System Description: GCLCprover + GeoThms

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`

1 Introduction

Dynamic geometry tools (e.g., *Cinderella*, *Geometer's Sketchpad*, *Cabri*, *Eukleides*[1]) visualise geometric objects, allow interactive work, and link formal, axiomatic nature of geometry (most often — Euclidean) with its standard models (e.g., Cartesian model) and corresponding illustrations. These tools are used in teaching and studying geometry, some of them also for producing digital illustrations. The common experience is that dynamic geometry tools significantly help students to acquire knowledge about geometric objects. However, despite the fact that geometry is an axiomatic theory, most (if not all) of these tools concentrate only on concrete models of some geometric constructions and not on their abstract properties — their properties in deductive terms. The user can vary some initial objects and parameters and test if some property holds in all checked cases, but this still does not mean that the given property is valid.

We have extended GCLC, a widely used dynamic geometry package,[2] with a module that allows formal, deductive reasoning about constructions made in the main, drawing module. The built-in theorem prover (GCLCprover in the following text), is based on the area method [1, 2, 6]. It produces proofs that are human-readable (in LaTeX form), and with a justification for each proof step. It is also possible, via a conversion tool, to reason about constructions made with Eukleides [7, 9]. Hence, the prover can be used in conjunction with other dynamic geometry tools, which demonstrates the flexibility of the developed deduction module. Closely linked to the mentioned tools is GeoThms — a web

* This work was 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. Also, partially supported by Serbian Ministry of Science and Technology grant 144030.

** This work was partially supported by programme POSC.

[1] See `http://www.cinderella.de`, `http://www.keypress.com/sketchpad/`, `http://www.cabri.com`, `http://www.eukleides.org`

[2] GCLC (originally a tool for producing geometrical illustrations for LaTeX, hence its name — *Geometrical Constructions → LaTeX Converter*) [3, 4] provides support for a range of geometrical constructions, isometric transformations, parametric curves, but also for symbolic expressions, and program loops. The basic idea behind GCLC is that constructions are formal procedures, rather than drawings. Thus, producing illustrations is based on "describing figures" rather than of "drawing figures".

tool that integrates dynamic geometry tools, geometry theorem provers, and a repository of geometry theorems and proofs. This integrated framework for constructive geometry, provides an environment suitable for new ways of studying and teaching geometry at different levels.

2 GCLCprover

Automated theorem proving in geometry has two major lines of research: synthetic proof style and algebraic proof style (see, for instance, [5] 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 efficient, but the proofs they produce do not reflect the geometric nature of the problem and they give only a yes/no conclusion. Synthetic methods attempt to automate traditional geometry proof methods.

The area method. This method (in the core of the prover built into GCLC) is a synthetic method providing traditional (not coordinate-based), human-readable proofs [1, 2, 6]. 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 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 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. Some steps require proving some lemmas (giving proofs on different levels).

Geometrical quantities. In our implementation of the area method, we deal with the following basic geometric quantities: *ratio of directed segments* ($\frac{\overline{AB}}{\overline{CD}}$), *signed area* ($S_{ABC}$ — signed area of a triangle ABC) and *Pythagoras difference* ($P_{ABC} = AB^2 + CB^2 - AC^2$) (for details see [8]). The conjecture is built from these geometric quantities (over points already introduced within the current construction), eventually combined together by standard arithmetic operators. A wide range of geometric conjectures can be simply stated in that way.

Properties of the area method. The procedure based on the area methods is terminating, sound, and complete: it can prove *any* geometry theorem expressed in terms of geometric quantities, and involving only points introduced by using a specific set of constructions (see below). Therefore, the procedure is a decision procedure for the described fragment of geometry. This fragment can be defined as axiomatic quantifier-free theory with the set of axioms equal to the set of all simplification and elimination rules (taken as not-oriented equalities). It can be easily shown that this theory is a sub-theory of Euclidean geometry augmented by the theory of real numbers. The method does not have any branching, which makes it very efficient for many non-trivial geometry theorems. The method can transform a conjecture given as a geometry quantity of a degree d, involving n constructed points, to a quantity not involving constructed points, and with a degree at most $5d3^{5n}$ [1], while this number is usually much less, and not reached, also thanks to the used simplification steps.

Primitive steps. Our theorem prover is a sort of rational reconstruction of the area method. The proofs are built from primitive steps: elimination steps and simplification steps. Simplifications are made explicit and based on rewrite rules. We divide simplification steps into two groups: *(i)* algebraic simplifications — apply simplification rewrite rules (not directly related to geometry, but to the properties of reals) such as: $x + 0 \rightarrow x$, $\frac{x}{y} + \frac{u}{v} \rightarrow \frac{x \cdot v + u \cdot y}{y \cdot v}$, etc; *(ii)* geometric simplifications — apply simplification rewrite rules, directly related to geometric quantities such as: $P_{AAB} \rightarrow 0$, $S_{ABC} \rightarrow S_{BCA}$. All simplifications and elimination lemmas are proved in full details in [8].

Integration. It is often the case that an application providing different functionalities is built around a theorem prover. In GCLC, we have faced the challenging problem of integrating a theorem prover into well-developed tool with well defined set of functionalities, and we have succeeded in building a system where the prover is tightly integrated. This means that one can use the prover to reason about a GCLC construction (i.e., about objects introduced in it), without adapting it for the deduction process — the user only needs to add the conclusion he/she wants to prove. GCLC and GCLCprover share (only) the parsing module, which is responsible for processing the input file and passing to GCLCprover the construction steps performed. These steps are internally transformed into primitive constructions of the area method, and in some cases, some auxiliary points are introduced. The constructions accepted by GCLCprover are: construction of a line given two points; an intersection of two lines; the midpoint of a segment; a segment bisector; a line passing through a given point, perpendicular to a given line; a foot from a point to a given line; a line passing through a given point, parallel to a given line; an image of a point in a given translation; an image of a point in a given scaling transformation; a random point on a given line.

3 GeoThms

GeoThms is a framework that links dynamic geometry tools (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 software and automatic theorem provers (GCLC, Eukleides, and GCLCprover, for the moment) and its repository of theorems, figures and proofs, gives the user the possibility to easily browse through the list of geometric problems, their statements (both in natural-language form and as GCLC/Eukleies code), illustrations and proofs, and also to interactively use the drawing and proving programs.

4 Implementation and Experiences

The GCLCprover was implemented in C++ (having around 7000 lines of code) and is very efficient. The theorem prover produces proofs in LaTeX form and a report about the proving process: whether the conjecture was proved or disproved, data about CPU time spent, and the number of proof steps performed (in several categories). At the beginning of the proof, auxiliary points are defined. For each proof step, there is a justification, and (optionally) its semantics counterpart (not used in the proof itself, but it can be used for testing conjectures). The prover can prove many complex geometric problems in milliseconds, producing short and readable proofs.[3] Results shown in Table 1 were obtained on a PC Intel Pentiun-IV, 3.2GHz, 1GB RAM. Let us consider, as a simple example, the *Midpoint's theorem*, which can be expressed and proved within GCLC. The proof produced in 0.002s is very small and readable (see Figure 1).

Theorem	elimination steps	geometric steps	algebraic steps	time (sec)
Ceva	3	6	23	0.001
Gauss line	14	51	234	0.029
Midpoint	8	19	45	0.002
Thales	6	18	34	0.001
Menelaus	5	9	39	0.002
Pappus' Hexagon	24	65	269	0.040
Areas of Parallelograms	62	152	582	0.190

Table 1. Experimental results

GeoThms is implemented in MySQL and PHP and uses LaTeX (and some other auxiliary tools) to format the output and show data in a web-page.

GCLC is available from: `http://www.matf.bg.ac.yu/~janicic/gclc/`, and GeoThms is accessible from: `http://hilbert.mat.uc.pt/~geothms`.

5 Future Work and Conclusion

The GCLCprover and GeoThms are parts of an integrated framework for constructive geometry, providing an environment suitable for studying and teaching geometry. In this system, the axiomatic nature of geometric objects is tightly linked to their standard representation (in Cartesian plane) and the formal reasoning is linked to human intuition. We believe that such a system brings new dimension in teaching and studying geometry. This system, and the GEX tool[4] (new version is currently under development) are, to our knowledge, the only dynamic geometry tools with automated deduction modules (however, unlike GCLCprover, the GEX prover implements an algebraic proof method).

We are planning to extend the prover with support for additional sets of constructions and additional heuristics, and to use the prover for run-time control of correct geometric constructions. We are also considering implementing

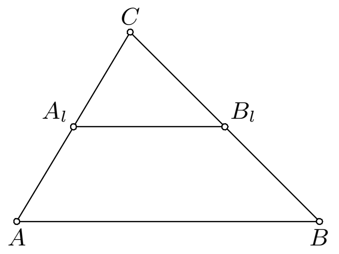
[3] Some theorems need more then 10000 steps to be proved.

[4] GEX tool: `http://woody.cs.wichita.edu/gex/7-10/gex.html`

```
point A 5 5
point B 45 5
point C 20 30
midpoint B_1 B C
midpoint A_1 A C

drawsegment A B
drawsegment A C
drawsegment B C
drawsegment A_1 B_1
cmark_b A
cmark_b B
cmark_t C
cmark_lt A_1
cmark_rt B_1

prove
 { equal
  { signed_area3 A_1 B_1 A }
  { signed_area3 A_1 B_1 B }
 }
```



$S_{A_l B_l A}$	$=$	$S_{A_l B_l B}$	by the statement (value -125=-125)	(0)
$S_{B_l A A_l}$	$=$	$S_{B_l B A_l}$	by geometric simplifications (value -125=-125)	(1)
$\left(S_{B_l AA} + \left(\frac{1}{2} \cdot (S_{B_l AC} + (-1 \cdot S_{B_l AA}))\right)\right)$	$=$	$S_{B_l B A_l}$	by Lemma 29 (point A_l eliminated) (value -125=-125)	(2)
...				
0	$=$	$\left(S_{BCB} + \left(\frac{1}{2} \cdot (S_{BCC} + (-1 \cdot S_{BCB}))\right)\right)$	by Lemma 29 (point B_l eliminated) (value 0=0)	(11)
0	$=$	$\left(0 + \left(\frac{1}{2} \cdot (0 + (-1 \cdot 0))\right)\right)$	by geometric simplifications (value 0=0)	(12)
0	$=$	0	by algebraic simplifications (value 0=0)	(13)

Q.E.D.

Fig. 1. Midpoint's Theorem: the GCLC code with the conjecture ($AB \| A_1B_1$, expressed as $S_{A_1B_1A}$=$S_{A_1B_1B}$) (left), the corresponding illustration and a part of the proof (right)

(and linking to dynamic geometry tools) other methods for automated proving of geometry theorems. Regarding GeoThms, we are planning to work on further integration of the visualisation tools and proving tools, and on further functionalities for interactive work.

References

1. Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated production of traditional proofs for constructive geometry theorems. *Proceedings LICS*, pages 48–56. IEEE Computer Society Press, June 1993.
2. Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated generation of readable proofs with geometric invariants, I. *JAR*, 17:325–347, 1996.
3. Mirjana Djorić and Predrag Janičić. Constructions, instructions, interactions. *Teaching Mathematics and its Applications*, 23(2):69–88, 2004.
4. Predrag Janičić and Ivan Trajković. Wingclc — a workbench for formally describing figures. In *Proceedings of SCCG 2003*, ACM Press, USA, 2003.
5. Noboru Matsuda and Kurt van Lehn. Gramy: A geometry theorem prover capable of construction. *JAR*, (32):3–33, 2004.
6. Julien Narboux. A decision procedure for geometry in coq. In *Proceedings TPHOLS 2004*, volume 3223 of *Lecture Notes in Computer Science*. Springer, 2004.
7. Christiam Obrecht. Eukleides. http://www.eukleides.org/.
8. Pedro Quaresma and Predrag Janičić Framework for Constructive Geometry (based on the area method). CISUC Technical Report 2006/001, 2006.
9. Pedro Quaresma and Ana Pereira. Visualização de construções geométricas. *Gazeta de Matemática*, 2006. To appear.