Visualizing Geometrical Statements with GeoView

Yves Bertot [1,2]

Lemme Team
INRIA
Sophia Antipolis, FRANCE

Frédérique Guilhot [3]

Lemme Team
INRIA
Sophia Antipolis, FRANCE

Loïc Pottier [4]

Lemme Team
INRIA
Sophia Antipolis, FRANCE

Abstract

We describe a tool that combines a general purpose theorem prover and an off-the-shelf interface for dynamic geometry drawing to enhance man-machine interaction involving geometrical proofs. With our tool, we can edit the statements of geometrical theorems, construct and verify their proofs with the theorem prover, and visualize the statements using the drawing tool. The key component is an algorithm that computes the data needed to draw a construction from the formal statement of the theorem. The paper includes some examples of output from our combined tool, called GeoView.

Key words: theorem proving, geometry, dynamic geometry drawing.

[1] Thanks to Frédéric Kotecki, the author of GeoplanJ for his collaboration
[2] Email: `Yves.Bertot@sophia.inria.fr`
[3] Email: `Frederique.Guilhot@sophia.inria.fr`
[4] Email: `Loic.Pottier@sophia.inria.fr`

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

1 Introduction

General purpose theorem provers based on higher-order logic usually provide the possibility to develop proof interactively. Because of the interactive aspects, users need to develop competences in logic and reasoning. It is natural to think that this requirement makes this kind of theorem provers a good tool to help students learn mathematics.

To study this use of interactive theorem provers in the domain of geometry, we have developed a library corresponding to the courses available in the French high-school system. These courses encompass bi-dimensional euclidean geometry, with triangles, circles, basic transformations such as translations, rotations, homotheties and tri-dimensional geometry with parallelism, incidence and orthogonality problems between lines and planes. We have chosen to avoid analytic geometry, where reasoning becomes less important than computing.

All this work was performed using a graphical user-interface that made it possible to reconcile the requirements for a formal language (coming from a theorem prover) and the usual mathematical notations (more adapted for communication between humans, like teachers and students). In this context, it soon became clear that "a good drawing is better than a long explanation". We designed an extension of the graphical interface, such that drawings are automatically associated to mathematical formulas to show the meaning of these statements. Our main design decision was to reuse existing drawing tools. We have concentrated on tools already available to French math teachers, especially GeoplanJ [9], mainly because it is available with a GNU GPL license. Our result combined tool is called GeoView [12].

This paper is organized as follows: in the second section, we describe the proof development tool that we use and the theory of geometrical facts that we have developed in this environment; in the third section, we summarize the functionalities of the independent drawing tool we integrated in the proof environment. The fourth section describes the crux of our contribution: an ordering algorithm to transform logical statements into figure construction sequences. The fifth section illustrates our results with a collection of significative examples. The sixth section studies related work and brings a few concluding remarks.

2 Pcoq and the geometry library

Pcoq is the proof development environment [4] we develop as a front-end to the Coq system [18].

2.1 The Pcoq proof environment

Pcoq [1] is a user-interface for the general purpose theorem prover Coq that manipulates all formulas and commands as tree-like structures (also known as abstract syntax trees) rather than plain text. This characteristic makes it easy to provide facilities that are hard to obtain in environments where the structure is not given. For instance, Pcoq makes it easy to attach special mathematical notations to some functions, thus making possible special notations for vectors, parallel and perpendicular predicates, and angles. Examples of our mathematical notation will appear naturally in the figures below. An easy access to the structure of commands also makes it easy to attach sentences in natural language to geometrical statements. In figure 1, we can see a statement with the Coq default syntax and the same statement as it is displayed with Pcoq.

```
Lemma isosceles_median_bisector:
 (A, B, C, I : PO)
  ~ (A == I) ->
  ~ (B == C) ->
 (I == (midpoint B C)) ->
 (isosceles A B C) ->
 (cons_AV (vec A B) (vec A I)) == (cons_AV (vec A I) (vec A C)).
```

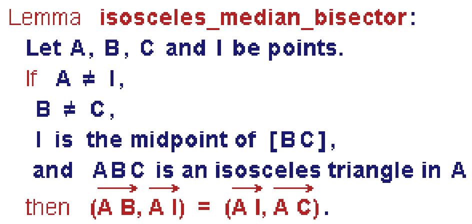


Fig. 1. Example of notations and sentences in natural language in Pcoq

This natural language presentation is mainly used for output, it cannot be used to input text (this would require natural language parsing) but sentence templates are proposed to the user for mouse directed input.

As published in previous works [1,15] natural language can also be used to give a readable form to the proofs built and verified by Coq. Combined with proof-by-pointing, all this gives a proof environment with good support for math teachers and pupils.

Using tree structures also plays a significant role in GeoView. The trees corresponding to theorem or conjecture statements can easily be analyzed to generate figure texts that are transferred to GeoplanJ (described in section 3) for drawing and displaying. There remains some work to connect the statements to proper figure constructions, this work is described in section 4.

2.2 Our Geometry library for Coq

We have used Pcoq to develop a library of geometry theorems that encompasses the course as it is taught in French high-schools. This library is based on a large collection of axioms that is not chosen to be minimal but rather to correspond to the level of details that students are expected to understand and master.

Students are not supposed to know about algebraic notions as vector spaces, affine spaces, or metrics, but in spite of this, they are supposed to get acquainted with vector manipulations such adding, scalar multiplication, scalar product and so on. The general notion of barycentric computation cannot be used systematically, but the notion of barycenter is available.

Our axiomatization for affine space is very related to Marcel Berger's description of barycenters and the universal space in his book [3]. In our library, most operations are not given by a definition, but by the assumption that some function exists and satisfies one or two axioms.

We only enumerate here the different basic notions of plane geometry we can represent with a two-dimensional drawing:

- vectors, alignment, barycenter, midpoint, centroid, parallelism,
- orthogonality, orthocenter, orthogonal projection,
- euclidean distance, isosceles triangle, perpendicular bisector, circle, line tangent to a circle, tangent circles,
- homothety, translation, reflection, rotation, direct similarity and composition of these transformations, inversion.
- complex numbers.

We proved some classical theorems in plane geometry, in the same way as they are proved in the high-school courses. Among them, we can cite: Thales, Menelaus, Ceva, Desargues, Pythagoras, Simson's line, Miquel, Euler's line, nine-point circle.

In this library, there are also chapters about trigonometry and three-dimensional geometry that we have not yet connected to a drawing tool.

3 GeoplanJ

GeoplanJ [9] is a dynamic geometry drawing tool developed at CNAM. In this section, we describe its normal behavior.

3.1 The objects manipulated in GeoplanJ

The user creates mathematical objects in the plane equipped with a predefined but invisible coordinate system. Objects can belong to several classes. Some are *drawable*: points, straight or curved lines, and others are *undrawable*: real numbers, vectors, homotheties, translations, etc. Some objects are *free*, like arbitrary points and others are *bound*, like lines defined to pass through other points, points belonging to a line, midpoints, etc. Some objects are *fixed* in the sense that their description in the invisible coordinate system is completely given and some objects are *variable* in the sense that their position can still change.

All objects have a *value* describing their position in the actual drawing. The variable objects may move as a result from interaction by the user, usually when he grabs a free point by pointing with the mouse and moves it by dragging. Thus, we can obtain several drawings for the same figure, corresponding to different values for the variable objects.

In figure 2, we show several drawings corresponding to the figure where A, B and C are free points, D is a free point on the line passing through C and parallel to line (AB), I and J are the midpoints of segments $[AB]$ and $[CD]$ respectively and the three lines (AC), (BD) ,and (IJ) are drawn.

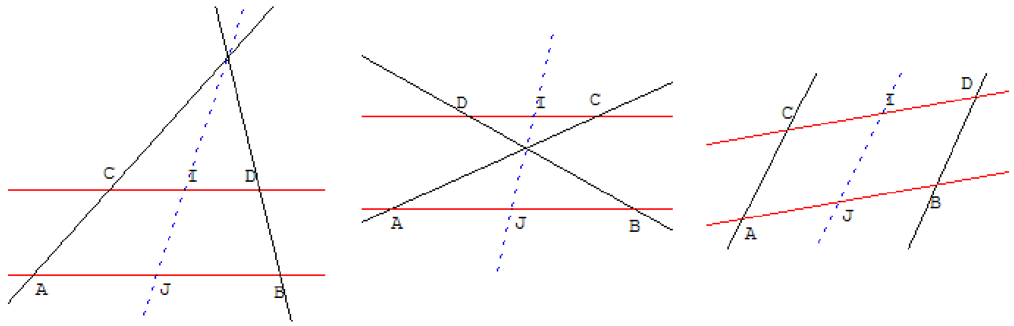


Fig. 2. Several drawings for the same figure

The distinction between variable and free objects is meaningful: some objects may be variable and bound at the same time. For instance, a point may be bound to lie on a straight line but still be allowed to vary its position on the line, as D in figure 2.

A GeoplanJ figure is a collection of descriptions of points, including the constraints that bind them together. Every figure can be saved as a text where all objects are enumerated. It can also contain extra information, like color, line style, names. It is possible to ask GeoplanJ to draw a figure simply by giving it the text describing it.

3.2 Integrating the GeoplanJ software

GeoplanJ is a partial Java port of the software GeoplanW [11], it is initially intended for use as an applet, but since it is provided as free software under a very liberal license, we have been able to use it and modify it for our purposes. The main part of our proof development tool is also programmed in the Java language, so that including GeoplanJ in our software was an easy matter.

The GeoplanJ uses exactly the same textual format as GeoplanW to describe the figures, so that the figures generated in our experiment can be re-used and modified using the two pieces of software.

4 GeoView

Some predicates occurring in geometrical statements, like *collinear*, *cocyclic* and *isosceles* are multi-directional: when these predicates take $n+1$ arguments, given n of these it is possible to determine a constraint on the last one. However, some work needs to be done to decide in which order these predicates are used to build the figure associated to a statement. This work is done by the algorithm we present in this section.

4.1 Data analysis

The data corresponding to a theorem statement given as input is transformed in a tree of type: $H_1 \rightarrow \ldots \rightarrow H_n \rightarrow C$ where H_i can be interpreted as premises of the theorem and C as its conclusion.

Then we distinguish between *binding* and *non-binding* geometrical constraints: for instance, the constraint (collinear ABC) is a binding one but (not(collinear ABC)) is non-binding: in general any three points in the plane are convenient. The constraints which appear in the conclusion are redundant for construction: they are non-binding.

We list all the binding constraints and the points which appear in them. From this list, an algorithm detailed in 4.2 determines a figure text with GeoplanJ syntax.

4.2 Constructing a point set from a geometrical constraints set

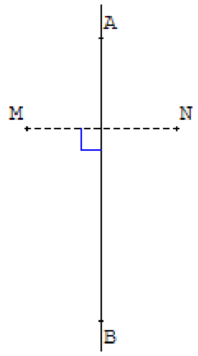
Given a finite set of points in the real plane, and a finite set of geometrical constraints on these points, we describe a simple algorithm that may produce a construction of these points, suitable to draw a picture of these points satisfying the geometrical constraints.

Geometrical constraints

A *geometrical constraint* can be formulated as a set of polynomial equalities on the coordinates of points: saying that points A,B, C are collinear is equivalent to one polynomial equality: $(x_C - x_A)(y_B - y_A) = (y_C - y_A)(x_B - x_A)$. A geometrical constraint is written as $(CP_1 \ldots P_n)$, where C is the constraint and $P_1 \ldots P_n$ are the points.

We will mainly restrict ourselves to constraints such that, for some choice of $n-1$ points among $P_1 \ldots P_n$, we can build the remaining point by simple geometric constructions from the others. For instance, for the constraint (collinear ABC), we can choose two points freely, the third being constrained to lie on the straight line defined by the two others.

To a constraint C we associate a list $T(C)$ which describes the degree of freedom of the points depending on their position in the constraint. If the point P_i is completely determined from the others, the constraint has the type 2 in position i (i.e. the two coordinates of the point are determined). If P_i can vary on a curve (for instance a circle or a straight line), the constraint is of type 1 in position i. If there are configurations of the points P_j ($j \neq i$) such that the constraint $(CP_1 \ldots P_n)$ cannot be satisfied, then we cannot choose freely the points P_j $(j \neq i)$ and we say that the constraint has the type 0 in position i.



For instance, for the constraint (reflection $ABMN$) which means that N is the image point of M by reflection with respect to line (AB), the point N is completely determined by the position of the three other points A, B and M, we say that the constraint is of type 2 in position 4.
But to build B, we cannot choose freely the three points M, N and A because A has to lie on the perpendicular bisector of the segment $[MN]$, we say that the constraint is of type 0 in position 2.

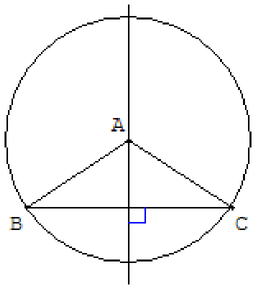
Here are two examples:

(i) For the constraint (collinear ABC) which means that A, B and C are collinear, we have $T(\text{collinear}) = (1, 1, 1)$.

(ii) For the constraint (reflection $ABMN$) which means that N is the image point of M by reflection with respect to line (AB), we have $T(\text{reflection}) = (0, 0, 2, 2)$.

Constructing points

To build a point using a constraint from the other points, we need to consider its position in the constraint, and we may have to build intermediate objects.



For instance, in the constraint (isosceles ABC) which means $AB = AC$, to build the point A, we build the perpendicular bisector of $[BC]$, but to build the point B we build the circle of center A with radius $[AC]$.

If a point appears in a position of type 1 in two constraints, we build the intersection of two geometrical objects (two straight lines, two circles or a straight line and a circle).

From constraints to construction

We can now describe how to build a set of points $P_1, \ldots, P_m$ satisfying a set of constraints

$(C_1 Q_{11} \ldots Q_{1r_1}), \ldots, (C_n Q_{n1} \ldots Q_{nr_n})$ where $\forall ij, Q_{ij} \in \{P_1, \ldots, P_m\}$.

We define strategy matrices with n lines and m columns. For each constraint, we define the *linked* point as the point which is built using this constraint from the other points. Each entry M_{ij} of the matrix M describes how the point P_j is used or produced with respect to the constraint C_i.

We need to find a matrix verifying the conditions:

(i) $\forall ij, M_{ij} \in \{-1, 0, 1, 2\}$,
(ii) $\forall ij, M_{ij} = -1 \Leftrightarrow P_j$ does not appear in the constraint C_i,
(iii) $\forall ij, M_{ij} > 0 \Rightarrow P_j$ appears in a position of type M_{ij} in the constraint C_i,
(iv) each line of M has exactly one strictly positive entry,
(v) $\forall j, \sum_i \sup(0, M_{ij}) \leqslant 2$,
(vi) the relation $\prec$ on points $P_1, ..., P_m$ defined by

$$a \prec b \Leftrightarrow \exists i, M_{ia} = 0 \,\text{and}\, M_{\text{ib}} > 0$$

is such that its transitive closure $\preccurlyeq$ is antisymmetric.

The condition (iv) means that each constraint will build a new point (possibly not completely determined if its position is of type 1).

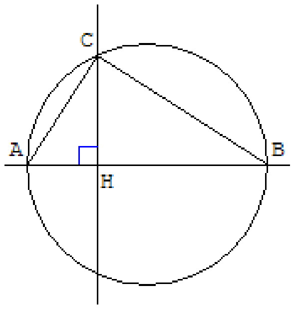
The condition (v) means that a point is built by one or two constraints, and if it is built by two it is the intersection of two curve objects.

The condition (vi) means that the construction does not loop: we do not use a built point to build an older one.

The algorithm to find such a matrix is now simple to describe. We first build a matrix M verifying conditions (i), (ii), (iii) and (iv), where for each line the positive entry is left-most, which is easy. We enumerate all matrices verifying conditions (i), (ii), (iii) and (iv), simply by lexicographically shifting the positive entry of each line to the right. We stop as soon as the matrix verifies conditions (v) and (vi), which are easy to check. Otherwise, we fail.

If this algorithm does not fail, we can use the matrix M directly to obtain the effective construction of the points: the minimal points for the relation $\preccurlyeq$ (which is a partial order) are taken as free points in the plane. From this set of points, called S_0, with the constraints, we can build a new set of points S_1 (which is the minimal points for $\preccurlyeq$ in $(\{P_1, \ldots, P_n\} - S_0)$, and so on. For each point, a GeoplanJ command is generated, using the constraints.

Let us take the following points and constraints as an example.



C_1: (collinear $A\ B\ H$)
C_2 :(ortho $H\ C\ A\ B$), i.e. the straight lines (HC) and (AB) are perpendicular.
C_3: (circle $A\ B\ C$), i.e. C is on the circle with diameter $[AB]$.
We have following types for positions:
$T(\text{collinear}) = (1, 1, 1)$
$T(\text{ortho}) = (1, 1, 1, 1)$
$T(\text{circle}) = (1, 1, 1)$

Let $P_1, \ldots, P_n = A, B, C, H$.

The initial matrix, verifying conditions (i), (ii), (iii) and (iv), is:

M	A	B	C	H
collinear	1	0	-1	0
ortho	1	0	0	0
circle	1	0	0	-1

It does not verify condition (v): the column A is linked by three constraints.

In lexicographic order, the next matrix verifying conditions (i), (ii), (iii) and (iv) is:

M	A	B	C	H
collinear	1	0	-1	0
ortho	1	0	0	0
circle	0	1	0	-1

It verifies condition (v), but not condition (vi): we have $B \prec A \prec B$

Continuing, we get the first matrix that verifies conditions (v) and (vi):

M	A	B	C	H
collinear	1	0	-1	0
ortho	0	0	1	0
circle	0	0	1	-1

We have then $B \prec A, H \prec A, A \prec C$.

The construction is then:

(i) Take B and H two free points in the plane.
(ii) Take A as a free point on the line (BH).
(iii) Take C as the intersection between the line orthogonal to (AB) and containing H and the circle with diameter $[AB]$.

Generic configurations, limitations and improvements

The whole discussion on free points (in the plane, on a straight line or a circle) only holds generically, i.e. outside a set of particular cases (for instance, three free points in a plane may be collinear, etc). But this set is of finite measure in $\mathbb{R}^n$, so when choosing free points at random, we have a high probability to be out of these degenerated situations.

This method is not complete. From a set of solvable geometrical constraints, it can fail to give a construction. The reason is that we work on non-linear constraints on points: this kind of constraints define a good degree of freedom not on points but on coordinates, or on more complex structures (determinants of projective points for instance).

Although not complete, the method has the advantage of being simple and fast, and it fails rarely in common use.

5 A few illustrating examples

After analyzing the matrix, a figure text is transferred to GeoplanJ so that the free points appear in green and the conclusion in red in the generated figure. The user can actually move the free points and the figure evolves accordingly.

5.1 Simson's line theorem

We proved in Coq Simson's line theorem, using oriented angles of vectors. This theorem states that: given a triangle and M a point in the plane, the three feet of the perpendiculars from M to the sides of the triangle are collinear if and only if M is on the circumcircle of the triangle.

Theorem **Simson_line**:
Let A, B, C, M, P, Q and R be points.
If **ABC is a triangle,**
BCM is a triangle,
ABM is a triangle,
ACM is a triangle,
P is the orthogonal projection of M on line (BC),
Q is the orthogonal projection of M on line (AC),
and R is the orthogonal projection of M on line (AB)
then **M is on the circumcircle of triangle ABC if and only if P, Q and R are collinear.**

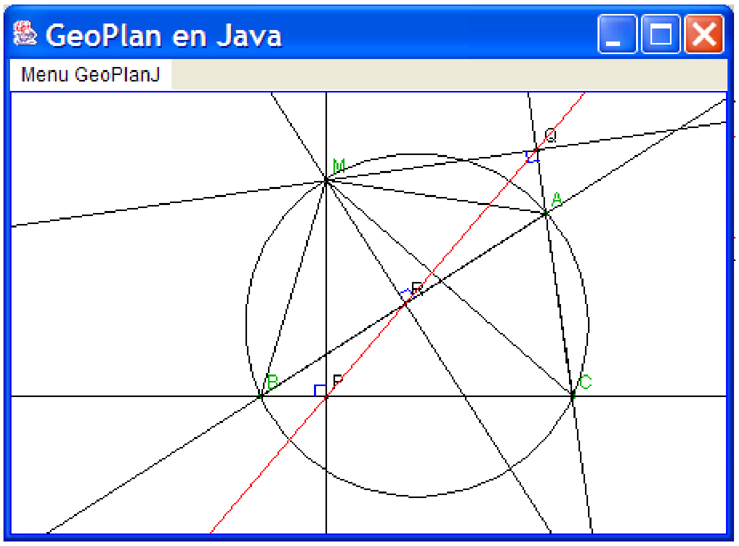


Fig. 3. Simson's theorem: statement in Pcoq and figure generated by GeoView

In figure 3, we show this theorem statement as it appears in Pcoq and the corresponding figure generated by GeoView. In this example, the conclusion of the theorem is an equivalence of two propositions. With GeoView, we consider the first one as a premise and the second one as the conclusion.

5.2 Nine-point circle theorem

We proved in Coq the nine-point circle theorem, also called Euler's circle theorem and Feuerbach's circle theorem, using vector calculus, oriented angles of vectors and homothety properties as in mathematics high-school.

Given a triangle, this circle passes through the three midpoints of the sides, the feet of the perpendicular to each side passing through the opposite point and also through the midpoints of the segments which join the vertices and the orthocenter. We also proved that the center of this circle is the midpoint of the segment joining the circumcenter O and the orthocenter H of the triangle and that the centroid G of the triangle lies on line (OH) which is called Euler's line.

In figure 4, we show this theorem statement as it appears in Pcoq and below the corresponding figure generated by GeoView where A, B and C are the only free points.

5.3 Disjunctions

Disjunctions that occur among the premises of a theorem imply that actually several figures must be considered, one for each disjunct.

Disjunctions that occur inside the conclusion of a theorem correspond to the fact that several configurations may occur for the same figure. This is illustrated in the theorem which states that: given a trapezoid $ABCD$ and I and J the midpoints of segments $[AB]$ and $[CD]$ respectively, the three lines (AC), (BD) and (IJ) either intersect or are parallel. The drawings corresponding to this theorem are shown in figure 2.

5.4 Representing real numbers

Real numbers that occur in vector expressions or geometric transformations such as rotations, similarities or inversions are displayed as points on a fixed straight line, displayed on top of the drawing, as it appears in figure 6. Moving the point with the mouse makes it possible to change the value of the real number, other geometrical objects that depend on this real move accordingly.

5.5 Existential quantification

As we already said, the constraints that appear in the conclusion of a statement are usually non-binding. If the conclusion is an existential quantification, then this introduces binding constraints, even though they occur in the conclusion.

Theorem **nine_point_circle**:
Let A, B, C, A', B', C', O, G, H, I, J, K, L, H1, H2 and H3 be points.
If **ABC is a triangle,**
A' is the midpoint of [BC],
B' is the midpoint of [AC],
C' is the midpoint of [AB],
G is the centroid of triangle ABC,
O is the circumcenter of triangle ABC,
H is the orthocenter of triangle ABC,
I is the midpoint of [OH],
J is the midpoint of [AH],
K is the midpoint of [BH],
L is the midpoint of [CH],
H1 is the orthogonal projection of A on line (BC),
H2 is the orthogonal projection of B on line (CA),
and H3 is the orthogonal projection of C on line (AB)
then
J is on the circumcircle of triangle A'B'C' and H1 is on the circumcircle of triangle A'B'C' and
K is on the circumcircle of triangle A'B'C' and H2 is on the circumcircle of triangle A'B'C' and
L is on the circumcircle of triangle A'B'C' and H3 is on the circumcircle of triangle A'B'C'.

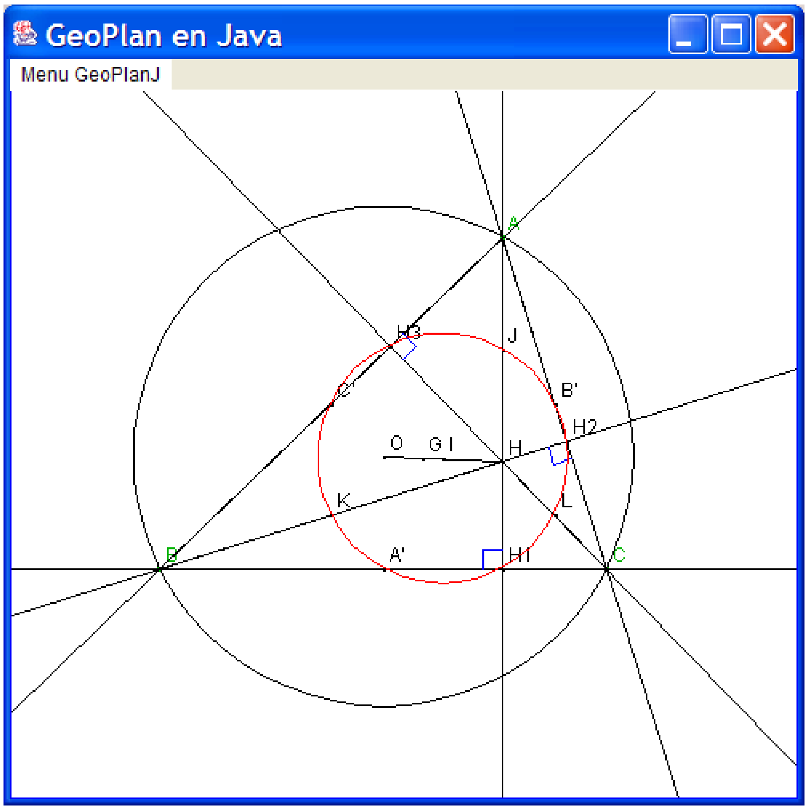


Fig. 4. Nine point circle theorem: statement in Pcoq and figure generated by GeoView

A good example is given by the composition of a translation by vector $\overrightarrow{AA'}$ and a non-trivial homothety of center I and ratio k. The theorem states that there exists a point J that is bound with the points I, A and A' and the ratio k of the homothety. The theorem statement appears in Pcoq as in figure 5.

```
Lemma composition_translation_homothety:
 ∀k : R.
 ∀I,  A,  A' : PO.
 k ≠ 1  =>
   k ≠ 0  =>
     ∃J : PO .
     ∀B,  B',  B1 : PO.
     B1 = (translation  A  A'  B)  => B' = (homothety  k  I  B1)  =>  B' = (homothety  k  J  B).
```

Fig. 5. Composition of a translation and a homothety: statement in Pcoq

The corresponding figure can be drawn as follows. In the dynamic figure, J moves whenever I, k and A and A' do, but does not move when B does. The point J is independent from B as is expressed by the nesting of universal or existential quantifiers in the theorem statement.

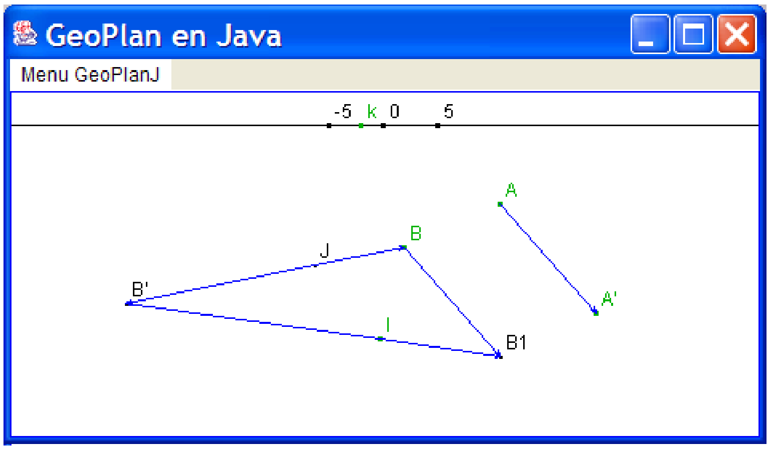


Fig. 6. Composition of a translation and a homothety: figure generated by GeoView.

5.6 *Representing complex numbers*

Complex numbers can be represented in the Gauss plane. To a given complex number in cartesian form $x + iy$, we associate the representation point with coordinates (x, y). In figure 7, we show the statement of a lemma that gives formulas to compute the real and imaginary values of a complex number from its polar form. The modulus r (distance) and an argument a of z (number of radians) are real numbers and are therefore displayed as points on a straight line on the top of the drawing. You can move the image point representing z by changing the values of r and a.

Lemma **conversion_polar_cartesian:**
$\forall z : C.\ \forall r,\ a,\ x,\ y : R.\ z \neq 0 \Rightarrow z = x + i\,y \Rightarrow z = r e^{ia} \Rightarrow x = r * \cos a \wedge y = r * \sin a.$

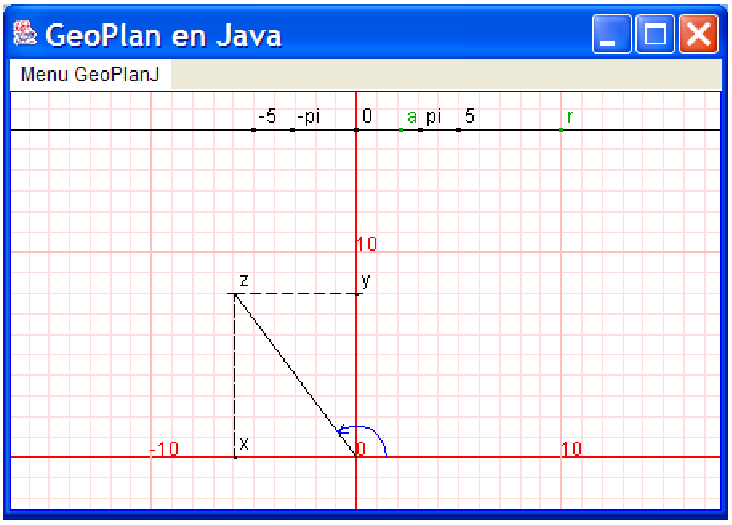


Fig. 7. Conversion formulas: statement in Pcoq and figure generated by GeoView

6 Concluding remarks

6.1 Related work

Our work is at the boundary between two fields: the first is concerned with interactive drawings, while the second is concerned with reasoning tools.

Drawing tools for geometric constructions abound. Many Java applets, that can be used in a web-browser can easily be found by searching on the world-wide web. For example, we can cite GEONExT [10] and the work of D.E. Joyce [13] that we could find and test easily.

In our work, we have concentrated on tools already available to math teachers such as Cabri-Geometre [19], EUKLID DynaGeo [8] (which we have not used), and GeoplanW and GeospaceW [11].

Reasoning tools for geometry can also be organized in two categories. A first category uses analytic methods to reduce geometric problems to algebraic problems and powerful algebraic tools (such as gröbner bases) to solve the latter. A second category relies on a prover, often a first order prover to deduce facts from a collection of axioms describing geometry. In the first category, we can cite the work of Shang-Ching Chou [5] and in the second category the work of Dominique Py [17] or the Leibniz laboratory managed by Nicolas Balacheff [2]. Our work is closer to the second category, especially since the point of view that tools are a teaching aid prevails in all these experiments. In particular, Baghera [20] also provides ways to display the drawings associated to exercises but it takes care of organizing the classroom or the schoolbag from the perspective of the teacher or the pupil.

This work is also related to the effort to formalize geometry with general purpose theorem provers. We can cite efforts to formalize Hilbert's axiomatic description of geometry [14,6], geometry algorithms [16], and topological properties of surfaces [7].

6.2 Conclusion

It should be clear to the reader that the design we have described in this paper is easily adapted to other proof or drawing tools. The geometry library itself should easily be described with any other proof tool, even the axiomatization itself can be changed. The experiment only requires to map basic notions from the geometry library to the basic notions of the drawing tool: points, alignments, cocyclic constraints, etc. For instance, Cabri-Geometre is one of the most popular tools in use in the French school system and it seems easy to adapt our work to this drawing tool.

The model of interaction that is proposed in this experiment uses the drawing tool only as an output device: formulas are taken from the written form and transformed into figure descriptions. It would be interesting to consider a reverse interaction scheme, where a figure would be described using the mouse by interacting with the drawing tool and then translated into a statement or a proof step. In this extension, the drawing tool could be used to improve the input capabilities of the proof environment. The reverse is also relevant, the logical statements are specifications for drawings and could be used as a new input language for the drawing tool, more concise than the current text input format or mouse interactions.

Another perspective is to adapt this work to three-dimensional geometry. Representing three dimensional objects on the two-dimensional screen poses difficulties. Planes in the three-dimensional space are difficult to render in an efficient way. The usual trick is to represent a plane by a parallelogram included in that plane, but then two intersecting planes are not guaranteed to exhibit an intersection in any drawing. Conversely, two straight lines that do not intersect may appear to be intersecting when the image is rendered on a flat screen.

References

[1] A. Amerkad, Y. Bertot, L. Pottier, and L. Rideau. Mathematics and Proof Presentation in Pcoq. In *Workshop Proof Transformation and Presentation and Proof Complexities in connection with IJCAR 2001*, Siena, Italy, June 2001.

[2] N. Balacheff, http://www-didactique.imag.fr/.

[3] M. Berger. *Geometry I*, chapter "Barycenters; the Universal Space", pages 67–83. Springer, 1987.

[4] Y. Bertot and L. Thery. A Generic Approach to Building User Interfaces for Theorem Provers. *the Journal of Symbolic Computation*, Vol. 25, pages 161–194, 1998.

[5] S. Chou, X. Gao, and J. Zhang. *Machine Proofs in Geometry: Automated Production of Readable Proofs for Geometry problems*. World Scientific, 1994.

[6] C. Dehlinger and J.F. Dufourd. Formalizing the Trading Theorem for the Classification of Surfaces. In *TPHOLs'02*, pages 148–163. LNCS 2410, Springer Verlag, 2002.

[7] C. Dehlinger, J.F. Dufourd, and P. Schreck. Higher-Order Intuitionistic Formalization and Proofs in Hilbert's Elementary Geometry. In *Automated Deduction in Geometry*, pages 306–324, 2000.

[8] EUKLID DynaGeo, http://www.dynageo.de.

[9] GeoplanJ, http://mapage.noos.fr/fkotecki/geoplanj.html.

[10] GEONExT, http://geonext.uni-bayreuth.de.

[11] GeoplanW, http://www2.cnam.fr/creem/geoplanw/geoplanw.htm.

[12] GeoView, http://www-sop.inria.fr/lemme/geoview/geoview.html.

[13] D.E. Joyce, http://aleph0.clarku.edu/ djoyce/java/geometry/geometry.html.

[14] L.I. Meikle and J.D. Fleuriot. Formalizing Hilbert's Grundlagen in Isabelle/Isar. In *TPHOLs'03*, pages 319–334, 2003.

[15] H. Naciri and L. Rideau. Formal Mathematical Proof Explanations in Natural Language Using MathML: An Application to Proofs in Arabic. In *MathML International Conference: Hickory Ridge Conference Center, Chicago, IL, USA, June 28–30*, 2002.

[16] D. Pichardie and Y. Bertot. Formalizing Convex Hulls Algorithms. In *TPHOLs'01*, pages 346–361. LNCS 2152, Springer Verlag, 2001.

[17] D. Py. *Environnements Interactifs d'Apprentissage et Démonstration en géométrie*. Habilitation à Diriger des Recherches, Université de Rennes 1, 2001.

[18] The Coq Development Team. The Coq Proof Assistant: Reference Manual: Version 7.2. Technical Report RT-0255, INRIA, February 2002.

[19] J. Vincent. *Exploring 2-Dimensional Space with Cabri Geometry II*. Mathematical Association of Victoria.

[20] C. Webber and S. Pesty. Emergent Diagnosis via Coalition Formation. In Garijo F., editor, *8th Iberoamerican Conference on Artificial Intelligence (IBERAMIA 2002)*, pages 755–764, Spain, November 2002. LNAI 2527, Springer Verlag.