

## Automated Theorem Proving in GeoGebra: Current Achievements

Francisco Botana<sup>1</sup> · Markus Hohenwarter<sup>2</sup> ·  
Predrag Janičić<sup>3</sup> · Zoltán Kovács<sup>2</sup> ·  
Ivan Petrović<sup>3</sup> · Tomás Recio<sup>4</sup> ·  
Simon Weitzhofer<sup>2</sup>

Received: 20 March 2014 / Accepted: 11 March 2015 / Published online: 29 March 2015  
© Springer Science+Business Media Dordrecht 2015

**Abstract** GeoGebra is an open-source educational mathematics software tool, with millions of users worldwide. It has a number of features (integration of computer algebra, dynamic geometry, spreadsheet, etc.), primarily focused on facilitating student experiments, and not on formal reasoning. Since including automated deduction tools in GeoGebra could bring a whole new range of teaching and learning scenarios, and since automated theorem proving and discovery in geometry has reached a rather mature stage, we embarked on a project of incorporating and testing a number of different automated provers for geometry in GeoGebra. In this paper, we present the current achievements and status of this project, and discuss various relevant challenges that this project raises in the educational, mathematical and software contexts. We will describe, first, the recent and forthcoming changes demanded by our project, regarding the implementation and the user interface of GeoGebra. Then we present our vision

---

✉ Zoltán Kovács  
zoltan@geogebra.org

Francisco Botana  
fbotana@uvigo.es

Markus Hohenwarter  
markus.hohenwarter@jku.at

Predrag Janičić  
janicic@matf.bg.ac.rs

Ivan Petrović  
ivan.petrovic.matf@gmail.com

Tomás Recio  
tomas.recio@unican.es

Simon Weitzhofer  
simon@geogebra.org

of the educational scenarios that could be supported by automated reasoning features, and how teachers and students could benefit from the present work. In fact, current performance of GeoGebra, extended with automated deduction tools, is already very promising—many complex theorems can be proved in less than 1 second. Thus, we believe that many new and exciting ways of using GeoGebra in the classroom are on their way.

**Keywords** Secondary education · Interactive learning environments · Intelligent tutoring systems · Automatic theorem proving

## 1 Introduction

Proofs in mathematical education play an important role in understanding mathematics and developing student skills in problem solving and discovering facts in real life and science. One traditional way for using proofs in developing mathematical skills is to teach Euclidean geometry as a mainstream topic, since it is an “empirical theory” and one of the most well-established theories of all ([24], p. 896). For instance, by measuring the angles of triangles by a protractor we always get a sum near 180 degrees, and these empirical results yield a simple theorem.

Modern ways of teaching geometry include using dynamic geometry tools, even among teachers who stick to traditional methods. With dynamic geometry tools, the user can create and manipulate geometric constructions. Thus, the user can start a construction with several points, build new objects depending on the existing ones, and then move the starting points to explore how the whole construction changes, while keeping the established interrelations among its different components. In this way, the user can test a given or conjectured thesis, for instance, that some three points are always (for whatever positions of the initially given points) collinear, or that some two constructed lines are always (for every placement of the starting points) parallel etc. However, this is only thesis *testing* and not *proving*. Even if a thesis is affirmatively tested for hundreds of different starting points, it still does not mean that the thesis will be *always* true. The only way to show that the thesis is always true (i.e., that it is a geometry theorem) is to *prove* it. There are several methods for automated theorem proving (ATP) in geometry, but the way humans prove theorems is still very difficult to get automated.

In this paper we present our initial efforts in integrating ATP features in *GeoGebra* [26]. GeoGebra is an open-source mathematics software application for learning and teaching, with millions of users worldwide, and hence an excellent choice for showing and promoting the benefits of ATP educational scenarios. We will briefly present several theorem provers

---

<sup>1</sup> Department of Applied Mathematics I, Escola de Enxeñería Forestal, University of Vigo at Pontevedra, Campus A Xunqueira, 36005 Pontevedra, Spain

<sup>2</sup> Department of Mathematics Education, Johannes Kepler University, Altenbergerstr 69, 4040 Linz, Austria

<sup>3</sup> Department for Computer Science, Faculty of Mathematics, University of Belgrade, Studentski trg 16, 11000 Belgrade, Serbia

<sup>4</sup> Department of Mathematics, Statistics and Computation, Faculty of Sciences, University of Cantabria, Avenida de los Castros, s/n, 39071 Santander, Spain

already integrated in GeoGebra, some challenges, some examples and some possible applications in education. At the current stage, we do not aim at obtaining readable proofs from the provers. Instead, we will focus on using the new ATP features for guiding the user exploration process, since ATP would allow GeoGebra to automatically provide information on whether some user-conjectured thesis is valid or not.

In Section 2 we briefly overview the state of the art concerning the merging of dynamic geometry programs and ATP. Section 3 describes our main contribution concerning the inclusion of ATP in GeoGebra. Section 4 discusses in some detail two examples and provides performance data on a collection of examples. Section 5 focuses on educational questions. Finally, Section 6 summarizes our results and future plans.

## 2 Dynamic Geometry Software and Theorem Proving

Although Sketchpad [47] is commonly considered as the generic ancestor of current computer graphic software, the first computer program able to construct and manipulate geometric constructions in microcomputers, the *Geometric Supposer* [46], can be traced back to 1981. Over a decade after, a new generation of personal computers supported the global spread of dynamic geometry in education, exemplified by *The Geometer's Sketchpad* [28] and *Cabri Geometry* [2]. The accuracy of constructions and the visual evidence for properties provided by this piece of software were sometimes used as a replacement of proof [25, 48]. Automatic checking abilities—through the numerically approximate verification of a conjectured property in a large number of cases, yielding a highly probable claim—introduced in newer versions (of Cabri, for instance) reinforced lessening the role of proving when using these learning environments. Reacting to these techniques and exploiting theoretical developments mainly coming from academia, automated deduction techniques have recently started to enrich the field of dynamic geometry. We briefly review some dynamic geometry software (DGS) equipped with automated proving and other related features.

ATP in geometry has a history of more than fifty years [14]. Initial attempts to implement automatized theorem proving in geometry appear, in the realm of Artificial Intelligence (AI), in the 50's, when Gelernter created a theorem prover that could find solutions to a number of problems taken from high-school textbooks in plane geometry [22]. The impact of Gelernter's geometry machine led to a line of work within the AI context, on systems able to automatically build geometry proofs. An early example (from the late 70's) is *Geom*, a Prolog-based geometry theorem-prover [15]. Other systems worth mentioning are *Chypre* [3], *Cabri-Euclide* [38] or *Geometrix* [23].

On one hand, the greatest accomplishments (i.e., in terms of the complexity of the theorems to be proven) of ATP in geometry have been achieved by algebraic methods, in which the geometric statement is first translated to an algebraic counterpart and then is subject to some computer algebra manipulation. In this category we can mention the non-probabilistic, multiple checking approach which is behind the “exact check” method we will refer to in the next section [59], but the two most important methods in this group are Wu's method [9, 55] and the Gröbner bases method [8, 31]. Both of them can efficiently prove (or disprove) complicated geometry assertions; however, they output only a yes/no answer and do not provide human readable, traditional geometry proofs. Other algebraic approaches, such as the coordinate-free methods known as the area method [10, 30] and the full angle method [11, 12], also deal with complex expressions involving certain geometry quantities, but the proofs they produce are, sometimes, short and readable. There are methods,

such as the deductive database method [11], that can generate readable proofs (e.g., in terms of higher-order lemmas), but they still have a smaller scope than the algebraic provers.

On the other hand, recent interest on formal provers has led to the design of systems where proofs are verified by proof assistants such as *Coq*. For instance, *GeoView*<sup>1</sup> is a tool that combines a dynamic geometry drawing tool *GeoplanJ* with *PCoq*, a user interface for the general purpose proof assistant *Coq* ([4]). The statements of plane geometry theorems and their proofs are manually constructed and then verified within *Coq* proof assistant. Dynamic geometry figures can be automatically generated from *PCoq* theorem statements. A related program is *GeoProof*,<sup>2</sup> an interactive geometry tool that can communicate with the *Coq* proof assistant to perform interactive proofs of geometry theorems [41, 42]. Other formal systems like *E* [1] focus on diagrammatic reasoning by translating Euclid's Elements to a faithful axiomatic system which can be handled algorithmically.

Concerning algebraic provers, we can mention *Discover* [7], that combines a standard DGS with calls to some computer algebra systems (CoCoA: [16], and Mathematica: [54]) for automated discovery in Euclidean geometry. For a user-defined construction, conditions for some property to hold can be automatically discovered and then formally checked, using the Gröbner basis method. Another example of this kind is *GCLC*, a DGS with custom specification language for representing geometry constructions and geometry theorems. The program has three theorem provers built-in: provers based on the area method, Wu's method and the Gröbner bases method [29].

Let us mention a few other examples in this direction, such as *Geometry Expert*<sup>3</sup> (GEX, [13]), a DGS focused on ATP, that implements Wu's, the Gröbner basis, vector, full angle, and the area method. Another example is *Java Geometry Expert*<sup>4</sup> (JGEX, [57, 58]), under development from 2004, a new, Java version of GEX. JGEX combines dynamic geometry, automated geometry theorem proving, and, as its most distinctive part, visual dynamic presentation of proofs. It provides a series of visual effects for presentation of proofs which can be visualized either manually or automatically. Within the program distribution, there are more than six hundred examples of proofs.

*GEOTHER*<sup>5</sup> [50] is an environment that combines drawing routines and interface written in Java with five algebraic theorem provers implemented in Maple. On the bases of the textual description of a conjecture, *GEOTHER* automatically produces dynamic diagrams, i.e., assigns coordinates to the involved points in an appropriate manner. *Geometry Explorer* [53] is a dynamic geometry tool that produces human-readable proofs of properties of constructed objects, using the full-angle method. It can produce diagrammatic proof visualizations that aim to be more intuitive than textual proofs. *MMP/Geometer*<sup>6</sup> automates geometric diagram generation, geometry theorem proving, and geometry theorem discovering [21]. *MMP/Geometer* implements Wu's method, the area method, and the geometry deductive database method. Conjectures are given in a restricted pseudo-natural language or in a point-and-click manner.

As a final example, but of a different approach to proving properties, we can refer to *Cinderella* [32, 33], an interactive geometry system that uses randomized theorem

<sup>1</sup><http://www-sop.inria.fr/lemme/geoview/geoview.html>

<sup>2</sup><http://home.gna.org/geoproof/>

<sup>3</sup><http://www.mmrc.iss.ac.cn/gex/>

<sup>4</sup><http://woody.cs.wichita.edu/>

<sup>5</sup><http://www-salsa.lip6.fr/wang/GEOTHER/>

<sup>6</sup><http://www.mmrc.iss.ac.cn/xgao/software.html>

checking for analyzing constructions. It is not a symbolic, deductive theorem proving method, but a probabilistic method for checking whether a conjecture is likely a theorem.

The above systems with ATP features can efficiently prove many complex geometry theorems, but these ATP features are not primarily designed for applications in education, i.e., as a helping tool in a wider process of exploring and discovering conjectures by the students. They are, in many aspects, still only academic tools, in the prototype phase, not yet well distributed, maintained or not fully operative. This is why we would like to fill the gap by introducing a different solution to be much more useful for the school community.

### 3 Proving Capabilities in GeoGebra

GeoGebra is already a well-developed framework with a wide range of functionalities and with a stable interface familiar to millions of users. On the other hand, state of the art theorem provers also have features that are difficult to change (e.g., the precise way to introduce the conjectures to be proved, or the sort of output results, etc.). Hence, adding proving capabilities to GeoGebra, via several theorem provers integrated (see Section 3.4), poses a series of challenges, at different levels (e.g., user interface, internal representation and communication, etc.), sometimes confronted with our main goals.

#### 3.1 Goals

We have the following goals in adding proving capabilities to GeoGebra:

1. **Intuitive interface:** The user interface should remain as intuitive as possible. GeoGebra is primarily not for (deductive) proving but for experimenting, and we want to provide a simple interface for both teachers and students. Integration of proving capabilities should follow the *de facto* standards of the user community.
2. **Simplified output:** Details of a proof should not be shown to the user at this stage. In fact, as mentioned above, the most efficient proving methods do not produce readable proofs, but only a yes/no answer, following often very long internal algebraic computations, such as sequences of elimination steps. Also, even “degeneracy conditions” [9, 45] should be hidden for most users.
3. **Small size and efficiency:** Code size of the implementation matters: after introducing ATP features, GeoGebra should start not much slower than before. Also, the execution speed is important: we expect a yes/no answer for most classroom problems within a second.
4. **Usability in different GeoGebra subsystems:** Re-use of a yes/no answer may be useful for other subsystems. For example, since GeoGebra could be used for computer aided assessment (CAA) for open ended tests, a proving subsystem could enable a quick evaluation whether the student created a solution that is different from the one provided by the teacher, but is still mathematically equivalent to it.
5. **Modular architecture:** The architecture of the system should be modular in order to allow adding and using multiple methods for theorem proving. Since different methods may have different efficiency, it would be useful to provide an automatic way to select the most promising method for the given statement.

### 3.2 New GeoGebra Commands

The usual way for the user to define a statement in GeoGebra is to create a Boolean query, e.g. asking if lines  $a$  and  $b$  are parallel ( $\mathbf{a} \parallel \mathbf{b}$ ) or certain quantities are equal ( $\mathbf{x}^2 + \mathbf{y}^2 == \mathbf{z}^2$ ). Normally, GeoGebra decides whether a Boolean expression is true or not by using numerical computations. However, the new **Prove** command, that returns true/false/undefined for the given user input, uses symbolic (deductive) methods to determine whether a statement is generically true (i.e., a theorem) or not. If GeoGebra cannot determine the answer, the result is undefined.

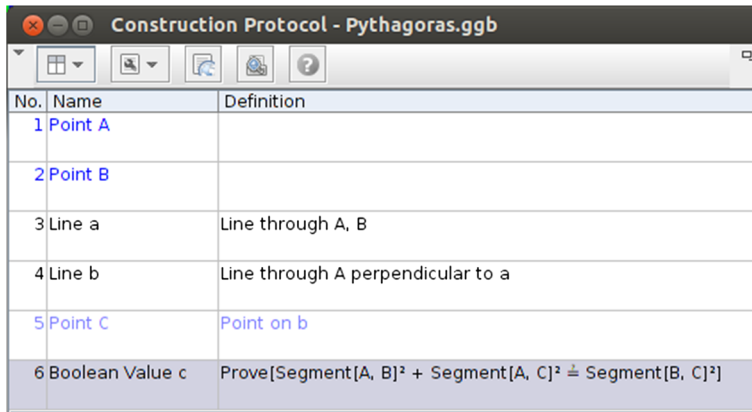
We also created the **ProveDetails** command to get more details *exactly* when the statement is true. There may usually be some minor relations which must not hold to ensure the statement to be true: for example, many Euclidean theorems for triangles are not valid if the triangle is degenerate, i.e., its third vertex lies on the opposite side (i.e., the triangle has an area of zero). For most students these fine details are usually not interesting since the teacher silently assumes some small extra conditions during the construction steps. But for a computer these details are not negligible: an automated proof in the background will classify the statement if it is always true or true only under certain conditions.

The output of the **ProveDetails** command is an empty list  $\{\}$  if GeoGebra cannot determine the answer, a list with one element: **{false}** if the statement is not a theorem (i.e., if it is not generically true), a list with one element: **{true}** if the statement is always true, or a list with the Boolean value true and another list for the degeneracy conditions, if the statement is valid in general but under certain conditions. In this last case, if all conditions in the additional list are false, then the statement is true. This means that the list of these degeneracy conditions is just a sufficient but not necessary list of assumptions, and also it cannot be guaranteed that the list is the simplest possible one.

In some cases, the **ProveDetails** command cannot translate the degeneracy conditions to human readable form. In such cases **{true, {?"..."}}** will be returned.

Both new GeoGebra commands accept a wide variety of Boolean expressions as input. On one hand, these expressions can represent a certain geometric correlation: equality, parallelism, orthogonality, collinearity, concurrency or concyclicity. Here are some examples of providing these properties:  $A \stackrel{?}{=} B$  (or **A==B** or **AreEqual[A,B]**),  $c \parallel d$  (or **AreParallel[c,d]**),  $e \perp f$  (or **ArePerpendicular[e,f]**), **AreCollinear[g,h]**, **AreConcurrent[C,D,E]**, **AreConcyclic[P,Q,R,S]**. On the other hand, the user can type a relation given by an equation (e.g.  $\mathbf{a} + \mathbf{b} == \mathbf{c}$ ). Logical operators and functions like  $\vee$ ,  $\wedge$  and negation are not supported at the moment.

It is also possible to use some specific quantities or even complicated expressions inside the input formula, like the sum of the square of distances between some given points. For classroom use, the preferred way may still be to define these quantities as previous steps in the construction itself. For example, in Fig. 1 the student proves the Pythagorean theorem by constructing the right triangle  $BAC$  and typing the proper input for the **Prove** command by simply referencing the segments of the triangle. The same result could be done by defining lengths  $\mathbf{x} = \text{Segment}[A,B]$ ,  $\mathbf{y} = \text{Segment}[A,C]$ ,  $\mathbf{z} = \text{Segment}[B,C]$  and use **Prove** $[\mathbf{x}^2 + \mathbf{y}^2 == \mathbf{z}^2]$ . Even in this simple case, it is not straightforward to decide which approach is better for the classroom use, but the teacher has the freedom to choose the better formulation for the educational situation.



No.	Name	Definition
1	Point A	
2	Point B	
3	Line a	Line through A, B
4	Line b	Line through A perpendicular to a
5	Point C	Point on b
6	Boolean Value c	Prove[Segment[A, B] <sup>2</sup> + Segment[A, C] <sup>2</sup> = Segment[B, C] <sup>2</sup> ]

**Fig. 1** Construction protocol for stating Pythagoras' theorem

In our opinion, this design is already rich enough to cover many theorems in Euclidean geometry. Also it is simple enough to help the student distinguishing between the hypotheses of the theorem (appearing as construction steps) and the thesis (the input of the **Prove** and **ProveDetails** commands).

### 3.3 Examples

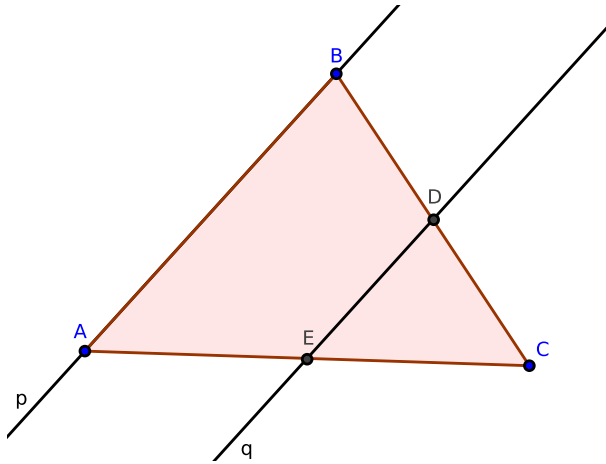
Let us suppose that we have defined three free points,  $A=(1,2)$ ,  $B=(3,4)$ ,  $C=(5,6)$ . The command **AreCollinear**[A,B,C] yields true, since a numerical check is used on the current coordinates of the points. On the other hand, **Prove**[AreCollinear[A,B,C]] will return false as an answer, since the three free points are not collinear in general, i.e., considering they are just constrained to be “free”.

Second, let us define a triangle with vertices  $A$ ,  $B$  and  $C$ , and define  $D=MidPoint[B,C]$ ,  $E=MidPoint[A,C]$ ,  $p=Line[A,B]$ ,  $q=Line[D,E]$ . Now both  $p||q$  and **Prove**[ $p||q$ ] return **true**, since a midline of a triangle will always be parallel to the appropriate side (see Fig. 2).<sup>7</sup> In addition, also **ProveDetails**[ $p||q$ ] returns {**true**} because the statement is true without any further condition for the points.<sup>8</sup>

Third, as a more complex example let us consider Pappus's hexagon theorem. Let  $A$ ,  $B$ ,  $C$  and  $D$  be free points and let us put points  $E$  on  $AB$ ,  $F$  on  $AC$ , and  $G$  on  $AD$ . Now let  $H$ ,  $I$  and  $J$  be the intersection points of  $CD$  and  $FG$ ,  $BD$  and  $EG$ ,  $BC$  and  $EF$ , respectively. Pappus claims that the points  $H$ ,  $I$  and  $J$  will be collinear. This statement is true, however, only when a set of conditions is already met: for example if  $AD$  and  $CE$  are parallel, the intersection point  $H$  cannot even be defined in the Euclidean plane (but still may be meaningful in the projective plane). GeoGebra can give a quite detailed answer on what conditions should be assumed. Namely, **ProveDetails**[AreCollinear[H,I,J]] will return

<sup>7</sup>The parallel sign must be inserted as a special character in GeoGebra by clicking first the  $\alpha$  icon on the right side of the Input Bar which opens a window, and then the correct character can be chosen—it is the 8th element in the 4th row. Another method is to select the correct Unicode character from a different application and paste it into the Input Bar in GeoGebra.

<sup>8</sup>In the future the output of **ProveDetails** command may include other pieces of information about the computation, for example, the calculation time and methods used.



**Fig. 2**  $p \parallel q$  when  $D$  and  $E$  are midpoints of  $BC$  and  $AC$

**{true, {"AreCollinear[D,E,A], AreEqual[DE,BC], AreEqual[EA,BC], AreEqual[F,A], AreParallel[DB,EF], AreParallel[FA,BC]"}}**. This means that if

- $D$ ,  $E$  and  $A$  are not collinear, *and*
- lines  $DE$  and  $BC$  are different, *and*
- lines  $EA$  and  $BC$  are different, *and*
- points  $F$  and  $A$  are different, *and*
- lines  $DB$  and  $EF$  are not parallel, *and*
- lines  $FA$  and  $BC$  are not parallel

then  $H$ ,  $I$  and  $J$  will be collinear. This set of conditions is strict in the sense that by omitting any element of it the theorem may be no longer valid.

For educational use this result (which is obtained by OpenGeoProver by using Wu's method)<sup>9</sup> is too long and unnecessarily complicated.<sup>10</sup> Luckily, there is another technique (described in [45] and now fully implemented in GeoGebra) which usually obtains a smaller list for degeneracy conditions. For Pappus's hexagon theorem the smallest possible lists are:

1. –  $A$ ,  $B$  and  $F$  are not collinear, *and*  
–  $BC$  is not perpendicular to  $AC$ .
2. –  $A$ ,  $B$  and  $C$  are not collinear, *and*  
–  $A$ ,  $B$  and  $F$  are not collinear.
3. –  $A$ ,  $B$  and  $C$  are not collinear, *and*  
–  $A$ ,  $C$  and  $D$  are not collinear.

For a student user, of course, it would be important that GeoGebra selects the “easiest” or “most beautiful” one of the possible set of conditions. In this third example the third set is the best: it contains only free variables and it is visually straightforward.

<sup>9</sup>GeoGebra must be started to achieve this output by adding `--prover=engine:OpenGeoProver, method:Wu` to the command line [49].

<sup>10</sup>It is well known that there are several mathematical and computational difficulties when defining and obtaining degeneracy conditions, such as those described by [51] in the GEOTHER system.



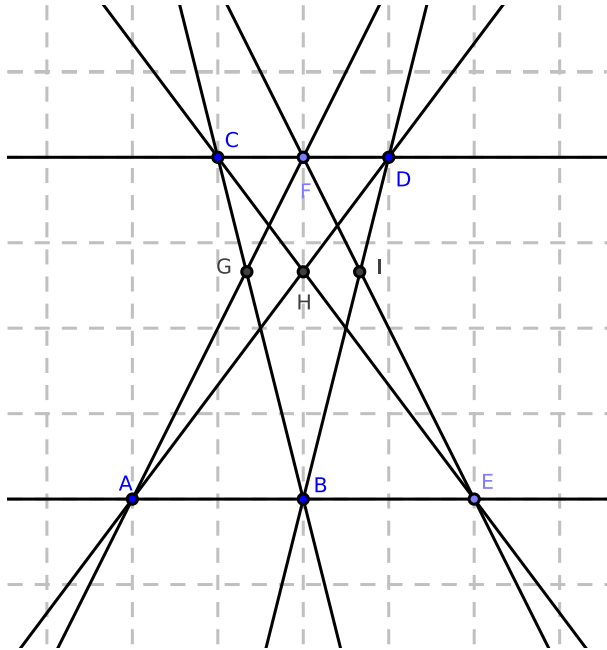


Fig. 3 Pappus's hexagon theorem realized in a good-looking configuration

From the perspective of automated proofs and programming, however, it can be difficult to make such a decision. It is easy to draw the case of the theorem shown in Fig. 3 when  $AB \parallel CD$  and  $AB \neq CD$ . Clearly, these assumptions imply both the second and

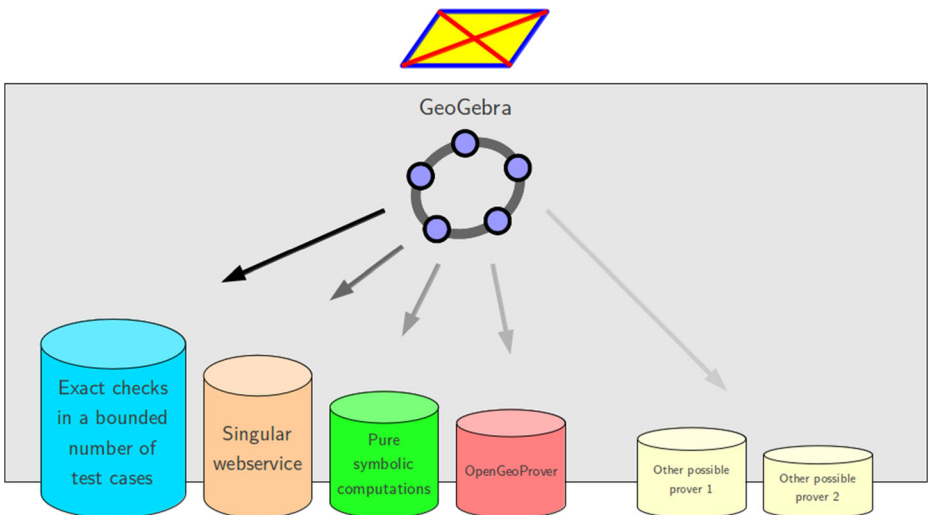


Fig. 4 GeoGebra can choose from several prover subsystems to answer questions

the third list of the conditions above, but to find this geometrically good-looking configuration, the background work of the teacher is still required. (Even in the very special case  $AB = BE = 2CF = 2FD$ , shown in the figure, it can still be a challenge for many students to find the arguments why the theorem holds.) On the other hand, the problem with  $AD \parallel CE$  is still not handled even in this case since the applied technique computes results in projective geometry and the non-Euclidean interpretation cannot be excluded (Fig. 3).

### 3.4 Methods Supported

The new GeoGebra commands proceed by launching the *prover* subsystem,<sup>11</sup> which uses the following engines (cf. the provided references to learn about how the engines work) to decide whether a statement is true:

1. Exact checks in a bounded number of test cases (“Engine 1”, [5], [34], [52]).<sup>12, 13</sup>
2. Algebraization of the given statement and then attempting to find its proof by using Gröbner bases computation (“Engine 2”, [6]). This engine<sup>14</sup> uses outsourced computations by the computer algebra system Singular (so the computation is very fast, [18]) running on a remote web server.
3. Outsourcing the decision to OpenGeoProver,<sup>15</sup> a stand-alone open source prover. OpenGeoProver currently supports Wu’s method (“Engine 3a”, [40]) and the area method (“Engine 3b”, [19]), but will be extended by additional methods in the future.

There is a built-in heuristic that, for a given statement, tries to find the most suitable engine (and the most applicable method within each engine, if it supports several ones) among the available ones. Currently we use a dummy heuristic which calls the provers in the order above. This is based on the expected time of calculation from our current benchmark examples. We are planning to develop more sophisticated portfolio solvers, successfully used in other automated reasoning domains [43, 56] (Fig. 4).

### 3.5 Programming Challenges

GeoGebra is complex software, written mostly in Java, by around 70 developers from several countries. The source code consists of about 7000 Java files representing more than 1,200,000 lines of code. OpenGeoProver was also a complex system already with more than 200 Java files. Enhancement of GeoGebra and OpenGeoProver required programmers with skills in programming, mathematics and community based development.

We had to improve both systems for building up an efficient intercommunication when creating the construction data structure inside GeoGebra and sending it to an acceptable

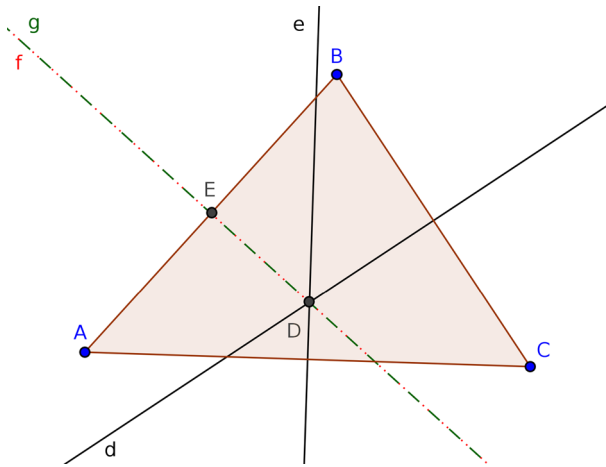
<sup>11</sup>See <http://dev.geogebra.org/trac/browser/trunk/geogebra/common/src/geogebra/common/util/Prover.java> for the detailed Java source code of the prover subsystem in GeoGebra.

<sup>12</sup>This method has a monitoring helper method called Pure symbolic prover which can be used for GeoGebra development purposes, but is too slow for regular use.

<sup>13</sup>The Java source code of this engine can be found at <http://dev.geogebra.org/trac/browser/trunk/geogebra/common/src/geogebra/common/kernel/prover/AbstractProverReciosMethod.java>.

<sup>14</sup>The Java source code of this engine can be obtained from <http://dev.geogebra.org/trac/browser/trunk/geogebra/common/src/geogebra/common/kernel/prover/ProverBotanasMethod.java>.

<sup>15</sup>The Java source code of OpenGeoProver can be found at <https://code.google.com/p/open-geo-prover/source/browse/#svn/branches/geogebra.ogp/OpenGeoProver>.



**Fig. 5** One of the six different formulations of the same geometry statement

form to OpenGeoProver for computation. Also for Engine 1 and 2 we had to create internal data structures for storing and computing polynomials efficiently enough. For Engine 2 we had to implement a lightweight communication protocol between GeoGebra and a web server which runs Singular remotely inside a Linux virtualization. We also had to make some security improvements in Singular to prevent anonymously sent unsandboxed system calls.

Since we use several independent provers and they do not share the same representation of polynomials and other abstract objects, it is out of our scope to describe the implementation details for each prover in this paper. The reader can find all programming nuisances in the freely available source code of each engine.

### 3.6 Achieved Goals

Here we refer to goals described in Section 3.1.

The command line functionality for theorem proving tasks for GeoGebra is in a first stage, but it seems as a suitable integration into the standard user interface.

**Goal 1** would be further supported by adding a Prove tool with a dialog window. Also the extension of the Relation Tool, that automatically detects relations between geometry objects in the construction (numerically at the moment), could be extended by using symbolic computations. Engine 1 can be used by both the desktop and web versions of GeoGebra, but the other engines, however, are not prepared to be multiplatform yet.<sup>16</sup>

**Goal 2** seems to be a drawback for advanced users, but the **ProveDetails** command can be a good compromise. Showing a small set of degeneracy conditions and converting it into a visualized geometry content should be supported in the future.

<sup>16</sup>Engine 2 can already be utilized by using the embedded computer algebra system *Giac* [36] with limited capabilities.

**Goal 3** has been successfully achieved. Engine 1 often gives an answer within 20 ms on a modern workstation. Engine 2 is usually between 50 and 100 ms.<sup>17</sup> Engine 3 gives the result between 100 and 250 ms. Since Engine 3 consists of standalone implementations, it was required to attach it to GeoGebra as an external package. Its binary size is below 75 kilobytes and thus quite small.

**Goal 4** is work in progress.

**Goal 5** has been accomplished by design.

## 4 Classroom Examples and Benchmarks

In this section we provide some detailed comments on the performance of the GeoGebra prover engines over two elementary geometry problems. The first one is about the concurrency of the bisectors of the sides of a triangle; the second one is Simson's theorem. We will show some different formulations (i.e., construction steps for the hypotheses and thesis) of the same statements could have a non negligible impact on the different prover engines performance. We think this is an important observation, since end users of GeoGebra are students rather than researchers, and, thus, we have to take into consideration that they could be describing a given statement in rather unexpected ways.

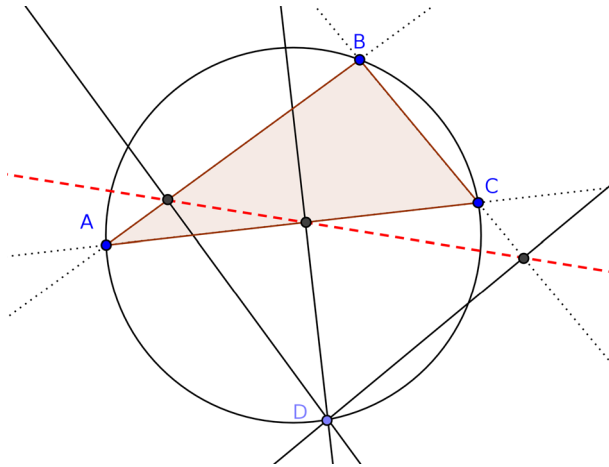
In the last part of this section we present some data on the performance of our ATP implementation on a benchmark suite.

### 4.1 Concurrency of Side Bisectors of a Triangle

Some test cases for the provers are defined in separate files called **circumcenter** $N$  ( $N = 1, \dots, 6$ ) in [35]. One can test the proving methods by exploring different formulations of the same geometry statement:

1. Triangle  $ABC$  is created as a polygon with free vertices. The perpendicular bisectors ( $d, e$ ) of two sides of the triangle are created, and their intersection point is  $D$ . A point  $E$  is the midpoint of the third side, a line  $f$  is **Line**[ $D, E$ ], and  $g$  is perpendicular bisector of the third side. The statement to decide is whether  $f$  is parallel to  $g$  (Fig. 5).
2. This is similar to the first configuration, but points  $A, B$  and  $C$  are created as free objects, and lines through them will be used instead of segments. For this reason the perpendicular bisector of the three *point-pairs* will be used (instead of the three *segments*). This also means that the lines of the sides are not used in the computation but for visualization only. (Internal GeoGebra representation for a triangle and a set of three free points is substantially different, this is why we need to consider this as a different case.)
3. Free points  $A, B, C$  and lines through them are created and perpendicular bisectors of all pairs of the free points are tested whether they are concurrent. By using the **AreConcurrent** command, this can be achieved in a convenient way.
4. Same as the previous, but we use a polygon instead of lines. (Again, this must be considered as a different case because of GeoGebra internals.)
5. We create free points  $A, B$  and  $C$  and their circumcircle. Then two of the bisectors of the pairs of free points are created and their intersection  $D$  is constructed. Now we

<sup>17</sup>Giac is about 3 times slower in the desktop version than Singular and there is another factor of 10 for the web version. Its overall performance is still acceptable in many classroom situations.



**Fig. 6** Simson’s theorem

measure the distance of  $D$  to two free points. If they are the same, it means  $D$  is the same distance from all three free points, thus  $D$  is the center of the circumcircle. Finally, one shall obtain from uniqueness that the perpendicular bisectors are concurrent.

6. Finally, another approach is to create point  $D$  as intersection of bisectors of  $AB$  and  $AC$ , and create point  $E$  as intersection of bisectors of  $AC$  and  $BC$ . Now we prove that  $D$  equals  $E$ .

All formulations can be proved by all our proving methods, except that the 5th one cannot be computed by Engine 1 since it is not capable of dealing with circles. Benchmarking results are approximately the same for the same method for each configuration: Engine 1 returns the result in 8–13 ms, Engine 2 in 38–104 ms, Engine 3 in 98–187 ms or 87–100 ms (depending on the applied prover technique). Since GeoGebra selects Engine 1 as the preferred way for computation, this problem can be solved by GeoGebra usually near 10 ms (Table 1).

### 4.2 Simson’s theorem

Here we consider two possible formulations of Simson’s theorem.

**Table 1** Comparison of proving methods

Formulation	Engine 1	Engine 2	Engine 3a	Engine 3b
#1	10	57	187	100
#2	7	39	124	84
#3	8	38	126	94
#4	8	44	128	91
#5	n.a.	44	98	88
#6	13	104	118	87

1. Create a circle lying on the free points  $A$ ,  $B$ ,  $C$ . Put  $D$  on the circle and define the triangle  $ABC$ . Create perpendiculars on  $D$  to the side lines of the triangle. Create the side lines, too. Create the intersection points of the perpendiculars and the side lines. Prove that they are collinear (by using the new **AreCollinear** command). (See Fig. 6.)
2. Similar to the first formulation, but do not create a triangle, just use side lines.

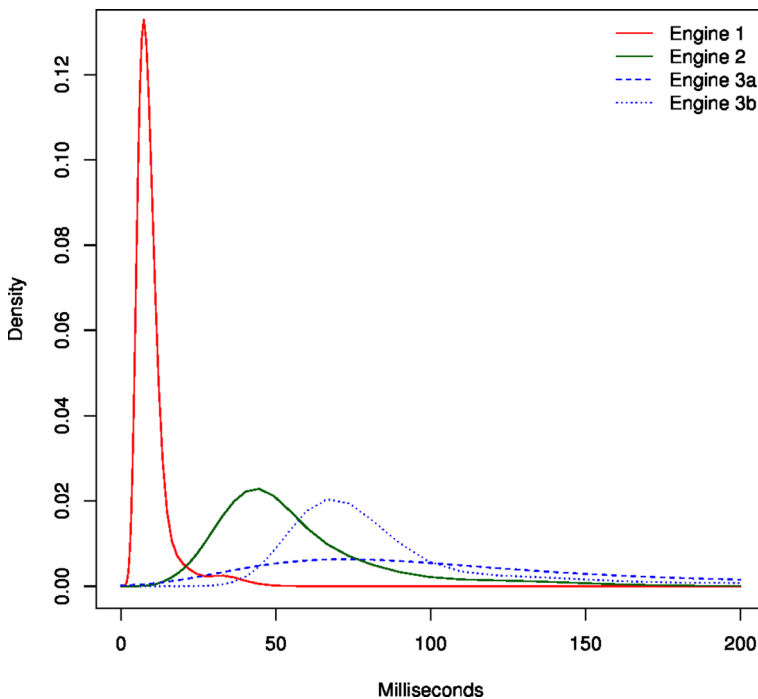
One can see that the second construction has less complexity since there are fewer objects to consider. Despite that, Engine 1 cannot handle this construction at the moment (because of the same reason mentioned in Section 4.1). By contrast, Engine 2 can compute the result by using the algorithm described in [17] (pp. 300–303) and sending the computation request to Singular. The final result is returned in less than 80 ms for both formulations.

Also Engine 3 does a good job with both prover techniques. Wu's method takes about 250 ms and the area method solves the problems in about 150 ms.

### 4.3 Other Tests

GeoGebra has an automated benchmarking suite to measure 60 different conjectures for measuring performance of all implemented prover methods [37].

The prover subsystem in GeoGebra can still be considered as just a prototype, but in many cases it does a remarkable job. On a benchmark set of 60 conjectures, GeoGebra gives 53 correct answers since Engine 1 gives 30 results; Engine 2, 47 results, and Engine 3 gives 47 and 42 correct results for the two different techniques, respectively (the remaining 7 tests



**Fig. 7** Density estimate of benchmark time output visualized on the working tests

return “undefined”) [37]. However, the test database is created for internal testing, and a third-party database is planned to be used soon.

Making a thorough comparison between the presented provers and other provers is quite a formidable task and it is not within the goals of our paper. Even if we manage to make some automatic translation between input formats—this will not lead to a fair comparison. Namely, it is not only an input language issue what is at stake, but also deeper expressibility—some provers natively support some geometric constructs, while for some one has to deal with them in some less efficient way. So, the benchmark set and the translation used could be easily biased towards one or another system. On the other hand, we do believe that the time has just arrived to start creating wide and well-thought open databases with the opportunity to compare open systems.

Still, the overall conclusion is that GeoGebra, endowed with the ATP features we have introduced, could already be used for theorem proving in education. (See also Fig. 7 for a visual comparison of its embedded provers.)

## 5 Educational Relevance

In this section we discuss some basic aspects for teaching proofs in a classroom and how ATP and DGS tools could improve the educational process.

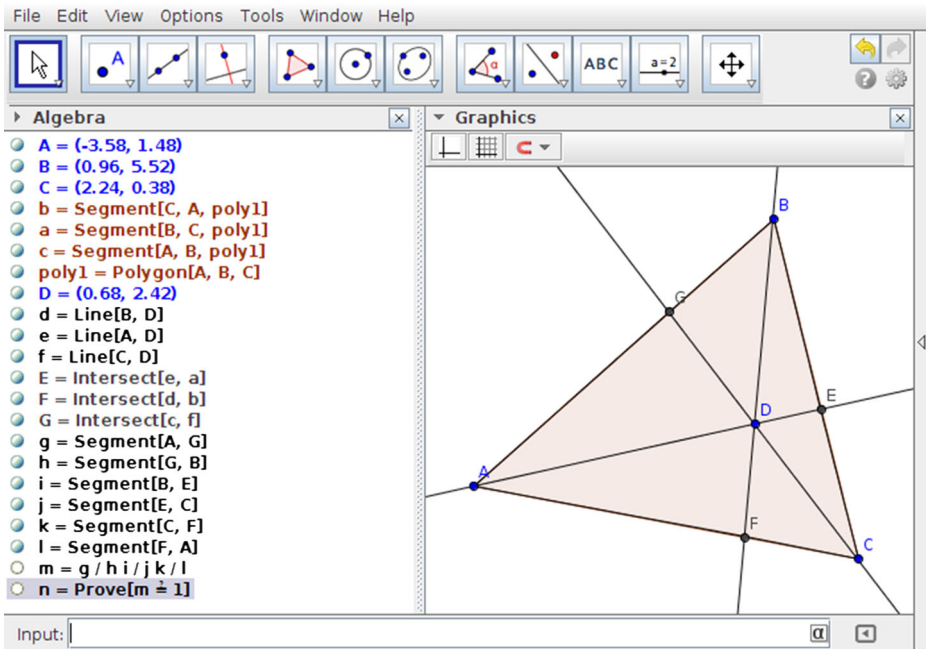
### 5.1 Fundamental Aspects

DeVilliers [20] describes how proofs can support student understanding of mathematical concepts. It outlines the following steps in problem solving towards a proof:

1. Introduction, prerequisites
2. Discovery
3. Verification (testing)
4. Intellectual challenge
5. Systematization (the proof itself)

Step 1 should make the students understand the problem by describing the topic and the involved objects and by presenting a closer look to the applicable techniques. Step 2 should let students to use the related tools on their own, and to make them comfortable enough with the topic as to allow the formulation of conjectures and guesses. This may require much more time than teachers usually have in classroom teaching. At Step 3 students (with or without the help of the teacher) should make as many tests as required for being convinced about the conjectured properties. Step 4 is for collecting and attempting to assemble as many related pieces of information as possible, to prepare for Step 5, which is about constructing the actual solution of the given problem, and about presenting a clear and rigorous reasoning explaining the truth or falsity of the different required steps.

Computers, namely DGS tools, can help students in Steps 1 and 2, in order to get familiar with the involved mathematical objects and to visualize their properties. At Step 1 it is not required to use an ATP-capable DGS in the classroom. For example, to introduce Ceva’s theorem (Fig. 8), it is enough to use a system which can draw lines, create the corresponding intersection points, to measure length of segments and to compute divisions and products. On the other hand, Steps 2 and 3 could be greatly enhanced by computer assistance. For



**Fig. 8** Ceva's theorem in GeoGebra: Given a triangle  $ABC$ , let the lines  $AD$ ,  $BD$  and  $CD$  be drawn from the vertices to a common point  $D$  to meet opposite sides at  $E$ ,  $F$  and  $G$  respectively. Then  $AG/GB \cdot BE/EC \cdot CF/FA = 1$

example, a DGS/ATP system could systematically help the user conjecturing in the correct direction. See Section 6 for our further plans along these lines.

Of course, DGS tools using probabilistic methods for verification can give almost sure results very quickly, and they may be incorrect only in very rare cases. In spite of the possible practical demand to use it, we emphasize here that there is a sharp dividing line in education use between “almost sure” and “sure”. Unlike other disciplines, mathematics as science can indeed provide “sure” results about relations of its abstract objects. This is a special property of mathematics and logic, making mathematics a complete different discipline than others. From the educational point of view, students need to make a difference between “ $2+2$  equals always 4” and “ $2+2$  equals most of the time 4”. Actually, in real life the every day rules have just a certain possibility: physical laws (like gravity) mostly work in normal circumstances (at least, they usually do), and machines controlled by physical laws (like a vacuum cleaner) work in most cases without being repaired for a long time. On the contrary, mathematical laws are essentially different.

Existing ATP tools usually have a more precise conclusion, but using them may be inadequate from the educational point of view: they may require using special specification languages, they could have a non-intuitive user interface, and, in some cases, results may be given too slowly. Our improvements on GeoGebra may be a bridge between DGS and ATP systems, since its intuitive user interface could help the average student to describe the problem and to ask questions to the underlying prover engines.



Nevertheless, Step 4 is not covered by our work at the moment. In this step the student should collect minor facts of the geometrical construction which will build up the complete sequence of reasons explaining how to solve a given problem. GeoGebra already has a partial support for this by providing the Relation Tool, but a detailed investigation is not yet supported. An interesting approach for collecting information about an existing construction is Z. Magajna's *OK Geometry* software [39] which could be a step forward in helping students to build up their own proofs.

Finally, Step 5 is still a question of automated finding and verifying human readable proofs, including the consideration of possible ways on how a student could handle a point-and-click user interface to design and to describe the whole proof of a geometric theorem.

As [24, p. 905] emphasizes that *proofs which best promote understanding . . . much more likely to yield not only knowledge that, but also knowledge why*. In our vision GeoGebra could focus on this approach in future enhancements of the proving subsystem.

## 5.2 Other Uses and Abuses

Step 3 only gives a yes/no answer (or unknown) for a statement like a black box. While this seems to be just a small step if we are to consider the computer a real tutor in learning mathematics, we must emphasize that generating full proofs by an ATP may also be misleading. First of all, today's ATP systems usually generate too long proofs which cannot show the beauty and elegance of geometrical proofs. Moreover, many methods do not provide geometric proofs, but algebraic computations. That is why we simply try to give a yes/no answer at the moment. Of course, such an answer can also be dangerous if no preparation was done in the previous classroom process, i.e., if Steps 1 and 2 were not fully elaborated in advance. This can lead to student responses like 'So what?' if the software simply tells whether a theorem is true or not.

An important possible use of a yes/no answer is automated checking of open ended tests. For example, the teacher asks the student to create a right triangle by using a DGS. When the teacher designs this question, he/she could be thinking of one correct construction, where the third vertex is the output and it depends on the first two vertices as inputs, by using Euclidean steps only (i.e., only a compass and a ruler). During the test time the teacher's construction steps will be hidden for the student, but a built-in ATP system could check if the output vertex of the student coincides with the one of the teacher's template—even if the student intermediate steps are different from those of the teacher. Here an ATP tool could give a sure yes/no answer, and the computer would be able to decide whether the student has solved the problem correctly or not. Since GeoGebra (powered with ATP features) is already fast enough in such computations, we think it opens the door to create computer aided open ended tests in geometry. This subject has also been studied in [27], using a numerical comparison between template constructions and those provided by users.

We can also think of open ended tests where finding the intermediate steps can also be crucial. For example, given a circle  $c$  with its center  $O$  and an external point  $P$ , the student's task is to construct a tangent line  $t$  from  $P$  to  $c$ . The teacher knows that the basic idea for the usual solution is to create the midpoint  $M$  of  $OP$ . It helps drawing a second circle  $d$  with center  $M$  aligning on both  $O$  and  $P$ , and now the intersection points of  $c$  and  $d$  will define the tangent points (because of Thales' circle theorem). Here finding the importance of  $M$  is already an intellectual challenge and thus the teacher may highlight "important intermediate points" in his/her template, not only the possible final results  $t_1$  and  $t_2$ . In case

the student finds  $M$  by constructing it somehow the software tool could give encouragement by confirming the good direction.

Ultimately, we also expect a dramatical change of the idea of mathematical reasoning from the teacher's perspective as well. Proof is traditionally considered as a human act which requires intellectual work. But to utilize a computer to obtain the conclusion is actually something purely mechanical. This may change the teacher's role fundamentally, raising her to a higher level in the education process.

## 6 Conclusions and Future Work

We released the theorem prover subsystem as a part of GeoGebra version 5, in September 2014. In the next forthcoming GeoGebra versions we plan to enhance the existing engines and to add implementations of other proving methods to OpenGeoProver. Furthermore, in the future we want to make use of the database of the *GeoThms* project [44] as a benchmark, and possibly use Chou's [9] and Wang's [50] collections as well.

Integration of theorem proving features in GeoGebra is not an ad-hoc task, but a complex process yielding an evolving system, meeting users' needs and progress in theorem proving technology. After getting feedback about the current features, the long term plans are careful GUI changes in GeoGebra, that will turn using proving features more comfortable for the end user. From the educational perspective, GeoGebra could then be used as an expert system in elementary geometry which not only tells a yes/no answer but is capable of showing a step-by-step explanation if a machine generated proof is considered human readable. Such efforts have already been started by extending the OpenGeoProver with the ability to generate more readable proofs, based on the mass point method ([60]).

When a construction is given, GeoGebra could also automatically identify certain "interesting" properties on the construction. For example, when the circumcenter, the centroid and the orthocenter of a triangle have already been constructed, GeoGebra could "know" that these points are collinear and provide this information to the user when asked. Such an auto-relation feature could extend the already existing Relation Tool.

GeoGebra could also give a counterexample when the checked statement is not always true. For Engine 1 this could be achieved immediately by post-processing its internal computations.

Despite the fact that these improvements should be intuitive enough we still plan to involve a wider group of experts to help creating explanatory materials for teachers and students and share them with the community. Some demonstrational examples are already available to introduce the new GeoGebra commands (see, for instance <http://wiki.geogebra.org/en/ProveDetails.Command>, <http://tube.geogebra.org/student/m55158>, <https://www.youtube.com/watch?v=7aDe0YMm-OE> or <http://tube.geogebra.org/student/b104296>).

The overall goal of all these improvements is to support the problem solving process of students related to proving, in particular, in the geometry context.

## References

1. Avigad, J., Dean, E., Mumma, J.: A formal system for Euclid's Elements. *Rev. Symb. Log.* **2**(4), 700–768 (2009)

2. Baulac, Y., Bellemain, F., Laborde, J.M.: *Cabri Geometry II*. Dallas, Texas Instruments (1994)
3. Bernat, P.: Using dynamic geometry environments for problem solving: CHYPRE: an interactive environment for elementary geometry problem solving. Abstract of a presentation at The 8th International Congress on Math Education (ICME 8). Retrieved 06.08.13, from, [http://mathforum.org/mathed/seville/bernat/abst\\_bernat.html](http://mathforum.org/mathed/seville/bernat/abst_bernat.html). Seville (1996)
4. Bertot, Y., Guilhot, F., Pottier, L.: Visualizing geometrical statements with GeoView. *Electr. Notes Theor. Comput. Sci.* **103**, 49–65 (2004)
5. Botana F., Kovács, Z., Recio, T., Weitzhofer, S.: Implementing theorem proving in GeoGebra by using a Singular webservice, or by exact check of a statement in a bounded number of test cases, <http://ggbl.idm.jku.at/~kovzol/talks/eaca12/EACA2012-BotanaKovacsRecioWeitzhofer.pdf> (2012)
6. Botana, F., Kovács, Z., Weitzhofer, S.: Implementing theorem proving in GeoGebra by using a Singular webservice. In: *Proceedings EACA 2012. Libro de Resúmenes del XIII Encuentro de Álgebra Computacional y Aplicaciones*, pp. 67–70. Alcalá de Henáres, Universidad de Alcalá (2012)
7. Botana, F., Valcarce, J.: A dynamic symbolic interface for geometric theorem discovery. *Comput. Educ.* **38**, 21–35 (2002)
8. Buchberger, B. In: Rice, J.R. (ed.): *Applications of Gröbner Bases in Non-Linear Computational Geometry*, pp. 59–87. Springer, New York (1987)
9. Chou, S.C.: *Mechanical Geometry Theorem Proving*. D. Reidel Publishing Company, Dordrecht (1988)
10. Chou, S.C., Gao, X.S., Zhang, J.Z.: Automated production of traditional proofs for constructive geometry theorems. In: Vardi, M. (ed.) *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pp. 48–56. IEEE Computer Society Press (1993)
11. Chou, S.C., Gao, X.S., Zhang, J.Z.: *Machine Proofs in Geometry*. World Scientific, Singapore (1994)
12. Chou, S.C., Gao, X.S., Zhang, J.-Z.: Automated generation of readable proofs with geometric invariants (II). Theorem proving with full-angles. *J. Autom. Reason.* **17**, 349–370 (1996)
13. Chou, S.C., Gao, X.S., Zhang, J.Z.: An introduction to Geometry Expert. In: McRobbie, M.A., Slaney, J.K. (eds.) *CADE 13*, volume 1104 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag (1996)
14. Chou S.C., Gao X.S.: Automated reasoning in geometry. In: *Handbook of Automated Reasoning*. Elsevier, and MIT Press (2001)
15. Coelho, H., Moniz-Pereira, M.: Automated reasoning in geometry theorem proving with Prolog (1986). *J. Autom. Reason.* **2**, 329–390 (1986)
16. CoCoATeam: CoCoA: A system for doing Computations in Commutative Algebra. Retrieved 02.09.13, from <http://cocoa.dima.unige.it> (2012)
17. Cox, D., Little, J., O’Shea, D.: *Ideals, Varieties and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra (Undergraduate Texts in Mathematics)*. Springer-Verlag, Secaucus, NJ (2008)
18. Decker, W., Greuel, G.-M., Pfister, G., Schönemann, H.: Singular 3-1-6 — A Computer Algebra System for Polynomial Computations. Retrieved 02.09.13, from, <http://www.singular.uni-kl.de> (2012)
19. Desfontaines, D.: Theorem proving in GeoGebra: Implementing the Area Method into OpenGeoProver (Internship report). Retrieved 02.02.13, from, <http://www.eleves.ens.fr/home/desfonta/InternshipReport-v2.pdf> (2012)
20. DeVilliers, M.: Rethinking Proof with Sketchpad. Key Curriculum Press. Retrieved 02.02.13, from, <http://mzone.mweb.co.za/residents/profmd/proof.pdf> (1999)
21. Gao, X.S., Lin, Q.: MMP/Geometer a software package for automated geometric reasoning. In: Winkler, F. (ed.) *Automated Deduction in Geometry: 4th International Workshop, (ADG 2002)*, volume 2930 of *Lecture Notes in Computer Science*, 44–66. Springer-Verlag (2004)
22. Gelernter, H.: Realisation of a geometry-proving machine. In: *Proceedings of the International Conference on Information Processing*, 273–282, Paris, vol. 15-20, p. 1959 (1959)
23. Gressier, J.: Geometrix IV. Retrieved 06.08.13, from, <http://geometrix.free.fr> (2013)
24. Hanna, G., Jahnke, H.N.: Proofs and proving. In: A. J. Bishop A. J., Clements K., Keitel C., Kilpatrick J., Laborde C. (eds.) *International Handbook of Mathematics Education, Part Two*, pp. 877–908. Kluwer Academic Publishers, Dordrecht (1996)
25. Hanna, G.: The ongoing value of proof. *J. Math. Didaktik* **18**(2), 171–185 (1997)
26. Hohenwarter, M.: Ein Softwaresystem für dynamische Geometrie und Algebra der Ebene, master thesis. Paris Lodron University, Salzburg (2002)
27. Isotani, S., Brandão, L.O.: An algorithm for automatic checking of exercises in a dynamic geometry system: iGeom. *Comput. Educ.* **51**, 1283–1303 (2008)
28. Jackiw, N.R.: *The Geometer’s Sketchpad, v3.0*. Key Curriculum Press, Berkeley, CA (1995)

29. Janičić, P.: Geometry constructions language. *J. Autom. Reason.* **44**(1-2), 3–24 (2010)
30. Janičić, P., Narboux, J., Quaresma, P.: The area method: a recapitulation. *J. Autom. Reason.* **48**(4), 489–532 (2012)
31. Kapur, D.: Using Gröbner bases to reason about geometry problems. *J. Symb. Comput.* **2**(4), 399–408 (1986)
32. Kortenkamp, U.: Foundations of Dynamic Geometry. Ph.D. Dissertation, ETH, Zurich (1999)
33. Kortenkamp, U., Richter-Gebert, J.: Using automatic theorem proving to improve the usability of geometry software. In: Workshop on Mathematical User Interfaces (2004)
34. Kovács, Z., Recio, T., Weitzhofer, S.: Implementing theorem proving in GeoGebra by exact check of a statement in a bounded number of test cases. In: Proceedings EACA 2012. Libro de Resúmenes del XIII Encuentro de Álgebra Computacional y Aplicaciones, pp. 123–126. Alcalá de Henáres, Universidad de Alcalá (2012)
35. Kovács, Z., Weitzhofer, S., Desfontaines, D., Janičić, P.: Test cases for benchmarking statements. Retrieved 11.09.12, from, <https://dev.geogebra.org/trac/browser/trunk/geogebra/test/scripts/benchmark/prover> (2012)
36. Kovács, Z., Parisse, B.: Giac and GeoGebra — improved Gröbner basis computations, Special Semester on Applications of Algebra and Number Theory, Workshop 3 on Computer Algebra and Polynomials. Retrieved 18.10.14, from, <https://www.ricam.oeaw.ac.at/specsem/specsem2013/workshop3/slides/parisse-kovacs.pdf> (2013)
37. Kovács, Z.: Prover benchmark for GeoGebra 5.0.14.0. Retrieved 10.09.14, from, <http://test.geogebra.org/~kovzol/data/Prove-20150219/> (2014)
38. Luengo, V.: Cabri-Euclide: Un micromonde de Preuve intégrant la réfutation, PhD thesis. Université Joseph Fourier, Grenoble (1997)
39. Magajna, Z.: An observation tool as an aid for building proofs. *Electronic J. of Mathematics and Technology* 5/3. Retrieved 02.02.13, from, <http://www.freepatentsonline.com/article/Electronic-Journal-Mathematics-Technology/270980194.html> (2011)
40. Marić, F., Petrović, I., Petrović, D., Janičić, P.: Formalization and implementation of algebraic methods in geometry. *Electron. Proc. Theor. Comput. Sci.* **79**, 63–81 (2011)
41. Narboux, J.: A graphical user interface for formal proofs in geometry. *J. Autom. Reason.* **39**(2), 161–180 (2007)
42. Narboux, J.: Geoproof, a user interface for formal proofs in geometry. In: Mathematical User-Interfaces Workshop, Schloss Hagenberg, Linz, Austria. Electronic proceedings at <http://www.activemath.org/workshops/MathUI/07/proceedings/Narboux-Geoproof-MathUI07.html> (2007)
43. Nikolić, M., Marić, F., Janičić, P.: Instance-based selection of policies for SAT Solvers. In: Theory and Applications of Satisfiability Testing – SAT 2009, volume 5584 of Lecture Notes in Computer Science, pp. 326–340. Springer-Verlag (2009)
44. Quaresma, P., Janičić, P.: GeoThms — a web system for euclidean constructive geometry. *Electron. Notes Theor. Comput. Sci. (ENTCS)* **174**(2), 35–48 (2007). doi:10.1016/j.entcs.2006.09.020
45. Recio, T., Vélez, M.P.: Automatic discovery of theorems in elementary geometry. *J. Autom. Reason.* **23**, 63–82 (1999)
46. Schwartz, J.L., Yerushalmy, M.: The Geometric Supposer. Sunburst Communications, Pleasantville, NY (1983)
47. Sutherland, I.E.: Sketchpad: A Man-Machine Graphical Communication System, Lincoln Laboratory, Massachusetts Institute of Technology via Defense Technical Information Center, Technical Report No. 296. Lexington, MA. Retrieved 02.02.13, from, <http://handle.dtic.mil/100.2/AD404549> (1963)
48. Tall, D.: Cognitive development, representations and proof. In: Proceedings of the conference Justifying and Proving in School Mathematics, pp. 27–38. Institute of Education, London (1995)
49. The GeoGebra Team: Reference: Command Line Arguments — GeoGebraWiki (2015). [http://wiki.geogebra.org/en/Reference:Command\\_Line\\_Arguments](http://wiki.geogebra.org/en/Reference:Command_Line_Arguments)
50. Wang, D.: Geother 1.1: Handling and proving geometric theorems automatically. In: Automated Deduction in Geometry, volume 2930 of Lecture Notes in Artificial Intelligence, pp. 194–215. Springer-Verlag (2004)
51. Wang, D.: Elimination Practice: Software Tools and Applications. London: Imperial College Press. Geother 1.1: Handling and proving geometric theorems automatically. In: Automated Deduction in Geometry, volume 2930 of Lecture Notes in Artificial Intelligence, pp. 194–215. Springer-Verlag (2004)
52. Weitzhofer, S.: Mechanic Proving of Theorems in Plane Geometry. Johannes Kepler University, Linz, Austria (2013). <http://test.geogebra.org/~kovzol/guests/SimonWeitzhofer/DiplArbeit.pdf>
53. Wilson, S., Fleuriot, J.: Combining dynamic geometry, automated geometry theorem proving and diagrammatic proofs. In: Workshop on User Interfaces for Theorem Provers (UITP) (2005)

54. Wolfram Research, Inc.: *Mathematica*, Version 3.0. Champaign, IL (1996)
55. Wu, W.T.: On the decision problem and the mechanization of theorem proving in elementary geometry. *Sci. Sin.* **21**, 157–179 (1978)
56. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla: Portfolio-based algorithm selection for SAT. *J. Artif. Intell. Res. (JAIR)* **32**, 565–606 (2008)
57. Ye, Z., Chou S.C., Gao, X.S.: An Introduction to Java Geometry Expert. In: *Automated Deduction in Geometry*, 7th International Workshop, ADG 2008, Shanghai, China, September 22–24, 2008, Revised Papers, volume 6301 of *Lecture Notes in Computer Science*, pp. 189–195. Springer-Verlag (2011)
58. Ye, Z., Chou, S.C., Gao, X.S.: Visually dynamic presentation of proofs in plane geometry. *J. Autom. Reason.* **45**(3), 213–241 (2010)
59. Zhang, J.Z., Yang, L., Deng, M.: The parallel numerical method of mechanical theorem proving. *Theor. Comput. Sci.* **74**(3), 253–271 (1990)
60. Zou, Y., Zhang, J.Z.: Automated Generation of Readable Proofs for Constructive Geometry Statements with the Mass Point Method. In: *Lecture Notes in Computer Science*, Volume 6877, *Automated Deduction in Geometry*, 221–258 (2011)