Geometry Tools GCLC and WINGCLC

Predrag Janičić

janicic@matf.bg.ac.yu
Faculty of Mathematics, University of Belgrade
Studentski trg 16, 11 000 Belgrade, Serbia

Abstract. We present geometry tools GCLC and WINGCLC. They are based on a custom-built language for declarative representation of geometry figures. There are automated geometry theorem provers built in GCLC that directly link visual and semantical geometrical information with deductive properties and machine–generated proofs. So, GCLC/WINGCLC can serve as a powerful mechanized geometric assistant. The main areas of applications of GCLC/WINGCLC are in mathematical education, in publishing, in storing mathematical knowledge, and in studies of automated reasoning.

1 Introduction

Euclidean geometry and geometric constructions have important role in mathematics and in mathematical education for thousands of years. In twentieth century, there was a shift from classical, synthetic geometry in favor of algebraic geometry in university education. However, synthetic geometry still holds a very important position in lower levels of mathematical education and also, in recent years, it has been making a comeback to university education, thanks to important applications in computer-aided design, computer graphics, computer vision, robotics, etc.

There is a range of geometry software tools, covering different geometries and geometrical problems. Many of them focus on Euclidean geometry and on construction problems. These problems are very suitable for interactive work and animations, typical for *dynamic geometry software* (e.g., *Geometer's Sketchpad*, *Cabri*, *Cinderella*). In dynamic geometry software, the user can create and manipulate geometric constructions. Typically, the user starts a construction with several points, construct new objects depending on the existing ones, and then move the starting points to explore how the whole construction changes. Dynamic geometry software can help teachers to illustrate and students to explore and understand abstract concepts in geometry. In addition, dynamic geometry software can be used for producing digital mathematical illustrations. In most of these tools, the user uses a graphical user interface, tools from toolbars, and the point-and-click approach for describing geometric constructions step-by-step.

GCLC is a tool for visualizing objects and notions of geometry and other fields of mathematics. The main distinctive feature of this tool compared to other geometry tools is that GCLC uses a special purpose, custom-built language

for describing geometric constructions. Therefore, rather than using the point-and-click approach to *draw* figures, in GCLC one *describes* constructions as formal procedures made of abstract steps. From the figure descriptions, corresponding illustrations (in Cartesian model of Euclidean plane) can be generated. This approach distinguishes the abstract nature of geometrical objects from their semantics, usual models, and visualizations.

The primary focus of the first versions of GCLC was producing digital illustrations of Euclidean constructions in LaTeX form (hence the name "Geometry Constructions → LaTeX Converter"), but now it is much more than that. For instance, there is support for symbolic expressions, for parametric curves and surfaces, for drawing functions, graphs, and trees, support for flow control, etc. Libraries of GCLC procedures provide additional features, such as support for hyperbolical geometry. Complex geometry theorems can be expressed and proved by automated geometry theorem provers built into GCLC. So, now GCLC provides mathematical contents directly linked to visual representation and supported by machine–generated proofs, and can serve as a powerful mechanized geometric assistant [5]. WINGCLC is a dynamic geometry tool built on top of GCLC with a graphical user interface and a range of additional functionalities.

The tool GCLC is under constant development since 1996. The underlying language has been only the subject of extensions, so the full vertical compatibility is kept with the earliest versions. GCLC/WINGCLC programs are implemented in the C++ programming language. GCLC consists of around 1Mb or 40000 lines of code. There are versions of GCLC for Windows and for Linux. The executable versions both for Windows and Linux have less than 1Mb. A version with a graphical user-friendly interface is available only for Windows. The applications GCLC and WINGCLC are accompanied by a detailed user manual and are freely available from `http://www.matf.bg.ac.rs/~janicic/gclc` and from EMIS (The European Mathematical Information Service) servers (`http://www.emis.de/misc/index.html`). They have thousands of users worldwide and their main areas of applications are in:

- publishing, i.e., in producing digital mathematical illustrations;
- storing mathematical contents;
- mathematical education;
- studies of automated geometrical reasoning.

2 Describing Geometric Constructions in GCLC

A geometric construction is a sequence of specific, primitive construction steps. These primitive construction steps are also called *elementary constructions by ruler and compass* and they are:

- construction (by *ruler*) of a line such that two given points belong to it;
- construction of a point such that it is the intersection of two lines (if such a point exists);

– construction (by *compass*) of a circle such that its center 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).
– construction of intersections between two given circles (if such points exist).

By using this set of primitive constructions, one can define more involved, compound constructions (e.g., construction of right angle, construction of the midpoint of a segment, construction of the perpendicular bisector of a segment, etc.). In order to describe geometric constructions, it is usual to use higher level constructions as well as the primitive ones.

GCLC follows the idea of formal constructions and the need of distinguishing abstract nature of geometrical objects and their semantics and usual models. A geometric construction is a mere procedure of abstract steps and not a figure, but for each Euclidean construction, there is its counterpart in the standard Cartesian model. GCLC provides easy-to-use support for all primitive constructions, for a range of higher-level constructions, and support for visualizing these constructions. Although motivated by the geometric constructions by ruler and compass, GCLC provides a support for some non-constructible objects too — for instance, in GCLC it is possible to determine/use a point obtained by rotation for 1°, although it is not possible to construct that point by ruler and compass).

The syntax of the language of GCLC is very simple and intuitive [7]. It is a high-level language designed for mathematicians and not a machine-oriented script language (which are used internally in some geometry tools). The descriptions of mathematical objects are easily comprehensible to mathematicians, and in the same time, the commands enable describing very complex objects in a very few lines.

In order to reduce syntactic overhead and to improve simplicity and readability, the language of GCLC is format-free, there are no command separators/terminators, arguments of commands are separated by white spaces, and the use of brackets is very limited. The language is dynamically typed, i.e., variables are not declared and can change their types during program execution. There is support for arrays and there are flow control structures *if-then-else* and *while*-loop, sufficient for the language to be computationally complete (in a sense in which, for instance, the languages C or Pascal are computationally complete). There is support for user-defined procedures and parameters are always passed by reference (so one procedure can return several results), unless they are numerical constants. Other files (for instance, containing libraries of some procedures) can be included.

There are around 150 elementary commands, but they are intuitive, so fundamentals of the language can be acquired in a very short time.

Some commands of GCLC are aimed at describing a *contents* (geometrical or other mathematical objects), while some are aimed at describing a *presentation* (i.e., visualization of the described objects). According to their semantics, the commands can be divided into the following groups:

Basic definitions: commands for introducing points, for defining a line on the basis of two selected points, defining a circle, a numerical constant, etc.

Basic constructions: constructions of intersection points for two lines, for a line and a circle, construction of the midpoint of a given segment, the bisector of an angle, the perpendicular bisector of a segment, the line passing through a given point and perpendicular to a given line, the line passing through a given point and parallel to a given line, etc.

Transformations: commands for translation, rotation, line-symmetry, half-turn, and also for some non-isometric transformations like scaling, circle inversion, etc.

Calculations, expressions, and loops: commands for calculating angles determined by triples of points, distances between points, for generating (pseudo)random numbers, for calculating symbolic expressions, support for *if-then-else* structures and *while*-loops, etc.

Drawing commands: commands for drawing (in various modes) lines, line segments, circles, arcs, ellipses, etc.

Labelling and printing commands: commands for labelling and marking points, and for printing text in various ways;

Cartesian commands: commands for direct access to a user–defined Cartesian system. A user can define a system, its unit and, within the system, he/she can define points, lines, conics, tangents, curves given in parametric form, etc. Similar support is available for 3D Cartesian space.

Low level commands: commands for changing line thickness, color, clipping area, figure dimensions, etc.

Commands for describing animations: commands for specifying animations (within WINGCLC). Several points can move from one position to another; points can also be traced (i.e., a loci can be specified).

Commands for the geometry theorem proving: commands for specifying a geometrical conjecture, controlling a level of proof details, and a maximal number of proof steps or a time limit.

The example given in Fig. 1 illustrates one simple geometric construction. Groups of commands are explained by comments (marked by the symbol `%`) within the description itself. As in most geometry tools, descriptions of constructions in GCLC typically start with introducing several (usually very few) *free points* that do not depend on other points. After introducing free points, there is typically a part with construction steps (e.g., constructing new points and lines based on existing objects) and a part with labelling and drawing commands. While an Euclidean construction is abstract procedure, there is its counterpart in the standard Cartesian model that can be visualized. In order to visualize a described construction, its free points should be assigned concrete Cartesian plane coordinates, as illustrated by the example. By changing one of the three free points, the whole of the illustration is updated.

Example given in Fig. 2 illustrates some programming features of the language of GCLC, such as *while*-loop. The construction described within this example illustrates that, for any line segment AB, the locus of the points L such that

```
% free points
point A 10 10
point B 40 10
point C 30 40

% perpendicular bisectors of the sides
med a B C
med b A C
med c B A

% intersections of the bisectors
intersec O_1 a b
intersec O_2 a c

% labelling the points
cmark_lb A
cmark_rb B
cmark_t C
cmark_lt O_1
cmark_rt O_2

% drawing the sides of the triangle ABC
drawsegment A B
drawsegment A C
drawsegment B C

% drawing the circumcircle of the triangle
drawcircle O_1 A

% specifying a conjecture
prove { identical O_1 O_2 }
```

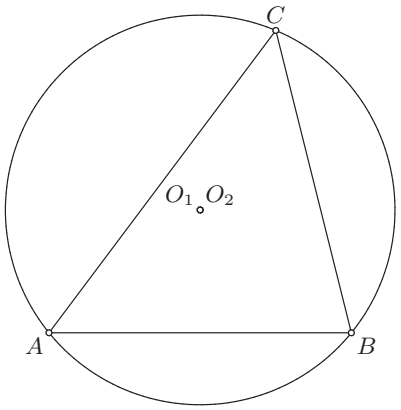

Fig. 1. Example of a description of a geometric construction (left) and the corresponding (LaTeX) output (right).

the angle ALB is right angle is the circle with the perimeter AB. The point B is rotated (giving the point B') around the point A for angle *phi* ranging from $0°$ to $70°$, and the point L is determined as the foot of the perpendicular from B to AB'. Points L for different values of *phi* are connected by line segments.

3 Automated Theorem Provers

Automated theorem proving in geometry has three major lines of research: algebraic proof, semi-algebraic proof style, and synthetic proof style that is based on artificial intelligence methods (see, for instance, [3] for a survey). Algebraic proof style methods are based on reducing geometric properties to algebraic properties expressed in terms of Cartesian coordinates. These methods are usually very

```
point A 5 5
point B 50 5

cmark_b A
cmark_b B

drawsegment A B
set_equal L_old B

number phi 0
while { phi<=70 }
{
       rotate B' A phi B
       line a B' A
       foot L B a

       drawsegment L L_old
       set_equal L_old L

       expression phi { phi+1 }
}

cmark_lt B'
drawdashsegment A B'
drawdashsegment B L
```

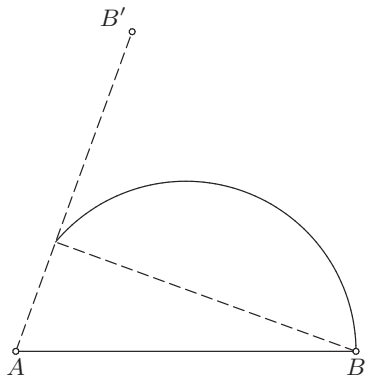

Fig. 2. Example of a construction description involving while-loop.

efficient, but the proofs they produce do not reflect the geometric nature of the problem. Synthetic methods attempt to automate traditional geometry proof methods and semi-algebraic methods are based on some sorts of calculations (e.g., over areas of triangles or over measures of angles), but still produce readable proofs. The system GCLC has three geometry theorem provers for Euclidean constructive theorems built in:

- a theorem prover based on the area method [6]. This method belongs to the group of semi-algebraic methods [1, 2]. It produces readable proofs, with justifications for each proof step.
- theorem provers based on the Gröbner bases method and on the Wu's method [11]. These methods belong to the group of algebraic methods.

All three provers can prove hundreds of complex geometrical theorems very efficiently, usually in only milliseconds (for a selection of proved theorems, see, for instance, a repository GeoThms [13]).

The theorem provers consider only abstract specification of the conjecture and do not consider Cartesian values of the points involved (they are used only for visualization). The proofs, given in terms of the supported methods, can be

exported, along with explanations for each proof step. Proofs are also accompanied by semantical counterparts — as a check if a conjecture is valid in the specific case, determined by the given Cartesian points. By this, GCLC directly links geometrical contents, its semantics, visual information, and machine–generated proofs.

The provers are tightly integrated in GCLC. This means that one can use the prover to reason about a GCLC construction without changing and adapting the description of the construction for the deduction process — only the geometry conjecture has to be provided within the `prove` command. One geometry conjecture is given in example in Fig. 1. It states that pairwise intersections of the side bisectors, the points `O_1` and `O_2`, are identical (i.e., that three side bisectors intersect at one point — at the center of the circumcircle).

The built-in theorem provers are also used for increasing power of the basic geometry tool — by certain deductive checks. GCLC detects syntactical errors in input files. The processor can also detect semantical errors — situations when, for a given concrete set of geometrical objects, a construction step is not possible. For instance, in the construction shown in Fig.1, it is impossible to construct a line O_1O_2, since the points O_1 and O_2 are identical. In such situations — when GCLC encounters a construction step that is semantically invalid (e.g., two identical points do not determine a line), it reports that the step is illegal with respect to a given set of free points. Then, it automatically invokes the theorem prover to check if a construction step is geometrically sound, i.e., if it is possible in general case. The prover is ran 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 [8].

4 Graphical Interface

WINGCLC is a Microsoft Windows application that provides graphical user interface to GCLC and a range of interactive functionalities and tools typical for dynamic geometry software (see Fig. 3) [9]. In WINGCLC, GCLC programs are edited in an integrated syntax coloring editor and are visualized within an internal viewer. The detected errors (syntactical and semantical) and reports by the theorem provers are listed in a debug/log window. Through the graphical interface, the user can choose among the available automated theorem provers (if any), among the available export formats (a simple LaTeX format, a LaTeX format based on the package `pstricks`, a LaTeX format based on the package `TikZ`, EPS (Encapsulated PostScript), and SVG (Scalable Vector Format)), can import from one of available import formats (e.g., the format of the tool JavaView (`http://www.javaview.de/`)), etc.

An animation in WINGCLC is defined (within GCLC code) as a formal construction with a set of free points that (simultaneously) move linearly from an initial to a destination position. Then, some points used in the construction can be *traced* (giving a *locus*), drawn in a selected mode. The points to be traced

Fig. 3. Screenshot of the WINGCLC application — the menus and toolbars (top), the syntax coloring editor (left), the debug/log window (left-bottom), the picture window (right), the status bar (bottom), the watch-window (right-top), the trace window (right-bottom).

(and the properties of the drawn loci) can be given either within the code, or through the *trace window*.

The *watch window* lists types and Cartesian values of selected objects in the current construction or in a particular animation frame. It can serve as a *geometric calculator* as it can be used for exploring geometry properties and conjectures. For instance, consider the example given in Fig. 1. It states that pairwise intersections of the side bisectors, the points `O_1` and `O_2`, are identical (i.e., that the side bisectors intersect at one point — at the center of the circumcircle). This property can be, in a sense, explored within WINGCLC. Namely, a value `d` can be defined to be the distance between `O_1` and `O_2`, and one can monitor the value of `d` to test that if it equals zero (for these and for any other three particular vertices). Of course, this is not a proof of the conjecture but a hint that it might be valid. For proper, deductive proofs, the built-in theorem provers are used (see Sec. 3).

For a described and visualized construction, the free points cannot be dragged (as in many other dynamic geometry tools). Instead, the user can select a free point (free points can be appropriately marked in the picture window) and its new position, and then can choose either to update a construction by this change or to define an animation. This approach is taken as in GCLC one can describe extremely complex constructions (much more complex than constructions described in the point-and-click manner) and it might be difficult to update in run-time both the text description and the picture.

One of the key ideas in development of WINGCLC was to keep the graphical interface as simple as possible and free of toolbars and tools for all construction steps and available commands. Although developing a point-and-click mode is

still an option, it is not likely that further development will go in that direction. Namely, the main comparative strength of GCLC/WINGCLC is the powerful, expressive, custom-built language. A move to the point-and-click approach would make just a yet another geometry tool in that family, typically not-suitable for describing complex geometrical configurations. Also, the largest part of the language of GCLC would remain unaccessible through the graphical interface (since there are around 150 commands) and there would still be a need for using scripting. In addition, there seem to be many users that prefer explicit textual descriptions to the point-and-click approach.

5 Areas of Applications

There are four main areas of application of GCLC:

Producing digital illustrations GCLC can serve as a tool for making high-quality digital mathematical illustrations in various formats. It has been used for producing digital illustrations for a number of mathematical books and journal articles.

Storing mathematical contents A lot of geometrical contents, both in education and in research, is of visual nature. A complex geometric construction may be illustrated by an image and can make understanding of the geometry text easier. However, without a given context and textual explanations of the constructions, it is unlikely that one can guess the correct specification of the construction. In addition, the author or a reader of a geometrical text, may need to alter an image, to modify some of its characteristics, to make it more general or more specific, and also to store it in a way that enables these sorts of transformations. Therefore, it is much preferable to have formal figure descriptions, rather than illustrations themselves. Geometrical contents stored in this way is easy to understand, visualize, modify, and process in different ways. Following this motivation, GCLC can serve as a mean for storing geometrical contents of visual nature in textual form [12].

Mathematical education In mathematical education, students can interactively use GCLC to make attempts in making constructions and/or exploring some geometrical objects, notions, ideas, problems, proofs, properties, etc. [4]. Rigorously describing geometrical objects is similar to programming, so construction problems can help students skilled in programming to understand geometry, while they can also help students acquainted with geometry and construction problems to understand programming. Visualizations and interactive work make teaching and studying geometry more interesting and more fruitful. The built-in theorem provers can help students link semantical and deductive aspects of geometry. GCLC is taught in a number of high-school and university courses on geometry and on technical writing. Due to the abstraction level required for describing constructions, GCLC is not best suited for use in primary schools.

Studies in automated geometrical reasoning The language of GCLC is expressible enough to cover a wide range of theorems in Euclidean plane geometry. GCLC has three powerful automated theorem provers built-in. Despite the fact that these theorem provers are probably the three most successful methods for automated theorem proving in Euclidean geometry, there are just a very few implementations. Given that the implementation of these methods within GCLC share many mechanisms and portions of code, GCLC can serve as a workbench for testing, comparing, and improving these methods, by combining them together or with some other methods.

6 Related Work

The tools GCLC/WINGCLC are related to the family of interactive geometry tool, but only the two (commercial) tools *Cabri* and *Geometer's Sketchpad* have history of continuous development longer than GCLC. Some of the related tools are: *Cabri* (`http://www.cabri.com/`), *Geometer's Sketchpad* (`http://www.dynamicgeometry.com/`), *GeoGebra* (`http://www.geogebra.org`), *DrGeo II* (`http://wiki.laptop.org/go/DrGeo`), *Kig* (`http://edu.kde.org/kig/`), *C.a.R* (`http://mathsrv.ku-eichstaett.de/MGF/homes/grothman/java/zirkel/doc_en/`), *KSEG* (`http://www.mit.edu/~ibaran/kseg.html`). The main distinctive feature of GCLC compared to these tools is the underlying programming language for describing geometric constructions. Some of the above tools enable recording construction steps and repeating them later on other given geometrical objects. Such recorded sequences of steps — macros (sometimes referred to as "scripts") are stored as files that are not editable and hence do not provide features of programming language. For accessing constructions and properties of constructed objects, some of the listed geometry tools enable scripting in general purpose languages (such as SmallTalk or Python). In this regard, GCLC is closely related to tools *Cinderella.2* (`http://doc.cinderella.de`) and `eukleides` (`http://www.eukleides.org`) that use special-purpose, custom-developed languages *CindyScript* and *Eukleides*.

Concerning automated theorem proving, there are several tools with some support in that respect. *Cinderella* uses a probability method for checking whether a conjecture is likely a theorem [10]. The tools *GEOTHER*(`http://www-calfor.lip6.fr/~wang/GEOTHER/`) *MMP/Geometer* (`http://www.mmrc.iss.ac.cn/~xgao/software.html`) *Geometry Explorer* [14]. *GeoProof* (`http://home.gna.org/geoproof/`) *Java Geometry Expert* (`http://woody.cs.wichita.edu/`) have built-in theorem provers, primarily based on algebraic methods. Some of these tools can automatically produces dynamic diagrams, based on the textual description of a conjecture or even visual dynamic presentation of proofs. In addition, some of the listed tools provide support for discovering geometry theorems. All of the above tools with deductive mechanisms (all the listed tools except *Cinderella*) were built with focus on automated theorem proving and available support for constructions and visualization typically directly reflect only their proving capabilities.

7 Conclusions

In this paper we presented the tools GCLC/WINGCLC for visualizing geometrical objects and notions, for teaching/studying geometry, and for producing geometry illustrations of high quality. GCLC uses the custom-built language for declarative representation of constructions, suitable for storing geometrical con-tents of visual nature. With such representation of information, the intended message and meaning of geometrical illustrations is possible to preserve and reconstruct.

After years of development, GCLC is much more than a geometry tool. There is support for symbolic expressions, for drawing parametric curves, for flow control structures, and WINGCLC makes GCLC an interactive, dynamic mathematical tool with a range of functionalities. The built-in geometry theorem provers can automatically prove a range of complex theorems. They directly link mathematical contents and its visualization to deductive information and machine–generated proofs. Thanks to that, GCLC can serve as a powerful mechanized geometry assistant. It fits into a wider context of emerging *intelligent geometrical software* that includes dynamic geometry tools, geometry theorem provers, repositories of geometry theorems, tools for visualization of geometry proofs, e-tutoring systems for geometry, etc.

References

1. C. C. Chou, OU X. S. Gao, and J. Z. Zhang. Automated production of traditional proofs for constructive geometry theorems. In *Eighth Annual IEEE Symposium on Logic in Computer Science*, 1993.
2. S.C. Chou, X.S. Gao, and J.Z. Zhang. *Machine Proofs in Geometry*. World Scientific, Singapore, 1994.
3. Shang-Ching Chou and Xiao-Shan Gao. Automated Reasoning in Geometry. In Handbook of Automated Reasoning (eds. Alan Robinson and Andrei Voronkov), Elsevier, 2001.
4. Mirjana Djorić and Predrag Janičić. Constructions, instructions, interactions . *Teaching Mathematics and its Applications*, 23(2):69–88, 2004.
5. 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)*, volume 4151 of *Lecture Notes in Computer Science*, pages 58–73. Springer-Verlag, 2006.
6. Predrag Janičić and Pedro Quaresma. System description: Gclcprover + GeoThms. In Ulrich Furbach and Natarajan Shankar, editors, *International Joint Conference on Automated Reasoning (IJCAR-2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 145–150. Springer-Verlag, 2006.
7. Predrag Janičić. Geometry Constructions Language. *Journal of Automated Reasoning*, 2009. to appear.
8. Predrag Janičić and Pedro Quaresma. Automatic verification of regular constructions in dynamic geometry systems. In F. Botana and T. Recio (Eds.), editors, *Automated Deduction in Geometry*, volume 4869 of *Lecture Notes in Artificial Intelligence*, pages 39–51. Springer-Verlag, 2007.

9. Predrag Janičić and Ivan Trajković. WinGCLC — a Workbench for Formally Describing Figures. In *Proceedings of the 18th Spring Conference on Computer Graphics (SCCG 2003)*, pages 251–256, Budmerice, Slovakia, April, 24-26 2003. ACM Press, New York, USA.
10. Ulrich Kortenkamp and Jürgen Richter-Gebert. Using automatic theorem proving to improve the usability of geometry software. In *Workshop on Mathematical User Interfaces*, 2004.
11. Goran Predović. Automated geometry theorem proving based on Wu's and Buchberger's methods. Master's thesis, Faculty of Mathematics, University of Belgrade, 2008. supervisor: Predrag Janičić.
12. Pedro Quaresma and Predrag Janičić. Integrating dynamic geometry software, deduction systems, and theorem repositories. In J.M. Borwein and W.M. Farmer, editors, *Mathematical Knowledge Management (MKM-2006)*, volume 4108 of *Lecture Notes in Artificial Intelligence*, pages 280–294. Springer-Verlag, 2006.
13. Pedro Quaresma and Predrag Janičić. Geothms — a web system for euclidean constructive geometry. *Electronic Notes in Theoretical Computer Science*, 174(2):35–48, 2007.
14. Sean Wilson and Jacques Fleuriot. Combining dynamic geometry, automated geometry theorem proving and diagrammatic proofs. In *Workshop on User Interfaces for Theorem Provers (UITP)*, 2005.