GeoGebra Discovery Automated Reasoning Tools
A Tutorial

Zoltán Kovács, Tomás Recio and M. Pilar Vélez

May 4, 2024

Abstract

This document introduces, describes and exemplifies the technical features of some recently implemented automated reasoning tools in a fork of the dynamic mathematics software GeoGebra. The new tools are based on symbolic computation algorithms, allowing the automatic and rigorous proving and discovery of theorems on constructed geometric figures. Some examples of the use in the classroom of such commands are provided, including one describing how intuitive handling of GeoGebra Discovery automated reasoning tools may result in unexpected outputs. In all cases the emphasis is made in the potential utility of these tools as a guiding stick to foster student activities (exploration, reasoning) in the learning of elementary geometry. Moreover, a collection of appendices describing other, more sophisticated, low-level GeoGebra Discovery tools (Prove, ProveDetails), as well as instructions on how to obtain the translation of GeoGebra Discovery commands into other languages, and details about debugging, are included.

1 Introduction

The software tool GeoGebra (`https://www.geogebra.org`) can support the teaching of Euclidean plane geometry theorems in various ways. In this tutorial we focus on some new features, based on symbolic computations, allowing the automatic proving and discovery of theorems on geometric figures constructed with GeoGebra Discovery, a fork of GeoGebra, available at `https://github.com/kovzol/geogebra-discovery`.

This novel technology attempts to address recent challenges in mathematics education [9, 8, 19, 16, 18], it is however still in an experimental phase in the classroom [14, 11]. This document summarizes its technical possibilities and describes in detail how to use them through some examples. The tutorial is aimed at users who already have some basic knowledge on GeoGebra and want to learn about the recently implemented automated reasoning commands.

The next Section 2 provides a brief introduction to GeoGebra Discovery, as the host software for the automated reasoning package. Section 3, after a global introduction to these tools, describes in detail, through examples, the characteristics of the `Relation`, `LocusEquation` and `Envelope` commands, considered as the most immediate and useful for the classroom context. It is appended with some technical remarks, providing some hints about cases of possible strange behaviour of these commands (for instance, that automatic reasoning is not yet possible for figures that include arbitrary degree curves).

Section 4 presents two detailed teaching scenarios that could be approached with the automated reasoning tools, involving, respectively, Thales' circle and the midline theorems. Further examples are summarily mentioned in 4.3. On the other hand, Thales' circle theorem is also addressed in Section 5, calling the reader to be beware of some common misinterpretations in using automated reasoning tools that may result in unexpected outputs.

The paper ends with some general conclusions and a collection of three appendices: one, that describes other, relatively sophisticated for just classroom use, automated reasoning commands (`Prove`, `ProveDetails`, etc.); a second appendix that explains how to get the translation of the commands to different languages; and, finally, a short note on how to get debug messages to allow a technical user to have further information about the performed computations.

2 Starting GeoGebra Discovery

GeoGebra Discovery is available on many platforms, including desktop or laptop computers with various operating systems installed.

Some specific GeoGebra Discovery tools could however differ on the different platforms. Also, the user experience on the various platforms may be different: the required symbolic computations may need a high amount of calculations and the underlying hardware or software components could or could not support some steps completely.

The recommended platform for classroom use can vary. The fastest results can be obtained by fast desktop (or laptop) computers, but in this case the software must be downloaded and installed by the user. Some examples in this tutorial will work only in the "desktop" version (which is created for Microsoft Windows, Apple Macintosh and Linux desktop and laptop computers). On the other hand, the "web" version (it is available at `https://autgeo.online`) does not require installation by the user: it will run in a modern web browser. The "web" version is however usually slower than the desktop one: the symbolic computations may be slower by a magnitude.

There is a continuous workflow improving GeoGebra Discovery's automated reasoning tools. Thus, the best practice is to always use the latest version from `https://github.com/kovzol/geogebra/releases`. A two-weekly update is usually performed for all versions. The list of newest changes can be found at `https://github.com/kovzol/geogebra/commits/master/`—although this piece of information is intended only for advanced users and developers. Everyone is welcome to join the mailing list at `https://groups.google.com/g/geogebra-discovery`.

Some tools provided by GeoGebra Discovery are also available in the mainstream version of GeoGebra. They are, however, not maintained by the GeoGebra team at the moment. As a result it is possible that GeoGebra Discovery performs significantly better in various application fields than the mainstream version. Documentation of the automated reasoning tools in GeoGebra can be found at `https://github.com/kovzol/gg-art-doc` and [13].

To get up-to-date information on recent papers on GeoGebra Discovery, the best option is to visit `https://github.com/kovzol/geogebra-discovery?tab=readme-ov-file#references`. Some of these papers are listed in the References, but not all of them.

3 Automated Reasoning Tools

Automated reasoning tools are a collection of GeoGebra Discovery features and commands that allow to conjecture, discover, adjust and prove geometric statements in a dynamic geometric construction.

To start with, the user needs to draw a geometric figure by using certain tools listed by default on top of the main window in GeoGebra and GeoGebra Discovery. After constructing the figure, GeoGebra has many ways to promote investigating geometrical properties of a figure, through various tools and settings:

1. By *dragging the free objects*, the behaviour of their dependent objects can be visually investigated.
2. The *Relation* tool helps comparing objects and obtaining relations among them.
3. By setting on/off the *trace* of a constructed object, the movement of a "descendant" object will be visualized when its "parent" objects are changing.
4. The *Locus* tool shows the trace of an object for all possible positions of a parent object moving on a certain path.
5. By typing the `Relation` or `Locus` command in GeoGebra's *Input Bar* more refined information can be obtained.

These methods are usually well known by the GeoGebra community, and therefore they are well documented, and many examples can be found on them at GeoGebra Materials (`https://www.geogebra.org/materials/`). On the other hand, GeoGebra also offers symbolic automated reasoning tools for generalizing some observed/conjectured geometric properties:

6. The Relation tool and command can be used to recompute the numerical results symbolically.
7. The `LocusEquation` command refines the result of the Locus command by displaying the algebraic equation of the graphical output (see Section 3.2.4 for a list of possible limitations).
8. The `LocusEquation` command can investigate implicit loci.
9. The `Envelope` command computes the equation of a curve which is tangent to a family of objects while a certain parent of the family moves on a path.

In addition, GeoGebra Discovery extends these feature by the following items:

10. The `LocusEquation` and `Envelope` commands are available as tools.

11. The Discover tool and command detect for geometric relationships between objects that are in true in general.
12. The `StepwiseDiscovery` command and the corresponding tool invokes the previously mentioned feature on each newly created point.
13. The `Plot2D` command plots a topologically faithful visual representation of logical connectives of algebraic equations or inequalities.
14. The Relation tool and command can compare lengths of segments symbolically. This may involve inequalities as well.
15. The `RealQuantifierElimination` command (available in the *CAS View*) takes the quantified input formula and computes an equivalent formula that does not contain any quantifiers.
16. The `ShowProof` command gives a step-by-step algebraic proof for the input geometric statement. Its output is shown in the CAS View.

3.1 High and low-level tools

GeoGebra Discovery provides the above high-level methods to promote investigating geometry theorems. These are considered as “high-level” tools because of the simplicity of their format; and, thus, they are intended to be directly used in classrooms. There are also other, more complicated, ways to learn more on the mathematical background of a given statement or just to help in troubleshooting. Those “low-level” methods are listed in Section 6, and therefore not suggested for direct use among students.

Obviously, some of the listed methods are easier, and others are more difficult. For instance, using the command line in the Input Bar in GeoGebra can be considered as a more difficult task for most users than using the *Toolbar*. Thus, it can be suggested that a teacher first shows the students how to deal with the easier methods, and later demonstrates the more demanding ways, when the students have enough experiments done and are a little bit more acquainted with the reasoning tools.

3.2 Tools with symbolic support

As mentioned above, some automated reasoning tools are provided with symbolic support. This feature allows to verify in a mathematically rigorous way general statements of Elementary Geometry that have been conjectured by the user.

A general hint is that the user should start GeoGebra Discovery on startup in the “graphing calculator” mode. This turns on showing the labels on each newly added object—this can be crucial for the Relation tool and command when reporting on various relations.

In most cases, however, the axes are not necessary to be shown: they can be switched off when the *Move* tool is active (it is the leftmost icon showing an arrow cursor) by right-click in the *Graphics View* and de-selecting the *Axes* setting.

In some cases, the *Algebra View* is not needed to be displayed—unless the equations of the implicit curves are to be investigated in detail. This can, however, be done also by changing the object’s label to contain its value, too. (To do so, by right-clicking on the object, choosing *Object Properties*, the user needs to set *Show Label* to *Value* on the *Basic* tab.)

For the `ShowProof` command it is necessary to have the CAS View open. Before issuing the command, however, it is suggested to clear its content (that is, delete all lines).

3.2.1 The Relation tool and command

GeoGebra Discovery’s Relation tool and command shows a message box that gives the user information about the relation between two or more objects. This command allows the user to find out numerically (that is, for the drawn construction with precisely assigned coordinates to each point) whether

- two lines are perpendicular,
- two lines are parallel,
- two (or more) objects (points, segment lengths, polygon areas, angles) are equal,
- a point lies on a line or conic,

- a line is tangent or a passing line to a conic,
- three points are collinear,
- three lines are concurrent (or parallel),
- four points are concyclic (or collinear).

Some of these checks can also be performed symbolically, that is, the statement can be verified rigorously for the general case (with arbitrary coordinates) and not only for the displayed concrete geometric construction.

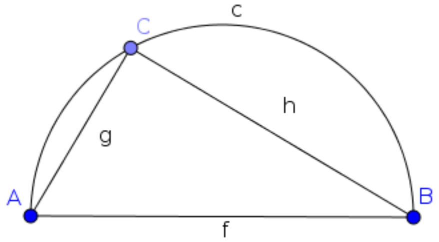
When using the Relation *tool*, the user points on two objects and gets a message box as shown in the figure below. Alternatively, two, three or four objects can be selected by the selection rectangle to invoke the message box. To prevent the user to select unneeded objects in the applet it is also possible to disallow selection of unnecessary objects by right clicking on the object, selecting Object properties, choosing the *Advanced* tab and unchecking *Selection allowed*.

When using the Relation *command*, the user types one of the following formulas in the Input Bar:

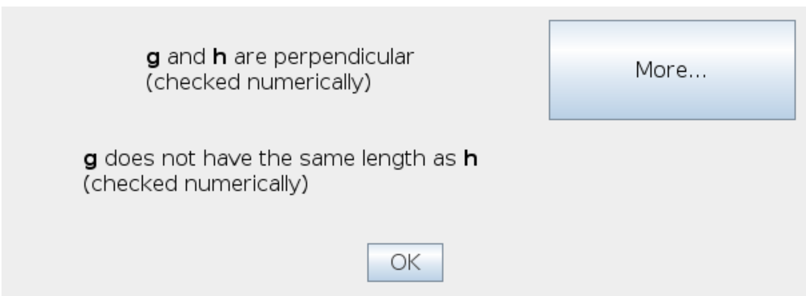
- Relation(<Object>, <Object>)
- Relation({ <Object>, <Object> })
- Relation({ <Object>, <Object>, <Object> })
- Relation({ <Object>, <Object>, <Object>, <Object> })

When the message box is shown with one or more true numerical statements on the objects, there may be a button "More…" shown if there is symbolic support for the given statement. When clicking "More…", shortly the numerical statement will be updated to a more general symbolic one, stating or denying the validity of the Relation for arbitrary instances of the given construction (i.e. if some two lines were perpendicular just in the precisely given position or if they are perpendicular in general, etc.).

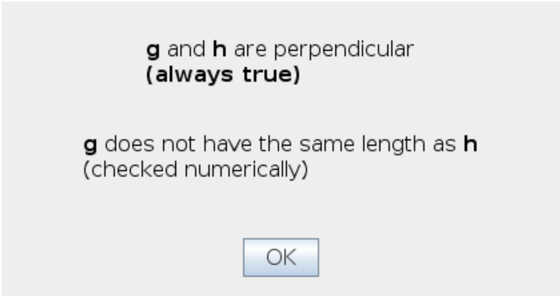
Example (Thales' circle theorem) Here we want to explore the possible perpendicularity of segments AC and BC, where C is a point on a circle, while AB is a diameter thereof. We can proceed as follows:



1. By using the *Segment* tool, construct AB.
2. By choosing the *Semicircle through 2 Points* tool, create arc c.
3. Put point C on c, by using *Point on Object*.
4. Create segments AC and BC and denote them by g and h, respectively.
5. Compare g and h by using the Relation tool and pointing on g and h by the mouse, or type Relation(g,h) in the Input Bar. The following message will be shown:



6. Click “More…”—the message will be changed as follows:



Remark that Relation (step 5) looks for relations between g and h from the coordinates and equations assigned to the drawn construction. However, by clicking “More…” (step 6) we verify that g and h are perpendicular for any points A and B we can choose at step 1.

Sometimes, the relationship among certain objects holds only under certain conditions, that is, not necessarily “always”. In such cases, if possible, some sufficient conditions would be displayed by the Relation tool. Otherwise GeoGebra Discovery just remarks that the statement is “true if non-degenerate”. This must be interpreted as meaning that the statement is “generally true”, but in some side cases (which are ‘a small number of cases’ compared to the general case) the statement may fail.

The symbolic result of Relation can be negative as well, even if the numerical check is positive. For example, by defining two points $P = (0,0)$ and $Q = (0,0)$ Relation compares them numerically, but the symbolic check will result in “P and Q are equal (false in general)”, because the two given points are considered, in the general symbolic approach, as two free points, with arbitrary coordinates.

A complete overview of the various results of Relation can be found in Section 6.1.4 on page 20.

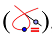
3.2.2 The LocusEquation tool and command

The `LocusEquation` command calculates the equation of a locus and plots it as an implicit curve. It is also available as a tool. There are two kinds of usages:

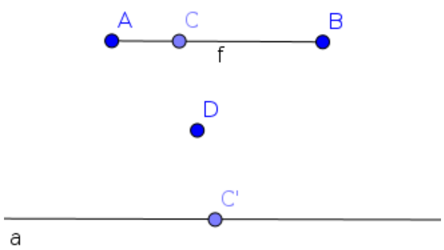
- **Explicit locus.** Consider an input point $\mathscr{I}$ on a path $\mathscr{P}$, some construction steps, and an output point $\mathscr{O}$. The task is to determine the equation $\mathscr{E}$ of the locus of $\mathscr{O}$ while $\mathscr{I}$ is moving on $\mathscr{P}$, and then plot $\mathscr{E}$. Point $\mathscr{I}$ is usually called *mover*, $\mathscr{O}$ is the *tracer*. $\mathscr{E}$ is called the *locus equation*, and its graphical visualization is the *locus*.

  The syntax of the command is

  ```
  LocusEquation( <Point Tracer>, <Point Mover> ).
  ```

  As a tool (), after selection it requires the two inputs described above.

  Example Let us present the second kind of usage through a particular way of building the symmetric of a segment with respect to a point. This particular example allows us to exemplify better some of the benefits and problems of the `LocusEquation` command.



  1. By using the *Segment* tool, construct AB. This automatically creates segment f.
  2. Put point C on f.

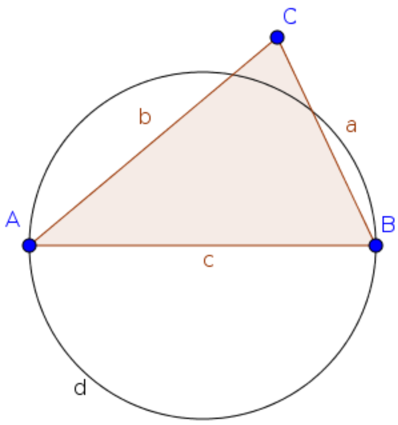
3. Create point D by using the *Point* tool.
4. By using the *Reflect about Point* tool, reflect C about D. This defines C'.
5. Type `LocusEquation(C',C)` in the Input Bar. Now the implicit curve a will be computed and plotted. This should be a segment (the mirror image of f about D), but GeoGebra Discovery needs—and automatically does—to handle f as a line instead of a segment (for algebraic geometric reasons regarding the involved symbolic computation algorithms), thus its mirror image is also a line.
   - Alternatively, you can select the LocusEquation tool and click C' and then C (or in a different order). This may be a simpler method for some users.
6. Try dragging each draggable objects. It can be visually concluded that the mirror image of a segment about a point is always parallel to the preimage.

- **Implicit locus.** Consider a given input point $\mathscr{I}$, either as a free point, or on a path $\mathscr{P}$. Moreover, assume some construction steps are also given. The user claims a Boolean condition $\mathscr{C}$ holds on some objects of the construction. The task is to determine an equation $\mathscr{E}$ such that for all points $\mathscr{I}'$ of it, if $\mathscr{I} = \mathscr{I}'$, then $\mathscr{C}$ holds. Again, $\mathscr{E}$ is called locus equation, and its graphical representation is the locus.

  The syntax of the command is

  ```
  LocusEquation( <Boolean Expression>, <Point> ).
  ```

  Example Given a triangle ABC, and the circle having AB as diameter, find the locus of C such that $AC^2 + BC^2 = AB^2$ (a converse of Pythagoras' Theorem).



1. By using the *Polygon* tool, construct triangle ABC. Now segments a, b and c will be automatically introduced by GeoGebra Discovery.
2. Type `LocusEquation(a^2+b^2==c^2,C)` in the Input Bar. Now the implicit curve d will be computed and plotted, which seems to be a circle. Note that *two* equal signs must be entered; another possibility is to use $\stackrel{?}{=}$ (by clicking the $\boxed{\alpha}$ icon, or inserting this symbol from an external application by using *Copy* and *Paste*).
3. Try dragging each draggable objects. It can be visually concluded that if C lies on a circle whose diameter is AB, then—because of the right property of the triangle—$a^2 + b^2 = c^2$ indeed follows.

A Boolean expression can be:

- An *algebraic* equation of labels of segments, e.g. `a^2+b^2==c^2`.
- An equality of two geometric objects, e.g. `A==B`. Again, note that *two* equal signs must be entered; other possibilities are to use
  - $\stackrel{?}{=}$ (by clicking the $\boxed{\alpha}$ icon, or inserting this symbol from an external application by using Copy and Paste), or
  - alternatively, `AreEqual(A,B)` for the complete Boolean expression.

- A check if two geometric objects are congruent, e.g. `AreCongruent(c,d)`.
- A check if a point is on a path, for example, on a line or circle, e.g. `A∈c`.
- A check if two lines or segments are parallel, e.g. `p∥q`. Here also `AreParallel(p,q)` can be used.
- A check if two lines or segments are perpendicular, e.g. `p⊥q`. Here also `ArePerpendicular(p,q)` can be used.
- `AreCollinear(A,B,C)` checks if points A, B and C are collinear.
- `AreConcurrent(d,e,f)` checks if lines d, e and f are concurrent.
- `AreConcyclic(A,B,C,D)` checks if points A, B, C and D are concyclic.

Symbols like $\in$, $\|$ and $\perp$ can be inserted by clicking the $\boxed{\alpha}$ icon, or from an external application by using Copy and Paste.

In many cases it may be useful to change the colour and the line thickness of the resulting curves, and to increase their layer number to ensure that other objects do not hide them. These settings can be changed in the Object properties window.

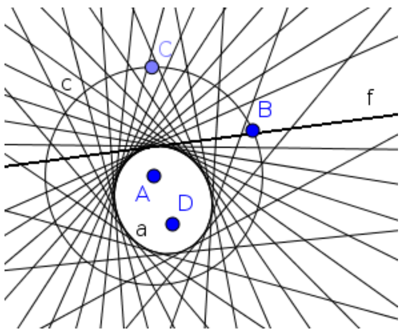
Further information and references can be found in [2].

3.2.3 The Envelope tool and command

This tool and command compute the equation of a curve which is tangent to a family of objects while a certain "parent" of the objects in the family moves on a path [5].

More precisely, given an input point $\mathscr{I}$ on a path $\mathscr{P}$, some construction steps, and an output path $\mathscr{O}$, either a line or a circle, the task is to determine the equation $\mathscr{E}$ of a curve $\mathscr{C}$ which is tangent to $\mathscr{O}$, while $\mathscr{I}$ is moving on $\mathscr{P}$. Then finally plot $\mathscr{E}$. $\mathscr{I}$ is called the mover. $\mathscr{E}$ is called the *envelope equation*, and its graphical visualization is the *envelope*.

Example A well known way to define an ellipse as an envelope of lines is as follows: Given a circle and an internal point of it. The curve which is tangent to the family of the perpendicular bisectors of a moving circumpoint of the circle and its internal point, is an ellipse.



1. By using the *Circle with Center through Point* tool, construct circle c with centre A and circumpoint B.
2. Put point C on c.
3. Create an arbitrary point D inside c.
4. Construct the *Perpendicular Bisector* f of segment CD by using its endpoints.
5. Type `Envelope(f,C)` in the Input Bar. Now the implicit curve a will be computed and plotted—the equation of the envelope is given in the Algebra Window and it is easily seen as the equation of a conic section. In the Geometry Window an ellipse is shown, the graphical representation of the computed algebraic equation.
   - Alternatively, you can select the Envelope tool and click f and then C (or in a different order, this is also allowed).

3.2.4 Technical notes

The following notes describe some situations that might occur when one of the previously described automated reasoning tools in GeoGebra Discovery uses symbolic computations:

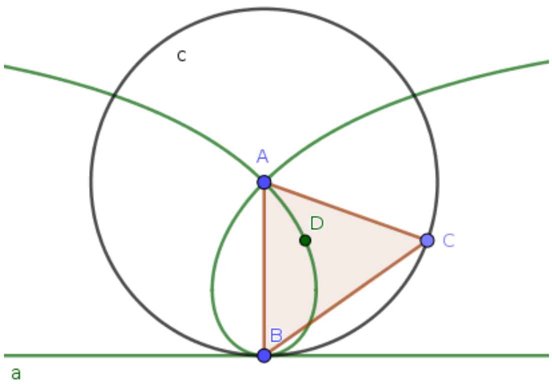
- Not all GeoGebra tools and construction steps are supported.
- The supported tools work only for a restricted set of geometric objects, i.e. using points, lines, circles, conics, and sometimes for implicit algebraic curves, but not for arbitrary curves.
- Rays and line segments will be treated as infinite lines. Circle arcs will be treated as circles.
- Computations of too complicated loci or envelopes may return 'undefined' in the Algebra View, meaning, for example, that the computation has not been achieved within the allowed time limit.
- Relationship proofs which yield too complicated computations will display the message "checked numerically". This must be interpreted as follows: GeoGebra Discovery was unable to decide if the relationship is valid in general, but the numerical results promise optimism. That is, the relationship can be false in general in this case, too (or not!).
- If there is no locus or envelope associated to a construction, then the output yields the empty implicit curve $0 = 1$. Example: for an arbitrary point P

  `LocusEquation(false,P)`

  returns the empty set.
- In some cases, all points of the plane fulfil the input requirements. For instance, the command

  `LocusEquation(true,P)`

  refers to all points P in the plane. In such cases the output of the command is the equation $0 = 0$.
- Sometimes, the output can include extra branches of the curve that are traditionally not considered to belong to the locus or envelope. For example, let points A and B be given, and also a third point C on the circle c with centre A and circumpoint B. Now let us consider the orthocentre D of triangle ABC. Then the command `LocusEquation(D,C)` results in a strophoid curve plus a line—here the line corresponds to a degenerate case of the triangle when $B = C$, but the line is actually not a part of the geometric locus.

  By dragging point C on the circle, one can find that the output contains an extra branch here. In general, to exclude all points that do not play a geometric role, one may need further investigations that are not supported by GeoGebra Discovery ART now. See [5] for some further details.
- The graph of the implicit curve may be inaccurate in some cases. This can be improved by using the `Plot2D` command (see below).

3.3 Discovering properties

In this section we give a short overview on the Discover tool and command, and give an example of stepwise discovery.

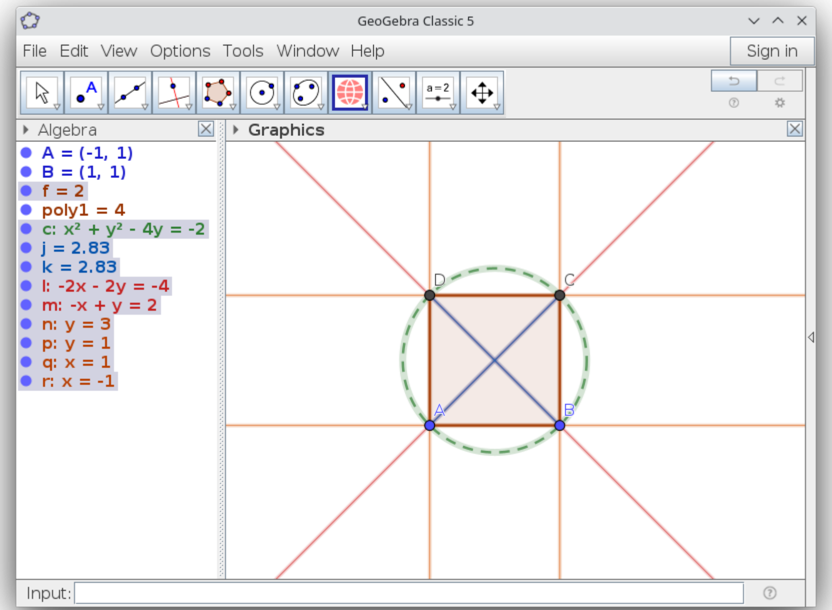
3.3.1 The Discover tool and command

The Discover tool and command detect for geometric relationships between objects that are in true in general. They require an input point which is the target of investigation. Each property is checked first numerically, and then, if the result is positive, a second symbolic test is performed internally via the low level `Prove` command (see below).

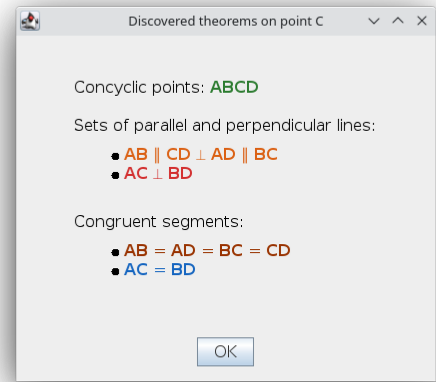
The syntax of the `Discover` command is

`Discover( <Point> ).`

Example We discover some simple properties of a square.



1. By using the *Regular Polygon* tool, construct points A, B, and enter 4 as number of sides. Two more points (C and D) will be created.
2. Type `Discover(C)`. The Graphics View will now include a couple of new objects, including the diagonals of the square, and several objects will be colored with matching colors, accordingly to certain relationships. In addition, a second window will show the relationships as a list:



- Alternatively, select the `Discover` tool and click point C.

3. You may want to deselect the objects that have been automatically marked during the discovery process. It is possible in the *Edit* menu by choosing *Select All* and then *Invert Selection*.

The discovery process searches for overlapping points, collinearity, concyclicity, parallelism and perpendicularity, and congruent segments. If the construction contains several points, the process can be time consuming and therefore a progress bar reports the percentual status. The computation cannot be interrupted in the current versions of GeoGebra Discovery.

3.3.2 The `StepwiseDiscovery` command and the corresponding tool

The `StepwiseDiscovery` command and the corresponding tool invokes the previously mentioned feature on each newly created point.

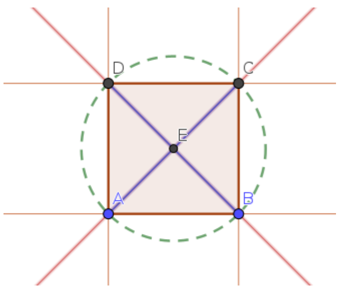
The syntax of the command is

```
StepwiseDiscovery( <Boolean> )
```

where the input sets the stepwise discovery mode on or off. If the mode is requested to be enabled, the input can be omitted.

Example (continued)

4. Type `StepwiseDiscovery(true)` in the Input Bar.
   - Alternatively, you can click the small icon in the Graphics View to enable the stepwise discovery mode. GeoGebra Discovery will inform you on the current working mode in this case.
5. Create the intersection point E of diagonals AC and BD. Now the following output will be shown:



6. Type `StepwiseDiscovery(false)` in the Input Bar.
   - Alternatively, you can click the small icon in the Graphics View to disable the stepwise discovery mode. GeoGebra Discovery will inform you on the current working mode in this case.

3.4 Commands that use cylindrical algebraic demposition

3.4.1 The `Plot2D` command

The `Plot2D` command plots a topologically faithful visual representation of logical connectives of algebraic equations or inequalities. In the background, the input formula (that consists of variables x and y) is analyzed by a subsystem that creates a cylindrical algebraic decomposition (CAD) accordingly. In this sense, the `Plot2D` command can be considered as a proof on the correct topology of the input formula.

The syntax of the command is

```
Plot2D( <Function> )
```

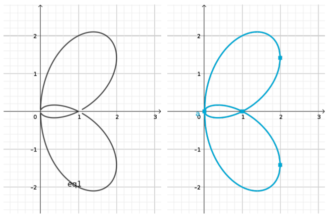
or

Plot2D(<Function>, "x") or Plot2D(<Function>, "y")

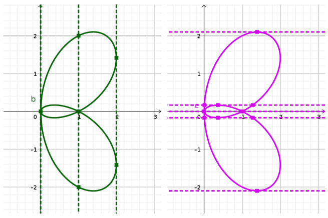
where the variable designates the projection variable of the CAD.

Example Visualization of the tacnode curve $(x^2+y^2-3x)^2-4x^2(2-x)=0$ may be inaccurate if it is plotted directly. By using the `Plot2D` command, it is possible to analyze the curve via a CAD and plot it accurately.

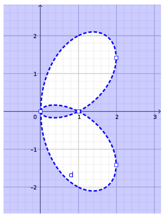
1. Type the equation of the tacnode curve in the Input Bar. An equation labeled with $eq1$ is created.
2. Type `Plot2D(eq1)` in the Input Bar.



3. To get the CADs accordingly to x or y, enter `Plot2D(eq1,"x")` or `Plot2D(eq1,"y")` in the Input Bar.



Note: If GeoGebra's default plotting routine (without CAD) would skip plotting because the input cannot be handled properly, the `Plot2D` command jumps in automatically. For example, changing the equation sign to `>` in the first step, we automatically get the following output:



3.4.2 Comparison of segments in the Relation tool and command

The Relation tool can compare lengths of segments symbolically. This may involve inequalities as well, and therefore it may require CAD computation. The command implicitly invokes the `Compare` low-level command (see below).

Example (Triangle Inequality)

1. Draw an arbitrary triangle ABC with sides a, b and c.
2. Type `Relation(a+b,c)`. GeoGebra Discovery informs the user that "$a+b$ and c are not equal (checked numerically)."
3. Now click "More...". Now the information "It is generally true that $(a+b)>c$ under the condition is not degenerate" is displayed in the window.

3.4.3 The `RealQuantifierElimination` command

The `RealQuantifierElimination` command (available in the *CAS View*) takes the quantified input formula and computes an equivalent formula that does not contain any quantifiers. The computation requires very sophisticated computations that are based on creating a CAD for the input formula. As a result, the user can rely on a black-box computation which ensures that the input and output formulas are equivalent.

The syntax of the command is

```
RealQuantifierElimination( <Expression> )
```

where the input can be an arbitrary quantified expression which contains logical connectives of polynomial equations or inequalities.

Example By entering `RealQuantifierElimination(∃x (a x²+b x+c=0))` in the CAS View, the output

$$4\,a\,c-b^2 \leq 0 \wedge \left(a \neq 0 \vee (c=0) \vee 4\,a\,c-b^2 < 0\right)$$

is shown. Indeed, it no longer contains the existential quantifier.

3.5 Proving statements step-by-step: the `ShowProof` command

By default, GeoGebra Discovery does not focus on presenting a proof of a statement. This is usually avoided to hide the very high number of atomic steps from the user. In some cases, however, it may be important to get a verifyable proof, maybe by manual human checking or for machine verification. The `ShowProof` command gives a step-by-step algebraic proof for the input geometric statement. Its output is shown in the CAS View.

The syntax of the command is

```
ShowProof( <Boolean Expression> )
```

where the input is the statement to prove. At the moment, a presentation of a proof may be unavailable in some cases, for example, if the input is an inequality, or, if the input statement is true on parts (see below). In such cases there may be a black-box proof computed by GeoGebra Discovery, and the result of such a computation can be trustworthy, however, a step-by-step proof cannot be provided.

Example (Thales' circle theorem, continued from page 4)

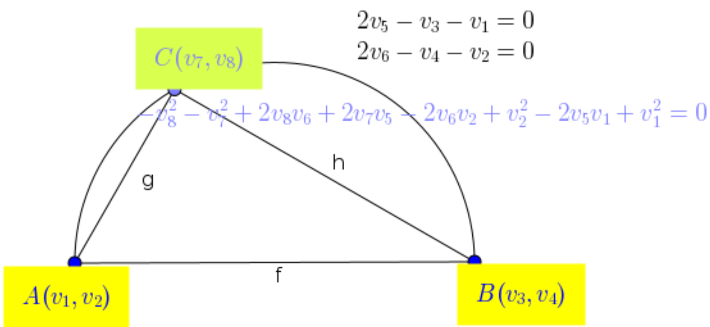
7. Open the CAS View.
8. Enter `ShowProof(f⊥g)` in the Input Bar. Now, after a little time the CAS View will contain the following entries:

1	Let A, B be arbitrary points.
2	Let c be the semicircle through A and B.
3	Let C be a point on c.
4	Let g be the segment A, C.
5	Let h be the segment C, B.
6	Prove that g ⊥ h.
7	The statement is true under some non-degeneracy conditions (see below).
8	We prove this by contradiction.
9	Let free point A be denoted by (v1,v2).
10	Let free point B be denoted by (v3,v4).
11	Object c introduces the following extra variables:
12	v5: x value of center of c
13	v6: y value of center of c
14	e1:=2*v5-v3-v1=0 → $\mathbf{e1: -v1 - v3 + 2\,v5 = 0}$
15	e2:=2*v6-v4-v2=0 → $\mathbf{e2: -v2 - v4 + 2\,v6 = 0}$
16	Considering definition C = Point(c):
17	Let dependent point C be denoted by (v7,v8).
18	e3:=-v8^2-v7^2+2*v8*v6+2*v7*v5-2*v6*v2+v2^2-2*v5*v1+v1^2=0 → $\mathbf{e3: v1^2 + v2^2 - v7^2 - v8^2 - 2\,v1\,v5 - 2\,v2\,v6 + 2\,v5\,v7 + 2\,v6\,v8 = 0}$
19	Thesis: g ⊥ h, in algebraic form:

When scrolling down, the second part of the proofs reads as follows:

20	T1:=-v8^2-v7^2+v8*v4+v7*v3+v8*v2-v4*v2+v7*v1-v3*v1=0 → $\mathbf{T1: -v7^2 - v8^2 - v1\,v3 + v1\,v7 - v2\,v4 + v2\,v8 + v3\,v7 + v4\,v8 = 0}$
21	Thesis reductio ad absurdum (denied statement):
22	v9: dummy variable to express negation
23	(T1*v9-1)=0 → $\mathbf{v9\left(-v7^2 - v8^2 - v1\,v3 + v1\,v7 - v2\,v4 + v2\,v8 + v3\,v7 + v4\,v8\right) - 1 = 0}$
24	e4:=-1-v9*v8^2-v9*v7^2+v9*v8*v4+v9*v7*v3+v9*v8*v2-v9*v4*v2+v9*v7*v1-v9*v3*v1=0 → $\mathbf{e4: -v7^2\,v9 - v8^2\,v9 - v1\,v3\,v9 + v1\,v7\,v9 - v2\,v4\,v9 + v2\,v8\,v9 + v3\,v7\,v9 + v4\,v8\,v9 - 1 = 0}$
25	Without loss of generality, some coordinates can be fixed:
26	{v1=0,v2=0,v3=0,v4=1} → $\mathbf{\{v1 = 0, v2 = 0, v3 = 0, v4 = 1\}}$
27	All hypotheses and the negated thesis after substitutions:
28	s1:2*v5=0 → $\mathbf{s1: 2\,v5 = 0}$
29	s2:-1+2*v6=0 → $\mathbf{s2: 2\,v6 - 1 = 0}$
30	s3:-v8^2-v7^2+2*v8*v6+2*v7*v5=0 → $\mathbf{s3: -v7^2 - v8^2 + 2\,v5\,v7 + 2\,v6\,v8 = 0}$
31	s4:-1+v9*v8-v9*v8^2-v9*v7^2=0 → $\mathbf{s4: -v7^2\,v9 - v8^2\,v9 + v8\,v9 - 1 = 0}$
32	Now we consider the following equation:
33	s1*(-v7*v9)+s2*(-v8*v9)+s3*(v9)+s4*(-1) → $\mathbf{1 = 0}$
34	**Contradiction! This proves the original statement.**
35	The statement has a difficulty of degree 2.

In addition, some objects of the Graphics View will be labeled by the corresponding variables and the equations that appear in the showed algebraic proof.



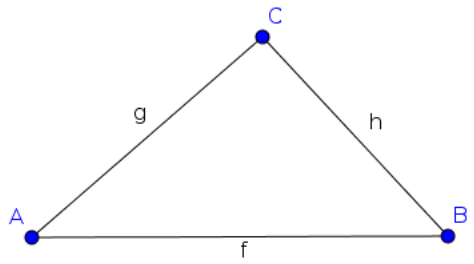
Sometimes the proof includes long lines that may contain many mathematical terms if certain polynomials are of high degree. Currently, the maximal length of a line in the CAS View is limited to 5000 characters for the `ShowProof` command. If the limit is exceeded, it is recommended to export the CAS View to HTML format, or to plain text in Maple, Mathematica or Giac format, for further investigation. The exported file always contains the full output without the limit for the maximal length of a line.

4 Classroom uses: conjecture, proof and generalization

Technically speaking, the easiest symbolic tool is the Relation tool in the list above. On the other hand, some teaching scenarios may require different tools to consider, or more than one tool, but in a different order than the one listed above.

4.1 Thales' circle theorem

In many traditional maths classes, Thales' circle theorem is stated in an explicit form: if C is on a semicircle, the segments g and h are perpendicular (see Section 3.2.1 on page 4). Obviously, the truth of such statement can be easily verified through the Relation and/or Prove commands. In this way, GeoGebra Discovery automated reasoning tools simply act here as a kind of encyclopaedic geometry coach, as a kind of "omniscient" teacher. But, we think it is far more interesting to approach this statement in a quite different way, formulating it as an open-ended question: *Let ABC be an arbitrary triangle. What is the geometric locus of C if the angle at C is to be right?* (See also [1] for a similar approach.)



In this approach it may make more sense to use the technically more difficult `LocusEquation(g⊥h,C)` command first, than finishing the construction and use the Relation tool or command directly. Moreover, the output of the `LocusEquation` command can suggest a conjecture for the students, namely that the locus curve is something like a circle passing through A and B. (The locus is a circle *without* points A and B.) The Algebra View shows the equation of the computed locus, this can be however difficult for younger learners to identify.

Finally, Thales' circle theorem can be generalized towards the theorem of the inscribed angle in a circle, that is, the angle does not change as its vertex is moved to different positions on the circle. In this case the condition is no longer $g \perp h$, but that the angle between them equals to a fixed one. GeoGebra Discovery currently supports entering this kind of investigation with the syntax

`LocusEquation(α==β,C)`

if α is fixed and $\beta = \angle ACB$.

To summarize this approach:

step 1: an implicit locus is computed with GeoGebra Discovery,

step 2: a conjecture for the output curve is made by the student,

step 3: the conjecture is checked by the Relation tool or command in GeoGebra Discovery,

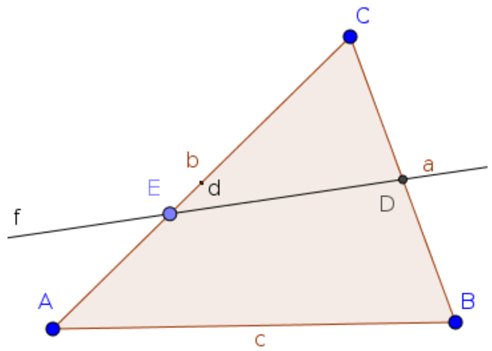
step 4: the proof can be optionally worked out by paper and pencil by the student,

step 5: the theorem can be generalized by plotting further implicit loci with GeoGebra Discovery—as further experiments for the student.

4.2 A worked-out example: The midline theorem

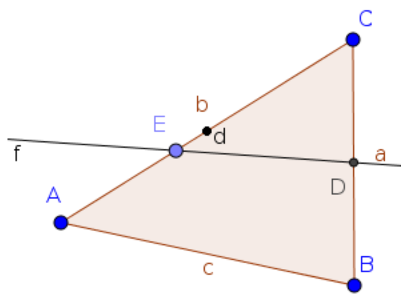
Here step-by-step instructions are provided on a possible way to investigate the midline theorem (stating that the line through the midpoints of two sides of a triangle is parallel to the third side) by using GeoGebra Discovery's automated reasoning tools. The midline theorem states that in a triangle, the segment joining the midpoints of any two sides will be parallel to the third side and half its length. Here we provide step-by-step instructions to formalize this theorem with GeoGebra Discovery.

Step 1



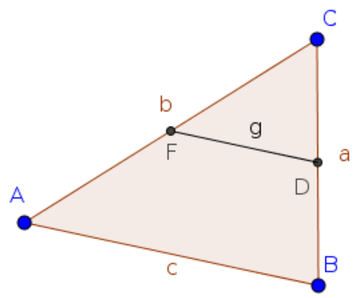
1. By using the *Polygon* tool, construct triangle ABC. This will automatically create segments a, b and c.
2. By using the *Midpoint or Center* tool, create the midpoint D of a.
3. Put point E on b.
4. Create line f which joins D and E.
5. Ask GeoGebra Discovery on the requirement for E in order to have f parallel to c: type `LocusEquation(c||f,E)` in the Input Bar. Now the implicit curve d will be computed and plotted, and it seems to be a single point. Note: it may be useful to change the colour and the line thickness of the implicit curve d, and also to increase its layer number to ensure that other objects do not hide it. Both settings can be changed in the Object properties window.

Step 2



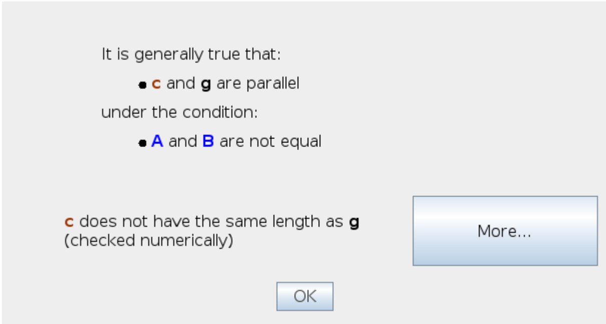
6. Drag the free objects and conjecture that E must be the midpoint of b.
7. To confirm this conjecture, create midpoint F of segment b (and align labels of d and F to avoid overlapping). Drag the free objects again.

Step 3



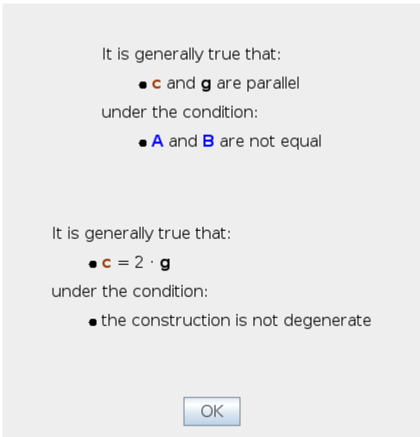
8. Make the objects E, f and d invisible by hiding them.
9. Join D and F by segment g.

10. Use the *Relation* tool to compare c and g. They seem to be parallel.
11. Click the first "More..." in the popup window and check symbolically that they are indeed parallel unless $A = B$ (but this is a degenerate case).



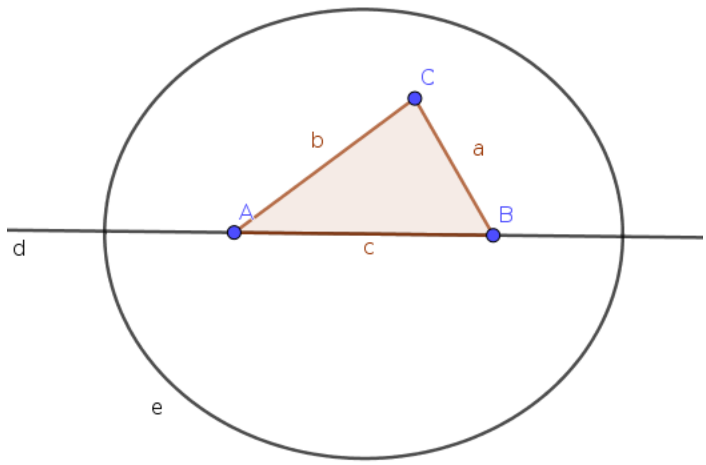
The students may continue with a next step 4, for instance, looking for an elegant way to prove this statement, or just stop here if there is no time for further work in the classroom.

Moreover, a further step 5 could be included after obtaining the classical proof, by considering related questions such as: it is true that c and g do not have the same length—but can g be computed by using the length of c? Maybe $c = 1.5 \cdot g$, or maybe more? By clicking the second "More..." we can learn that $c = 2 \cdot g$, and this can lead to related activities.

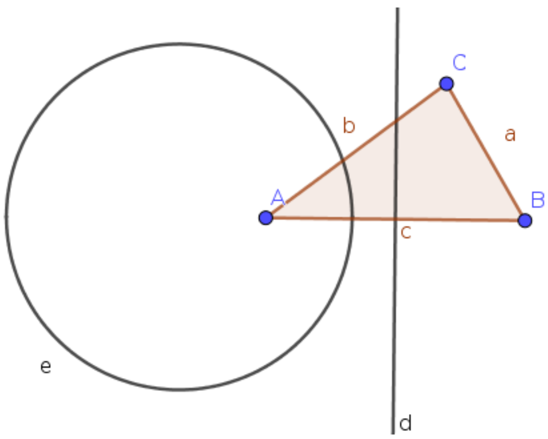


4.3 Further examples

The traditional Triangle Inequality can be translated into an equation, which can be subject to an investigation of degenerated triangles. As a generalization, the synthetic definition of the ellipse can be discovered. Recall the Triangle Inequality, concerning a triangle of sides a, b, c states that $a + b > c$. Now, by using GeoGebra Discovery ART and the command `LocusEquation(a+b==c,C)`, the output will be the line AB, which describes all degenerate triangles. On the other hand, by issuing `LocusEquation(a+b==2c,C)`, an ellipse will be drawn with foci A and B, focal distance $c/2$, semi-major axis c and eccentricity $1/2$. Similar investigations can be performed when using different ratios between $a + b$ and c.



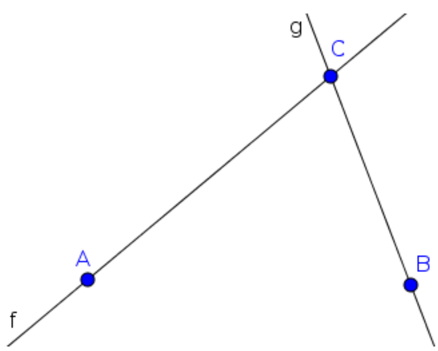
Another application, in a triangle ABC, is to derive the locus equation of C with the condition $a \stackrel{?}{=} b$ (step 1). Clearly, C must lie on the bisector of segment AB (step 2). As by explicitly putting C on the bisector, GeoGebra Discovery confirms that $AC = BC$ when starting the Relation tool's symbolic machinery (step 3). After proving the statement by traditional means (step 4), a generalization can be obtained by typing e.g. `LocusEquation(a==2b,C)`: this can be an interesting experiment for advanced learners too (step 5).



See [11] for additional examples.

5 Limitations: a case study of Thales' circle theorem

Intuitive use of GeoGebra automated reasoning tools may result in unexpected outputs in some cases. This subsection explains some common mistakes during their use, exemplified through an investigation on Thales' circle theorem (see Section 3.2.1 on page 4), as follows:



1. Create points A, B and C.
2. Create lines $f = AC$, $g = BC$.

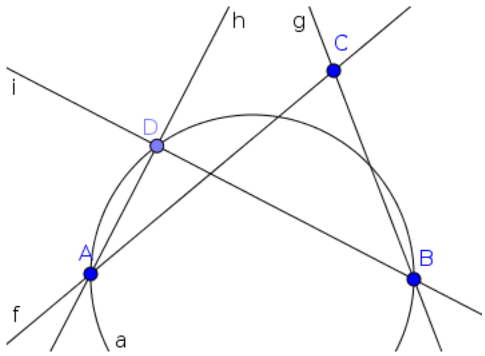
3. Check the result of the command `Relation(f,g)`: "f intersects with g".
4. Ask GeoGebra Discovery about geometric prerequisites of $f \perp g$:

   `LocusEquation(f⊥g,C).`

   An implicit curve a which seems to be a circle will be shown. The equation of the circle is given in the Algebra Window.
5. Move C in the neighbourhood of the implicit curve as close as possible. Now `Relation(f,g)` may still report that "f intersects with g". **Why? Because the point C may be not lying accurately enough on the circle. Depending on the adjusted rounding precision (see the *Options* menu) we might need to exactly state that it is on the circle to get the perpendicularity condition.**

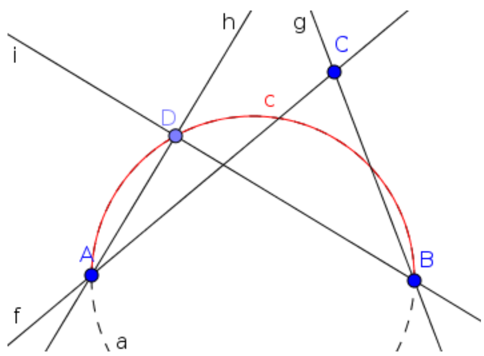
   (a) Try attaching point C on the obtained implicit curve a by using the *Attach / Detach Point* tool. **In fact, this is not allowed in GeoGebra, because by definition, a depends on C, and a circular dependency would occur when attaching C on a (i.e. a will depend on C and C will depend on a), and this would make no sense.**

   (b) Instead, create a new point D by putting it on the implicit curve a. This is allowed in GeoGebra. Create also lines $h = AD$, $i = BD$.



   (c) Check the result of the command `Relation(h,i)`: "h and i are perpendicular" when checked numerically. By clicking "More…" the result is however "checked numerically". **Why? Because GeoGebra interprets the underlying implicit curve as the result of a particular setup of the construction. In other words: in GeoGebra this implicit curve is a numerical object, it does not have a symbolic representation, as the result of a construction in terms of the given free points A, B, C. GeoGebra does not "know" that c is a circle with diameter A, B going through C. That is, symbolic checks based on using an implicit curve as one element of the construction are not possible.**

   (d) The proper way to finalize the steps in this approach is to create the circle with diameter AB with a Circle tool, for example by using the *Semicircle through 2 Points* tool, after detaching D from a and making a invisible. Now D can be attached to the semicircle.



   (Optionally the implicit curve can be set to visible by displaying it with a different style. In this example another style was used for the semicircle as well.) Finally `Relation(h,i)` will now yield the correct outputs, both numerically and symbolically.

6 Appendix

6.1 Low-level GeoGebra Discovery tools

Automated reasoning tools in GeoGebra Discovery are completed by some low-level tools, prepared for learning more, and in a more accurate way, about geometric properties.

6.1.1 The `Compare` command

The `Compare` command is similar to the `Relation` command with the difference that the output is put in a GeoGebra text variable. For instance, in the example of the Triangle Inequality, the command `Compare(a+b,c)` gives the textual output `(a+b)>c` and puts it, by default, in the variable $text1$. At the same time, the textual output is displayed in the Graphics View, by default in its origin.

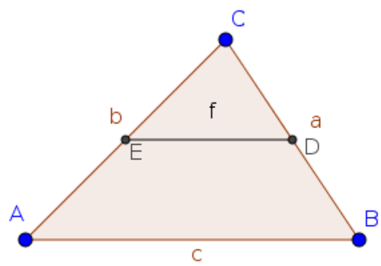
6.1.2 The `Prove` command

The `Prove` command decides if a geometric statement is true in general. It has three possible outputs:

- *true* means that the statement is always true, or true under some non-degeneracy [6, 7, 17] or essential [15] conditions, or true on parts, false on parts [4, 12].
- *false* means that the statement is false in general.
- *undefined* means that GeoGebra Discovery cannot decide because of some reason:
  - The statement cannot be translated into a model which can be further investigated. This usually means that algebraization of the statement failed because of
    - theoretical impossibility (e.g. using a transcendent function as a construction step, for example, sine of x),
    - missing implementation in GeoGebra Discovery.
  - The translated statement in algebraic geometry is too difficult to solve. This means that either there are too many variables, or the equations are hard to handle by the solver algorithm. This results in either a timeout or an out of memory error.
  - The solver algorithm was able to investigate the situation, but the result is ambiguous: either the statement is false, or it is true under certain conditions—but the algorithm was not able to decide which case is present.
  - There was an internal error in GeoGebra Discovery during the computations.

The `Prove` command may involve CAD operations if the default model (in algebraic geometry) cannot give a satisfactory answer.

Example In a triangle a segment joining the midpoints of two sides is parallel to the third side and half its length.



1. Construct the triangle *ABC* by using the *Polygon* tool.
2. Construct the midpoints D and E of sides a and b, respectively, by using the *Midpoint or Center* tool.
3. By using the *Segment* tool, create f by joining D and E.
4. Type `Prove(f||c)` to obtain *true* in the Algebra View as Boolean Value d. Note that the parallel sign must be entered by using either

- the list of the mathematical symbols by clicking the $\boxed{\alpha}$ icon in the Input Bar, or
- inserting this symbol externally by using Copy and Paste.
- Alternatively, `f||c` can be substituted by `AreParallel(f,c)` also.

5. Type `Prove(c==3f)`. Now the answer is *false*. Note that *two* equal signs must be entered; other possibilities are to use
   - `Prove(c`$\stackrel{?}{=}$`3f)`, or
   - `Prove(AreEqual(c,3f))`.

6.1.3 The `ProveDetails` command

The `ProveDetails` command has as similar behaviour as the `Prove` command and uses the same algorithms in the decision process, but it may provide more information on the results. It has three possible outputs:

- $\{$*true*$\}$ means that the statement is always true.
- $\{$*true,* $\{\ldots\}\}$ if the statement is true under some non-degeneracy [6, 17] or essential [15] conditions, or true on parts, false on parts [4, 12]: the denied form of these conditions are listed in the internal braces. (If the list remains "...", it means that no synthetic translation could be found.) If the conjunction of the negated conditions holds, then the statement should be true.
- $\{$*false*$\}$ means that the statement is false in general.

Example (continued)

6. Type `ProveDetails(c==3f)`. Now the answer is $\{$*false*$\}$.
7. Type `ProveDetails(c==2f)`. Now the answer is $\{$*true*$\}$.
8. Type `ProveDetails(c<3f)`. Now the answer is $\{$*true,*$\{$*"AreCollinear(A,B,C)","AreEqual(A,B)"*$\}\}$. This means that the statement $c < 3f$ is true if A, B and C are not collinear and $A \neq B$. The converse is not necessarily true: If A, B and C are collinear but $A \neq B$, the statement $c < 3f$ still holds. On the other hand, if $A = B$, $c = f = 0$ and therefore the inequality $c < 3f$ is false.

Note that GeoGebra Discovery tries to find out the conditions with a mixed strategy. Some parts of the conditions are added by obtaining possible non-degeneracy conditions algebraically and then the geometrically "simplest" one is selected, if there is such a condition—if not, GeoGebra Discovery adds "..." to the list of conditions. Also, some conditions may be forced in advance, before starting the computation, these usually include the assumption that the first two free points (in many cases, denoted by A and B) differ, or each three of the free points designate a non-degerate triangle.

6.1.4 A comparison of `Prove`, `ProveDetails` and `Relation`

The following table explains in a concise way the meanings of the outputs of the commands `Prove`, `ProveDetails` and `Relation`. The outputs of these three commands should never be contradictory but complimentary. For most users, however, the use of the `Relation` command is suggested.

The `Relation` window usually reports the results in a more geometrically readable form than the `ProveDetails` command, but with an equivalent meaning.

For further details see [10, 3], the current state of the prover has been, however, already improved since then.

GeoGebra Discovery outputs			Conclusion
Prove	**ProveDetails**	**Relation**'s symbolic window	
true	{true}	always true	The statement is true.
	{true,{*conditions*}}	generally true under *conjunction of the negations of the specified conditions*	The statement is true if none of the specified *conditions* hold. These negated conditions are sufficient, but maybe not necessary. There may be other sufficient conditions to make the statement true.
	{true,{"..."}}	generally true if non-degenerate	The statement is true if certain equations hold. These equations have no visually clear geometric meanings for GeoGebra Discovery.
	{true,{*conditions*},"c"}	true on parts, false on parts under *conjunction of the negations of the specified conditions*	The statement is true on parts, false on parts if none of the specified *conditions* hold. These negated conditions are sufficient, but maybe not necessary. There may be other sufficient conditions to make the statement true.
false	{false}	false in general	The statement is false.
undefined	{}	checked numerically	GeoGebra Discovery was unable to decide if the statement is true or false. The numerical check confirms the truth, but the symbolic check was unsuccessful due to computational difficulties, or the symbolic check for the given statement is not yet implemented.

6.2 Translation of GeoGebra Discovery commands

The names of GeoGebra Discovery automated reasoning tools may need to be translated to other languages. For example, the German translation of `Prove` can be `Prüfe`. To learn the translated command names the following steps are recommended:

1. Create a GeoGebra file which contains the required commands in the Algebra View.
2. Change the language in GeoGebra Discovery in the Options menu by choosing *Language*.
3. The command names will be automatically changed in the Algebra View.
4. Move the mouse over a command in the Algebra View and read its translated name off.

6.3 Debugging

Starting GeoGebra Discovery via command line there are more possibilities to investigate the results. Here the method on a typical Linux installation is demonstrated. We assume that a Snapcraft version is installed from `https://snapcraft.io/geogebra-discovery`.

The user needs to start GeoGebra Discovery by the following command:

```
geogebra-discovery --logfile=/dev/stdout --logshowcaller=false \
    --logshowtime=false --logshowlevel=false
```

A typical output looks like as follows:

```
Using AUTO
Using BOTANAS_PROVER
A = (3.42, 1.86) /* free point */
// Free point A(v1,v2)
B = (10.48, 3.1) /* free point */
// Free point B(v3,v4)
f = Segment[A, B] /* Segment [A, B] */
C = Point[f] /* Point on f */
// Constrained point C(v5,v6)
Hypotheses:
1. -v5*v4+v6*v3+v5*v2-v3*v2-v6*v1+v4*v1
```

```
g = Segment[A, C] /* Segment [A, C] */
h = Segment[C, B] /* Segment [C, B] */
Processing numerical object
Hypotheses have been processed.
giac evalRaw input: evalfa(expand(ggbtmpvarf))
giac evalRaw output: ggbtmpvarf
input = expand(ggbtmpvarf)
result = ggbtmpvarf
eliminate([ggbtmpvarf-((ggbtmpvarg)+(ggbtmpvarh))=0,ggbtmpvarh^2=v11^2,ggbtmpvarg^2=v12^2,ggbtmpvarf^2=v13^2],[ggbtmpvarh
     ,ggbtmpvarg,ggbtmpvarf])
giac evalRaw input: evalfa(eliminate([ggbtmpvarf-((ggbtmpvarg)+(ggbtmpvarh))=0,ggbtmpvarh^2=v11^2,ggbtmpvarg^2=v12^2,
     ggbtmpvarf^2=v13^2],[ggbtmpvarh,ggbtmpvarg,ggbtmpvarf]))
// Groebner basis computation time 0.000448 Memory -1e-06M
giac evalRaw output: {v11^4-2*v11^2*v12^2+v12^4-2*v11^2*v13^2-2*v12^2*v13^2+v13^4}
input = eliminate([ggbtmpvarf-((ggbtmpvarg)+(ggbtmpvarh))=0,ggbtmpvarh^2=v11^2,ggbtmpvarg^2=v12^2,ggbtmpvarf^2=v13^2],[
     ggbtmpvarh,ggbtmpvarg,ggbtmpvarf])
result = {v11^4-2*v11^2*v12^2+v12^4-2*v11^2*v13^2-2*v12^2*v13^2+v13^4}
giac evalRaw input: evalfa(eliminate([ggbtmpvarf-((ggbtmpvarg)+(ggbtmpvarh))=0,ggbtmpvarh=v11,ggbtmpvarg=v12,ggbtmpvarf=
     v13],[ggbtmpvarh,ggbtmpvarg,ggbtmpvarf]))
// Groebner basis computation time 0.000592 Memory -1e-06M
giac evalRaw output: {v11+v12-v13}
input = eliminate([ggbtmpvarf-((ggbtmpvarg)+(ggbtmpvarh))=0,ggbtmpvarh=v11,ggbtmpvarg=v12,ggbtmpvarf=v13],[ggbtmpvarh,
     ggbtmpvarg,ggbtmpvarf])
result = {v11+v12-v13}
giac evalRaw input: evalfa(simplify({v11^4-2*v11^2*v12^2+v12^4-2*v11^2*v13^2-2*v12^2*v13^2+v13^4}/{v11+v12-v13}))
giac evalRaw output: {v11^3-v11^2*v12+v11^2*v13-v11*v12^2-2*v11*v12*v13-v11*v13^2+v12^3+v12^2*v13-v12*v13^2-v13^3}
input = simplify({v11^4-2*v11^2*v12^2+v12^4-2*v11^2*v13^2-2*v12^2*v13^2+v13^4}/{v11+v12-v13})
result = {v11^3-v11^2*v12+v11^2*v13-v11*v12^2-2*v11*v12*v13-v11*v13^2+v12^3+v12^2*v13-v12*v13^2-v13^3}
giac evalRaw input: evalfa(factor(v11^3-v11^2*v12+v11^2*v13-v11*v12^2-2*v11*v12*v13-v11*v13^2+v12^3+v12^2*v13-v12*v13^2-
     v13^3))
giac evalRaw output: (v11-v12-v13)*(v11-v12+v13)*(v11+v12+v13)
input = factor(v11^3-v11^2*v12+v11^2*v13-v11*v12^2-2*v11*v12*v13-v11*v13^2+v12^3+v12^2*v13-v12*v13^2-v13^3)
result = (v11-v12-v13)*(v11-v12+v13)*(v11+v12+v13)
Trying to detect polynomial -v13-v12+v11
-v13-v12+v11 means h = f + g
Trying to detect polynomial v13-v12+v11
v13-v12+v11 means f + h = g
Trying to detect polynomial v13+v12+v11
v13+v12+v11 means f + g + h = 0, uninteresting
Thesis equations (non-denied ones):
2. v11^2-v6^2-v5^2+2*v6*v4-v4^2+2*v5*v3-v3^2
3. v12^2-v6^2-v5^2+2*v6*v2-v2^2+2*v5*v1-v1^2
4. v13^2-v4^2-v3^2+2*v4*v2-v2^2+2*v3*v1-v1^2
Thesis reductio ad absurdum (denied statement), product of factors:
(v13^4-2*v13^2*v12^2+v12^4-2*v13^2*v11^2-2*v12^2*v11^2+v11^4)*v14-1
that is,
5. -1+v14*v13^4-2*v14*v13^2*v12^2+v14*v12^4-2*v14*v13^2*v11^2-2*v14*v12^2*v11^2+v14*v11^4
substitutions: {v1=0, v2=0}
Eliminating system in 8 variables (5 dependent)
giac evalRaw input: evalfa([[ff:=\"\"],[aa:=eliminate2([v12^2-v6^2-v5^2,v11^2-v6^2-v5^2+2*v6*v4-v4^2+2*v5*v3-v3^2,-1+v14*
     v13^4-2*v14*v13^2*v12^2+v14*v12^4-2*v14*v13^2*v11^2-2*v14*v12^2*v11^2+v14*v11^4,v13^2-v4^2-v3^2,-v5*v4+v6*v3],
     revlist([v6,v11,v12,v13,v14]))],[bb:=size(aa)],[for ii from 0 to bb-1 do ff+=(\"[\"+(ii+1)+\"]: [1]:
     unicode95uunicode91u1]=1\");cc:=factors(aa[ii]);dd:=size(cc);for jj from 0 to dd-1 by 2 do ff+=(\"
     unicode95uunicode91u\"+(jj/2+2)+\"]=\"+cc[jj]); od; ff+=(\" [2]: \"+cc[1]);for kk from 1 to dd-1 by 2 do ff+=(\",\"+
     cc[kk]);od;od],[if(ff==\"\") begin ff:=[0] end],ff][5])
// Groebner basis computation time 0.000249 Memory -1e-06M
giac evalRaw output: "[1]: [1]:  unicode95uunicode91u1]=1  unicode95uunicode91u2]=1 [2]: 1,1"
Considering NDG 1...
Found a better NDG score (0.0) than Infinity
Statement is GENERALLY TRUE
Benchmarking: 38 ms
STATEMENT IS TRUE (yes/no: TRUE)
OUTPUT for ProveDetails: null = {true, {"f + h = g", "h = f + g"}}
```

There is intentionally no easier way to show the users this type of output. However, the last few lines of the debug information are available in GeoGebra Discovery in the *Help* menu, by choosing *About/License*, and clicking *System Information*—this copies the latest debug messages into the clipboard.

Acknowledgments

Authors would like to express thier gratitude to Prof. T. Dana-Picard, for his careful reading and many suggestions to improve earlier versions of this manuscript. Work partially supported by Spanish Research Project MTM2017-88796-P (for a previous version of this tutorial) and “Mathematical Visualization: Foundations, Algorithms and Applications”, PID2020-113192GB-I00 from the Spanish MICINN.

Biographical notes

- *Zoltán Kovács* (Jászberény, Hungary, October 31, 1975). M.Sc. (1999), József Attila University of Szeged, Ph.D. (2015), Johannes Kepler University of Linz. Assistant professor at The Private University College of Education of the Diocese of Linz, Institute of Initial Teacher Training (Austria), since 2015. Team member of Center of Mathematics Education of Linz at University of Linz, Department of Mathematics Education. Volunteer core developer at GeoGebra, since 2010. Author of over hundred published scientific papers and several scientific communications (including various software packages) in different international journals and conferences. Topics: Mathematics Education, Automated Reasoning in Dynamic Geometry, Computer Algebra. `https://matek.hu/zoltan`.

  Postal address: Private Pädagogische Hochschule der Diözese Linz. Salesianumweg 3, 4020 Linz, Austria.

- *Tomás Recio* (Oviedo, Spain, December 14, 1949). B.Sc., M.Sc. (1972), Ph.D. (1976), Universidad Complutense de Madrid. Professor of Algebra at the Universidad de Cantabria (Santander, Spain), since 1982. Founder and Chair of the Institute GeoGebra of Cantabria. Author of over hundred fifty published scientific papers and three hundred fifty scientific communications in different international journals and conferences. Topics: Real Algebraic Geometry, CAD, Robotics, Computer Algebra and Geometry, Automatic Reasoning in Dynamic Geometry, Mathematics Education. `http://www.recio.tk`.

  Postal address: Departamento de Matemáticas, Estadística y Computación. Facultad de Ciencias, Universidad de Cantabria. Avenida de los Castros, s/n, 39071 Santander, Spain.

- *M. Pilar Vélez* (Logroño, Spain, June 3, 1967). B.Sc., M.Sc. (1990), Ph.D. (1995), Universidad Complutense de Madrid. Professor of Applied Mathematics at the Universidad Antonio de Nebrija (Madrid, Spain), since 1997. Rector of Universidad Antonio de Nebrija from 2010 to 2014. Author of several scientific papers and scientific communications in different international journals and conferences. Topics: Real Algebraic Geometry, Computer Algebra, Automatic Reasoning in Dynamic Geometry, Mathematics Education. `https://orcid.org/0000-0002-5724-4300`.

  Postal address: Departamento de Ingeniería Industrial, Escuela Politécnica Superior, Universidad Antonio de Nebrija. c/ Pirineos, 55, 28040 Madrid, Spain.

References

[1] Artigue, M. What is inquiry-based mathematics education (IBME)? In: Artigue, M. and Baptist, P. (Eds.): *Inquiry in mathematics education. Fibonacci project*, p. 3–13. 2012.

[2] Abánades, M.A., Botana, F., Kovács, Z., Recio, T. and Sólyom-Gecse, C.: Development of automatic reasoning tools in GeoGebra. ACM Communications in Computer Algebra 50(3), p. 85–88. 2016.

[3] Botana, F., Hohenwarter, M., Janičić, Kovács, Z., Petrović, I., Recio, T. and Weitzhofer, S.: Automated Theorem Proving in GeoGebra: Current Achievements. Journal of Automated Reasoning 55.1, p. 39–59. 2015.

[4] Botana, F. and Recio, T.: On the Unavoidable Uncertainty of Truth in Dynamic Geometry Proving. Mathematics in Computer Science 10(1), p. 5–25. 2016.

[5] Botana, F. and Recio, T.: Computing envelopes in dynamic geometry environments. Annals of Mathematics and Artificial Intelligence 80(1), p. 3–20. 2017.

[6] Chou, S.-C.: *Mechanical geometry theorem proving.* Kluwer Academic Publishers, Norwell, MA, USA. 1987.

[7] Cox, D.A., Little, J., O'Shea, D.: *Ideals, varieties, and algorithms. An introduction computational algebraic geometry and commutative algebra.* 4th revised ed. Undergraduate Texts in Mathematics, Springer, Cham. 2015.

[8] Davis, P.: The rise, fall, and possible transfiguration of triangle geometry: a mini-history. The American Mathematical Monthly 102(3), p. 204–214. 1995.

[9] Howson, G. and Wilson, B.: *ICMI Study series: School mathematics in the 1990's.* Kuwait: Cambridge University Press. 1986.

[10] Kovács, Z.: *Computer Based Conjectures and Proofs in Teaching Euclidean Geometry*. PhD thesis. Johannes Kepler University, Linz. 2015.

[11] Kovács, Z.: Automated reasoning Tools in GeoGebra: A new approach for experiments in planar geometry. South Bohemia Mathematical Letters 25(1), p. 48–65. `http://home.pf.jcu.cz/~sbml/wp-content/uploads/Kovacs.pdf`. 2017.

[12] Kovács, Z., Recio, T. and Vélez, M.P.: Detecting truth on components. arXiv:1802.05875 [cs.AI]. 2018.

[13] Kovács, Z., Recio, T. and Vélez, M.P.: Using Automated Reasoning Tools in GeoGebra in the Teaching and Learning of Proving in Geometry. International Journal for Technology in Mathematics Education 25(2), p. 33–50. 2018.

[14] Kovács, Z., Recio, T., Richard, P.R. and Vélez, M.P.: GeoGebra Automated Reasoning Tools: A Tutorial with Examples. In: Aldon, G. and Trgalova, J. (Eds.): *Proceedings of the 13th International Conference on Technology in Mathematics Teaching*. `https://hal.archives-ouvertes.fr/hal-01632970`. 2017.

[15] Kovács, Z., Recio, T. and Sólyom-Gecse, C.: Automatic Rewrites of Input Expressions in Complex Algebraic Geometry Provers. In: Narboux, J., Schreck, P. and Streinu, E. (Eds.): *Proceedings of ADG 2016*, p. 137–143. 2016.

[16] Quaresma, P.: Towards an Intelligent and Dynamic Geometry Book. Mathematics in Computer Science 11, p. 427–437. 2017.

[17] Recio, T. and Vélez, M.P.: Automatic discovery of theorems in elementary geometry. Journal of Automated Reasoning, 23, p. 63–82. 1999.

[18] Richard, P.R., Oller, A.M. and Meavilla, V.: The Concept of Proof in the Light of Mathematical Work. ZDM – The International Journal on Mathematics Education, 48(6), p. 843–859. 2016.

[19] Sinclair, N., Bartolini Bussi, M.G., de Villiers, M. and Owens, K.: Recent research on geometry education: an ICME-13 survey team report. ZDM Mathematics Education, 48(5), p. 691–719. 2016.