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 Geometry 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*[3]) 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 the second given point belongs to it;
- construction of intersections between a given line and a given circle (if such points exist).

By using the 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 have to be distinguished from their usual interpretations. A geometric construction is a procedure consisting of abstract steps and it is not a picture, but for each construction there is its counterpart, its interpretation 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 geometrical constructions.

3.1 GCLC and Eukleides

GCLC is a tool for teaching and studying mathematics, especially geometry and geometrical constructions, and also for storing descriptions of mathematical figures and producing digital illustrations.[4] GCLC provides support for a range of geometric constructions and isometric transformations. In GCLC there is also

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

[4] 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.

support for symbolic expressions, second order curves, parametric curves, while-loops, 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, in GC language. These descriptions directly reflect 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 provides a range of additional functionalities to GCLC. It supports interactive work, animations, traces, "watch window" for monitoring values of selected objects etc. [13].

Eukleides[5] is an Euclidean geometry drawing language. There are two programs related to it. The first is `eukleides`, a compiler for typesetting geometric figures within a (La)TeX document. It can also convert such figures to EPS format or to various other vector graphic formats. The second is `xeukleides`, a GUI front-end for creating interactive geometric figures. This program can also be used for editing and tuning 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 GCLC or Eukleides.

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, [18] 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 do not reflect the geometrical nature of the problem and they give only *yes* or *no* conclusion. Synthetic methods attempt to automate traditional geometry proof methods producing human-readable proofs.

The area method is a synthetic method providing traditional (not coordinate-based), human-readable proofs. The proofs are expressed in terms of higher-level geometric lemmas and expression simplifications. The main idea of the 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., signed area of a triangle), without referring to Cartesian coordinates. The proof is then based on eliminating (in reverse order) the points introduced before, 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

[5] 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/`

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 any 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, correctness proofs for all simplification steps see [22].

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 a 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 [6, 7], we are aware of another two geometry prover based on the area method: one within the Theorema project [3], and one within the system Coq [20].

GCLCprover is tightly integrated with geometry tools such as GCLC. This means that one can use the prover to reason about a GCLC 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 GCLC 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[6] 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 (GCLC, Eukleides, and GCLCprover, for the moment) and its repository of theorems, figures and proofs, give the user the possibility to easily browse through the 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 programs and also supporting the automatic verification of regular constructions done with the DGSs.

4 Integrated Automated Verification System

The system for automated testing whether a construction is regular or illegal is built into the tool GCLC and uses GCLCprover.

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

If, within a description of a construction, there is a conjecture to be proved, GCLC automatically invokes the prover GCLCprover. A mechanism for automated deductive testing whether a construction is regular or illegal can be switched off or on (by using appropriate parameter when calling GCLC).

If the mechanism is turned on, while processing the input file (with a description of a geometrical construction), GCLC provides to its built-in theorem prover all construction steps performed (transformed into a suitable form). When the main module of GCLC encounters a construction step that cannot be performed (e.g., two identical points do not determine a line), it reports that the step is illegal with respect to a given set of fixed points, and then it invokes the theorem prover. The prover is run on the critical conjecture (e.g., it tries to prove that two points are identical) and, if successful, it reports that the construction step is always illegal/impossible.

In GeoThms, the automated verification mechanism is incorporated in the "Interaction with the Drawer" section. Whenever the user uses the DGS, the appropriate option is used and, if a deductive error occurs, an error message is shown, along with a link to a PDF file with the proof of the conjecture (automatically generated by the GCLCprover).

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

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

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

- introducing points;
- constructing lines by 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;

– taking a random point on a given line.

which are internally transformed into primitive constructions of the area method. For more details see [22].

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 that are not covered by the current version of our system are constructions of intersection points of circle and line, and of two circles. Corresponding geometry 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 cover also these sorts of constructions.

5 Worked Examples

In this section we give several examples for which our system can deductively confirm that are not 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 GCLC code for this construction is given in Figure 1. Figure 2 shows the illustration produced by GCLC.

```
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 for example with parallel lines

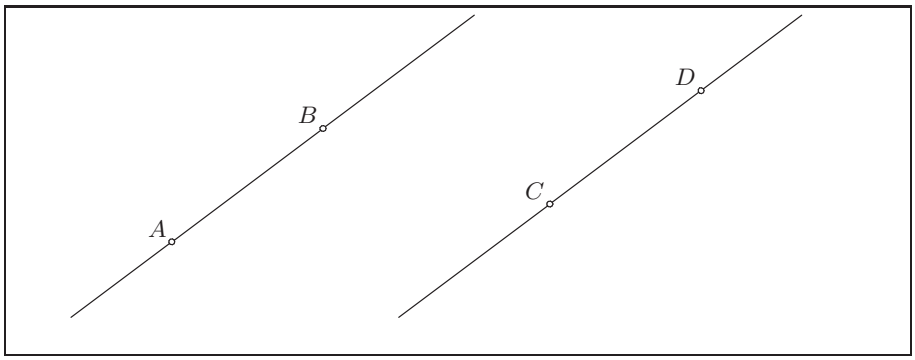

Fig. 2. Figure generated by GCLC for example with parallel lines

If we attempt to construct a point X as intersection of 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.
```

This information is semantic-based, it is true for the given particular points A, B, C, i.e., for these three particular points, lines p and q are parallel. However, if our deductive-check system is turned on (i.e., if the GCLC program was invoked with the option for deductive check), 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 are always parallel, 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 3. Note that the condition $p \| q$ is equivalent to the condition that areas of triangles ABD and CBD are equal.

5.2 Example 2

Consider the example given in Figure 4. This example is very similar to the previous one, the only difference is in the way the point D is determined. In

(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. 3. Proof of critical property for example 5.1

both cases the 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, the point D is determined by Cartesian coordinates, independently from the points A, B, C.

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

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

```
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 involved example (see Figure 5 and Figure 6): let O_1 and O_2 are 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 encounter this construction step, the system 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. 5. GCLC code for example with two identical points

5.4 Example 4

In the example shown in Figure 7 (with illustration shown in Figure 8), line t contains point A and is tangent to the circle k. The constructions is as follows: let

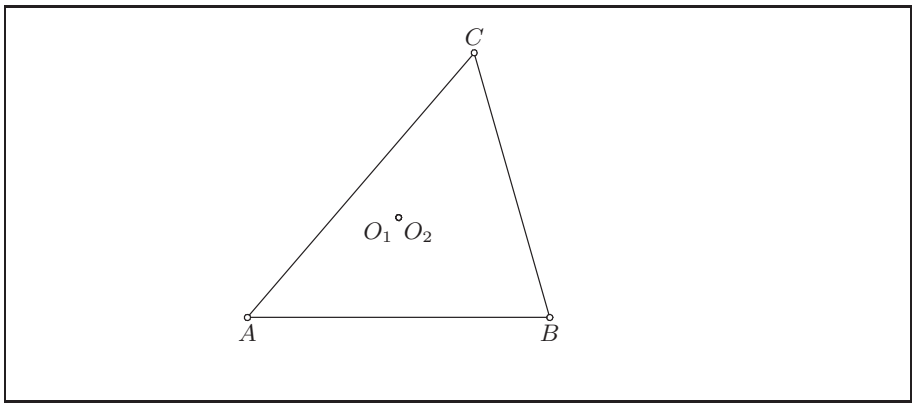

Fig. 6. Illustration for the regular part of the construction given in Figure 5.

k is the circle with centre O passing through the point X; let M be the midpoint of the segment OA; let l is the circle with centre M and passing through the point A; let T_1 and T_2 are the intersection points of the circles k and l. Since, T_2 belongs to l, it holds that the angle AT_2O is 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 intersection with k are two identical points P_1 and P_2. 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).

6 Further Work

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 step of the user, both in syntax, semantics, and deductive terms, and would serve as a teaching assistant for students 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 value to dynamic geometry tools. The system is used within dynamic geometry tools GCLC and Eukleides, in a wider context of our publicly available geometrical framework GeoThms. The underlying deduction module is based on the area method for Euclidean geometry. For future work, we are

```
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. 7. GCLC code for example with tangent

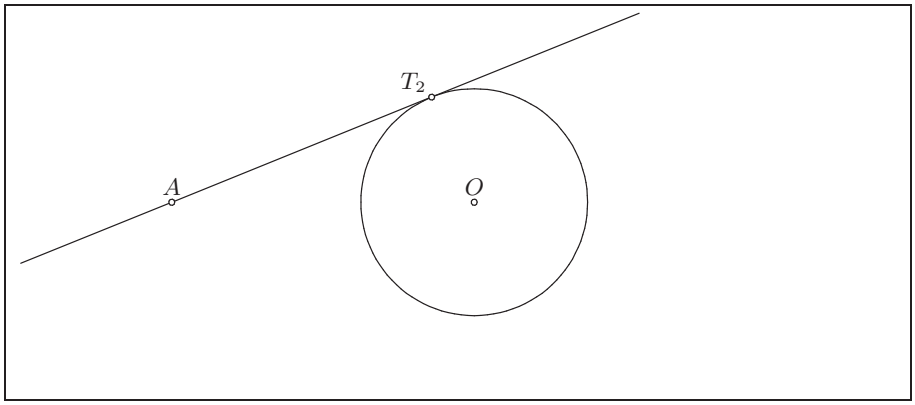

Fig. 8. Illustration for the regular part of the construction given in Figure 7.

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 students studying geometry.

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

References

1. Yves Bertot, Frédéric Guilhot, and Loïc Pottier. Visualizing geometrical statements with GeoView, 2004. `http://www-sop.inria.fr/lemme/geoview/geoview.pdf`.
2. Francisco Botana and José L. Valcarce. A dynamic-symbolic interface for geometric theorem discovery. *Computers and Education*, 38:21–35, 2002.
3. Bruno et.al. Buchberger. Theorema: Towards computer-aided mathematical theory exploration. *Journal of Applied Logic*, 2006.
4. Cabri site. http://www.cabri.com.
5. Cinderella site. http://www.cinderella.de.
6. 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.
7. 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.
8. Xiao-Shan Gao and Qiang Lin. MMP/Geometer A Software Package for Automated Geometric Reasoning. In Franz Winkler, editor, *Automated Deduction in Geometry: 4th International Workshop, (ADG 2002)*, Lecture Notes in Computer Science, 2930, pages 44–66, Springer-Verlag, 2004.
9. Geometer's Sketchpad site. http://www.keypress.com/sketchpad/.
10. GEOTHER site, http://www-calfor.lip6.fr/ wang/GEOTHER/
11. GEX site. http://www.mmrc.iss.ac.cn/gex/ and http://woody.cs.wichita.edu/gex/7-10/gex.html.
12. Nicholas Jackiw. *The Geometer's Sketchpad v4.0.* Emeryville: Key Curriculum Press, 2001.
13. Predrag Janičić GCLC – A Tool for Constructive Euclidean Geometry and More than That. In Nobuki Takayama, Andres Iglesias, and Jaime Gutierrez, editors, Proceedings of International Congress of Mathematical Software (ICMS 2006) LNAI 4151. Springer-Verlag, 2006.
14. Predrag Janičić and Pedro Quaresma. System description: Gclcprover + geothms. In Ulrich Furbach and Natarajan Shankar, editors, *IJCAR 2006*, LNAI 4130, pages p145–150. Springer-Verlag, 2006.
15. Fuchs, Karl and Hohenwarter, Markus. Combination of Dynamic Geometry, Algebra and Calculus in the Software System GeoGebra. In *Computer Algebra Systems and Dynamic Geometry Systems in Mathematics Teaching Conference*, Pécs, Hungary, 2004.
16. Ulrich Kortenkamp and Jürgen Richter-Gebert. Using automatic theorem proving to improve the usability of geometry software. In *Proceedings of the Mathematical User-Interfaces Workshop 2004*, 2004.
17. J.-M. Laborde and R. Strasser. Cabri-géométre: A Microworld of Geometry for Guided Discovery Learning. *Zentrablatt Für Didactic der Matematik*, 22(5), 1990.
18. Noboru Matsuda and Kurt Vanlehn. Gramy: A geometry theorem prover capable of construction. *Journal of Automated Reasoning*, (32):3–33, 2004.
19. MMP/Geometer site. http://www.mmrc.iss.ac.cn/ xgao/software.html.
20. Julien Narboux. A decision procedure for geometry in coq. In *Proceedings TPHOLS 2004*, volume 3223 of *Lecture Notes in Computer Science*. Springer, 2004.
21. Christiam Obrecht. Eukleides. http://www.eukleides.org/.

22. Pedro Quaresma and Predrag Janičić. Framework for Constructive Geometry (based on the Area Method). Technical Report 2006/001, Centre for Informatics and Systems of the University of Coimbra, 2006.
23. Pedro Quaresma and Ana Pereira. Visualização de construções geométricas. *Gazeta de Matemática*, (151), Junho 2006.
24. Pedro Quaresma and Predrag Janičić. GeoThms - Geometry Framework. Technical Report 2006/002, Centre for Informatics and Systems of the University of Coimbra, 2006.
25. Pedro Quaresma and Predrag Janičić. Integrating dynamic geometry software, deduction systems, and theorem repositories. In J. Borwein and W. Farmer, editors, *MKM 2006*, number 4108 in LNAI, pages 280–294, Heildelberg, 2006. Springer-Verlag.
26. Jürgen Richter-Gebert and Ulrich Kortenkamp. *Cinderella - The interactive geometry software*. Springer, 1999.
27. Dongming Wang. *GEOTHER 1.1: Handling and Proving Geometric Theorems Automatically* Automated Deduction in Geometry, 2002.
28. Sean Wilson and Jacques Fleuriot. *Combining Dynamic Geometry, Automated Geometry Theorem Proving and Diagrammatic Proofs*. Workshop on User Interfaces for Theorem Provers (UITP), 2005.