Automated Generation of Illustrations for Synthetic Geometry Proofs

Predrag Janičić[0000-0001-8922-4948]

Department for Computer Science
Faculty of Mathematics
University of Belgrade
Studentski trg 16
11000 Belgrade, Serbia

janicic@matf.bg.ac.rs

Julien Narboux[0000-0003-3527-7184]

UMR 7357 CNRS
University of Strasbourg
Pôle API, Bd Sébastien Brant
BP 10413
67412 Illkirch, France

narboux@unistra.fr

We report on a new, simple, modular, and flexible approach for automated generation of illustrations for (readable) synthetic geometry proofs. The underlying proofs are generated using the Larus automated prover for coherent logic, and corresponding illustrations are generated in the GCLC language. Animated illustrations are also supported.

1   Introduction

In geometry, proofs and sketches are closely related. Humans need sketches to visualize geometry statements, to reason about geometrical problems, and to understand geometry proofs more easily. That is why the use of dynamic geometry software is pervasive in education. However, many famous examples of incorrect or insufficiently justified steps in geometric reasoning can be attributed to the improper use of visual intuition based on geometric figures. This goes from unjustified proof steps in Euclid's Elements to proofs and statements which rely to implicit assumptions from the figure in the context of education [21], passing by incorrect proofs of Euclid's fifth postulate such as the one by Legendre. Since their early days, automated deduction systems for geometry have also tried to use the knowledge given by the figure[13]. Some authors, also studied how illustrations, following some rules, can provide proofs. This is called diagrammatic reasoning. Diagrammatic reasoning systems can be defined rigorously enough such that they behave as formal systems [29, 19].

In this paper, we present our ongoing work on visual illustrations for geometry proofs. We are not dealing with diagrammatic reasoning, rather we focus on generation of illustrations for proofs in geometry in the form of sketches similar to what a human would create. The underlying proofs are readable synthetic proofs generated by our automated theorem prover Larus for coherent logic [17]. The use of coherent logic is an essential ingredient of our approach. The proofs in coherent logic are organized in a purely forward reasoning style. Hence, the fact which has to be proved is not changed during the proof. In logical terms, we use the fact that coherent logic enjoys a complete deduction system which never modifies the right hand sides of the sequents. This contributes to the clarity of the visualization. We illustrate proofs as sketches that get updated by each proof step. The initial sketch depicts (a model of) the assumptions. Then, new known facts are introduced along the proofs and depicted dynamically by decorating the initial sketch.

The generated illustrations are stored in a domain specific, geometry oriented, language GCLC (developed by the first author) [14, 16]. The illustrations can also be presented as animations, in a step-by-step manner.

2   Related Work

There is a huge number of sources addressing the role of geometric figures in general, so we focus here only on a closely related work about the visualization of proofs in geometry which have either been generated automatically (by automated theorem proving) or machine checked (by interactive theorem proving).

Bertot et al. [4] proposed to embed a dynamic geometry software into PCoq [5, 1] (PCoq is a user interface for the Coq proof assistant [9]). The second author implemented a prototype dynamic geometry software GeoProof which can export statements in the syntax of Coq [20]. Gao's MMP/geometer software, an automatic geometry theorem prover based on algebraic methods, offered a way to draw figures corresponding to the statements [12]. Wang proposed generating figures from a set of geometrical constraints by decomposing the system of polynomials into irreducible representative triangular sets, and finding an adequate numerical solution from each triangular set [27]. The first author implemented the GCLC[1] software in which geometry configurations are described using a custom language, conjectures about configurations can be expressed and then proved automatically by several available automated theorem provers [14, 15]. Pham and Bertot implemented a prototype connecting GeoGebra to PCoq [23]. Botana et al. implemented automated theorem provers based on algebraic methods within GeoGebra [7].

All the above works are focused on the visualization of statements. Regarding visualization of proofs, Wilson and Fleuriot designed and implemented a tool for the visualization of proofs as direct acyclic graph with nodes depicted using geometric figures [28] for the full-angle method [8]. The most significant contribution related to visualization of geometric proofs is the work of Ye et al. implemented in JGEX [32]. Ye et al. use the algebraic characterization and triangulation to generate figures automatically for the statements. JGEX proposes several modes for illustration of proofs: illustration by animated diagrams, or animated diagrams + text, and they apply this method to two kinds of proofs: manually crafted proofs [30], or proofs generated automatically by the full-angle method or the deductive database method [31].

3   The Context

Synthetic Proofs in Coherent Logic   A formula of first-order logic is said to be *coherent* if it has the following form:

$$A_0(\vec{x}) \wedge \ldots \wedge A_{n-1}(\vec{x}) \Rightarrow \exists \vec{y}(B_0(\vec{x},\vec{y}) \vee \ldots \vee B_{m-1}(\vec{x},\vec{y}))$$

where universal closure is assumed, and where $0 \leq n$, $0 \leq m$, $\vec{x}$ denotes a sequence of variables $x_0, x_2, \ldots, x_{k-1}$ $(0 \leq k)$, A_i (for $0 \leq i \leq n-1$) denotes an atomic formula (involving zero or more variables from $\vec{x}$), $\vec{y}$ denotes a sequence of variables $y_0, y_2, \ldots, y_{l-1}$ $(0 \leq l)$, and B_j (for $0 \leq j \leq m-1$) denotes a conjunction of atomic formulae (involving zero or more of the variables from $\vec{x}$ and $\vec{y}$). If $n = 0$, then the left hand side of the implication is assumed to be $\top$ and can be omitted. If $m = 0$, then the right hand side of the implication is assumed to be $\bot$ and can be omitted. There are no function symbols with arity greater than zero. Coherent formulae do not involve negation. A coherent theory is a set of sentences, axiomatised by coherent formulae, and closed under derivability.[2]

---

[1] http://www.matf.bg.ac.rs/~janicic/gclc

[2] A coherent formula is also known as a "special coherent implication", "geometric formula", "basic geometric sequent" [10]. A coherent theory is sometimes called a "geometric theory" [18]. However, a much more widely used notion of "geometric formula" allows *infinitary disjuctions* (but only over finitely many variables) [26]. Coherent formulae involve only finitary

A number of theories and theorems can be formulated directly and simply in coherent logic (CL). Several authors independently point to CL (or rules similar to those of CL) as suitable for expressing (sometimes – also automating) significant portions of mathematics [11, 2]. In contrast to resolution-based theorem proving, in CL the conjecture being proved is kept unchanged and proved without using refutation, Skolemization and clausal form. Thanks to this, CL is suitable for producing readable synthetic proofs [6].

Every first-order theory has a coherent conservative extension [10], i.e., any first-order theory can be translated into CL, possibly with additional predicate symbols. Translation of FOL formulae into CL involves elimination of negations: negations can be kept in place and new predicates symbols for corresponding sub-formula have to be introduced, or negations can be pushed down to atomic formulae [24]. In the latter case, for every predicate symbol R (that appears in negated form), a new symbol $\overline{R}$ is introduced that stands for $\neg R$, and the following axioms are introduced $\forall \vec{x}(R(\vec{x}) \wedge \overline{R}(\vec{x}) \Rightarrow \bot)$, $\forall \vec{x}(R(\vec{x}) \vee \overline{R}(\vec{x}))$. In order to enable more efficient proving, some advanced translation techniques are used. Elimination of function symbols, sometimes called *anti Skolemization*, is also done by introducing additional predicate symbols [22].

For a set of coherent axioms AX and the statement $A_0(\vec{x}) \wedge \ldots \wedge A_{n-1}(\vec{x}) \Rightarrow \exists \vec{y}(B_0(\vec{x},\vec{y}) \vee \ldots \vee B_{m-1}(\vec{x},\vec{y}))$ to be proved, within the suitable, very simple CL proof system one has to derive $\exists \vec{y}(B_0(\vec{a},\vec{y}) \vee \ldots \vee B_{m-1}(\vec{a},\vec{y}))$, where $\vec{a}$ are fresh constants. Proofs can be built in the manner of forward reasoning, as illustrated by the following example.

Example 1 *Consider the following set of axioms:*

ax1: $\forall x\,(p(x) \Rightarrow r(x) \vee q(x))$

ax2: $\forall x\,(q(x) \Rightarrow \bot)$

and the following conjecture that can be proved as a CL theorem:

$\forall x\,(p(x) \Rightarrow r(x))$

Consider arbitrary a such that: $p(a)$. It should be proved that $r(a)$.

1. $r(a) \vee q(a)$ *(by MP, from $p(a)$ using axiom ax1; instantiation: $X \mapsto a$)*

  2. *Case $r(a)$:*

    3. *Proved by assumption! (by QEDas)*

  4. *Case $q(a)$:*

    5. $\bot$ *(by MP, from $q(a)$ using axiom ax2; instantiation: $X \mapsto a$)*

    6. *Contradiction! (by QEDefq)*

7. *Proved by case split! (by QEDcs, by $r(a), q(a)$)*

Larus is an automated theorem prover for coherent logic. It has several underlying proving engines, based on different approaches. Larus is publicly available and open-source.[3]

---

disjunctions, so coherent logic can be seen as a special case of geometric logic, or as a first-order fragment of geometric logic.

[3]https://github.com/janicicpredrag/Larus

4   Method for Generating Illustrations

The basic idea of the method is simple: if we know how to visually interpret all proof steps that introduce new objects or facts, then we know how to produce a complete illustration. Indeed, following the Curry-Howard correspondence which relates computer programs and proofs (a proof is a program, and the formula it proves is the type for the program), proofs of existential statements can be depicted by a function which takes the universally quantified geometric objects of the statement as input and construct a witness of the conclusion. The illustration is based on a sequence of such witnesses in one universum. We will focus on an usual choice – a Cartesian space (hence, for instance, geometry points will map to Cartesian points). Of course, our illustrations will be just one model for just one proof branch.

This idea is very well suited to coherent logic as an underlying logical framework (see section 3) and to forward chaining proofs, which also fits well into CL. In CL proofs, new facts are derived using *modusponens*, so we have to handle only these rule applications in the proof.

The above idea, obviously, cannot be applied in conjunction with resolution or saturation based theorem provers for first-order logic, even less in conjunction with algebraic provers (such as those based on Gröbner bases or Wu's method), nor it can be applied on proof traces produced by such provers.

Rule Applications Let us assume that, within the proof, there is an application of the following axiom (with the obvious intended meaning):

$$\forall x, y \ (point(x) \land point(y) \Rightarrow \exists z \ between(x, z, y))$$

It may have attached the following visual interpretation: „for two concrete Cartesian points a and b, the Cartesian point c such that c is between a and b can be created as the Cartesian midpoint of ab".

Assuming that the points a and b, occurring in the proof are associated Cartesian coordinates $(2, 5)$ and $(4, 11)$, if the above axiom has been applied to them, then the new witness point will have the associated Cartesian coordinates $(3, 8)$.

Not only new witnesses can be created, but also some new features can also be established and then illustrated. In the above example, we would also draw the segment ab. Furthermore, this approach can be extended to axioms that do not introduce new objects, but only establish some new facts. For instance, if the proof establishes that some three points are collinear, then we will highlight the Cartesian line that contain the three Cartesian points corresponding to them.

Axioms vs. Theorems The only thing that we need to illustrate a proof is visual interpretation of all applied axioms and lemmas/theorems within the proof. The visual interpretation of used theorems can be produced automatically recursively, using the same approach. Ultimately, what we need are only visual interpretations of all axioms, provided by a human.

Premises and Initial Configuration We explained how the figure is updated by each *modusponens* rule application. But how do we start the illustration in the first place? Recall that we prove theorems of the form: $A_0(\vec{x}) \land \ldots \land A_{n-1}(\vec{x}) \Rightarrow \exists \vec{y}(B_0(\vec{x}, \vec{y}) \lor \ldots \lor B_{m-1}(\vec{x}, \vec{y}))$ (and that all axioms used also have that form). In order to build the initial illustration we need some constants $\vec{a}$ such that: $A_0(\vec{a}) \land \ldots \land A_{n-1}(\vec{a})$ holds. But how can we find and illustrate such objects? We can do that by using the same approach as described above, applied to the theorem: $\exists \vec{x}(A_0(\vec{x}) \land \ldots \land A_{n-1}(\vec{x}))$.[4]
In fact, here we perform a sort of constraint solving using theorem proving. If we provide to the

---

[4]If, instead of just a model of the premises, we want to generate a *construction procedure* for a whole class of initial configurations, we can choose to quantify universally some of the variables, and apply the procedure to the statement:

prover axioms modeling only ruler-and-compass geometry, then we can obtain constructive proof of existence corresponding to a ruler-and-compass construction. But, if we provide more powerful axioms to the prover, such as the ability of trisecting the angle, then we could also, in principle, illustrate proofs such as a proof of Morley's theorem (this is only an example because Morley's theorem is currently way out of reach of our prover). This approach also requires some care: the witness of the existential should be taken as general as possible (in order to avoid misleading illustration). This problem can be addressed by adding non-degenerate conditions, for instance, that the points asserted to exist are pairwise distinct and non-collinear.

Case Splits There can be many case splits in the proof and it would make no sense to illustrate all proof leaves. Indeed, often some proof branches are contradictory. Not only that they are less interesting (they typically correspond to degenerative cases) but they do not have models. Hence, if we have several proof branches in one proof node, we do the following:

- if all of them end with contradiction, then they all belong to some upper contradictory proof branch, and we illustrate neither of them;[5]
- If there are some proof branches that do not end with contradiction, then one that corresponds to the model being built should be illustrated.

We could combine the above with other policies. For instance, for all case splits of the form $R(\vec{a}) \vee \overline{R}(\vec{a})$, we could follow only the negative branch, $\overline{R}(\vec{a})$ (for instance, three points are non-collinear) as generally they correspond to non-degenerated cases.

Randomization In order to make illustrations partly unpredictable and more interesting, some randomization may be added to the visual interpretation of the axiom. For example, if there is an axiom stating that for any two distinct points there is a third one between them, then in the visual counterpart, that third one could be chosen based on a pseudo-random number between 0 and 1.

5 Method Implementation

We implemented the described method within our automated theorem prover for coherent logic, Larus [17]. Larus' flexible architecture already had proof export to LaTeX and Coq supported. So, we have implemented just one class more – the class that exports generated CL proofs (in Larus' internal representation) to visual representation. The new code has less than 200 lines of C++.

For the target language, i.e., the language of visual representation we chose the GCL language [16], a rich, special purpose language for mathematical, especially geometry illustrations.

For each theorem τ (given in the TPTP format, the standard format for theorem provers [25]), once it has been proved, the prover generates a file that contains only a function that corresponds to that theorem, and a „main" file that includes files with all used axioms/theorems, along with the file for τ. That „main" file invokes the function for existence of premises for τ, and then the function for τ itself. The prover also generates a TPTP formulation of the theorem that corresponds to existence of premises for τ (see Section 4). Given that theorem was proved, its visual representation can also be obtain automatically.

$\forall \vec{x'} \exists \vec{x''} (A_0(\vec{x'}, \vec{x''}) \wedge \ldots \wedge A_{n-1}(\vec{x'}, \vec{x''}))$. Those variables $\vec{x'}$ quantified universally would then correspond to the *free points* of the figure. Depending on the choice of $\vec{x'}$, the statement can be a theorem or not, hence, the choice of the set of these free points should be made by the user.

[5] If the premises themselves are contradictory, then we do not provide an illustration. Actually, in some cases we could also somehow illustrate contradictory branches (as it is done in some textbooks) – for instance, if it is proved that three points are both colinear and non-colinear, we could draw a curved line that connect them.

Animations can be obtained simply by showing visualizations of proof steps one by one or, better still, by showing each step in some emphasized manner (or in another color) and then in a regular way.

Generated illustrations, stored as readable GCLC files, can be further modified and improved by a human.

We present one example in detail in the next section, and give two more examples in Appendix.

6   Example: Euclid's *Elements*, Book I, Proposition 11

As a main example, we use Proposition 11 from Euclid's *Elements*, Book I. Its original form reads as follows: „To draw a straight line at right angles to a given straight line from a given point on it." The statement represented in first-order logic, following the formalization of first book of Euclid's *Elements* as proposed by Beeson et al. [3], is the following:

$$\forall A, B, C \; \text{BetS} \, ACB \Rightarrow \exists X \; \text{Per} \, ACX$$

(BetS *ACB* means that *C* is strictly between *A* and *B*, Per *ACX* means that *ACX* is a right angle with the vertex *C*). The TPTP file with all needed axioms and lemmas[6] (listed as axioms as well) is as follows:

```
fof(lemma_betweennotequal,axiom, (! [A,B,C] :
    ((betS(A,B,C)) => ((( B != C ) & ( A != B ) & ( A != C )))))).
fof(lemma_extension,axiom, (! [A,B,P,Q] : (? [X] :
    ((( A != B ) & ( P != Q )) => ((betS(A,B,X) & cong(B,X,P,Q)))))))).
fof(proposition_01,axiom, (! [A,B] : (? [X] : ((( A != B )) =>
    ((equilateral(A,B,X) & triangle(A,B,X))))))).
fof(defequilateral,axiom, (! [A,B,C] :
    ((equilateral(A,B,C)) => ((cong(A,B,B,C) & cong(B,C,C,A)))))).
fof(defequilateral2,axiom, (! [A,B,C] :
    ((cong(A,B,B,C) & cong(B,C,C,A)) => ((equilateral(A,B,C)))))).
fof(lemma_doublereverse,axiom, (! [A,B,C,D] :
    ((cong(A,B,C,D)) => ((cong(D,C,B,A) & cong(B,A,D,C)))))).
fof(lemma_congruenceflip,axiom, (! [A,B,C,D] :
    ((cong(A,B,C,D)) => ((cong(B,A,D,C) & cong(B,A,C,D) & cong(A,B,D,C)))))).
fof(defcollinear,axiom, (! [A,B,C] : ((col(A,B,C)) =>
    ((( A = B )) | (( A = C )) | (( B = C )) |
    (betS(B,A,C)) | (betS(A,B,C)) | (betS(A,C,B)))))).
fof(defcollinear2a,axiom, (! [A,B,C] : ((( A = B )) => ((col(A,B,C)))))).
fof(defcollinear2b,axiom, (! [A,B,C] : ((( A = C )) => ((col(A,B,C)))))).
fof(defcollinear2c,axiom, (! [A,B,C] : ((( B = C )) => ((col(A,B,C)))))).
fof(defcollinear2d,axiom, (! [A,B,C] : ((betS(B,A,C)) => ((col(A,B,C)))))).
fof(defcollinear2e,axiom, (! [A,B,C] : ((betS(A,B,C)) => ((col(A,B,C)))))).
fof(defcollinear2f,axiom, (! [A,B,C] : ((betS(A,C,B)) => ((col(A,B,C)))))).
fof(lemma_collinearorder,axiom, (! [A,B,C] :
    ((col(A,B,C)) => ((col(B,A,C) & col(B,C,A) & col(C,A,B) & col(A,C,B) & col(C,B,A)))))).
fof(deftriangle,axiom, (! [A,B,C] : ((triangle(A,B,C)) => ((~ (col(A,B,C))))))).
fof(deftriangle2,axiom, (! [A,B,C] : ((~(col(A,B,C))) => ((triangle(A,B,C)))))).
fof(defrightangle,axiom, (! [A,B,C] : (? [X] :
    ((per(A,B,C)) => ((betS(A,B,X) & cong(A,B,X,B) & cong(A,C,X,C) & ( B != C ))))))).
fof(defrightangle2,axiom, (! [A,B,C,X] :
    ((betS(A,B,X) & cong(A,B,X,B) & cong(A,C,X,C) & ( B != C )) => ((per(A,B,C)))))).
fof(proposition_11,conjecture,(! [A,B,C] : (? [X] : ((betS(A,C,B)) => ((per(A,C,X))))))).
```

---

[6]The proof checked by Coq can be found here: https://github.com/GeoCoq/GeoCoq/blob/master/Elements/OriginalProofs/proposition_11.v

The proposition was proved automatically by the Larus prover, giving the following proof in LaTeX[7]:

Theorem 1 *proposition_11 :* $\forall A \ \forall B \ \forall C \ (betS(A,C,B) \Rightarrow \exists X \ (per(A,C,X)) \ )$

Proof:
  Consider arbitrary a, b, c such that: $betS(a,c,b)$. It should be proved that $\exists X \ per(a,c,X)$.

1. Let w be such that $betS(a,c,w) \wedge cong(c,w,a,c)$ (by MP, from $betS(a,c,b)$, $betS(a,c,b)$ using axiom lemma_extension; instantiation: $A \mapsto a, B \mapsto c, P \mapsto a, Q \mapsto c$)

2. Let $w1$ be such that $equilateral(a,w,w1) \wedge triangle(a,w,w1)$ (by MP, from $betS(a,c,w) \wedge cong(c,w,a,c)$ using axiom proposition_01; instantiation: $A \mapsto a, B \mapsto w$)

3. $w1 = c \vee w1 \neq c$ (by MP, using axiom eq_excluded_middle; instantiation: $A \mapsto w1, B \mapsto c$)

   4. Case $w1 = c$:

      5. $col(a,w,w1)$ (by MP, from $betS(a,c,w) \wedge cong(c,w,a,c)$, $w1 = c$ using axiom colEqSub2; instantiation: $A \mapsto a, B \mapsto w, C \mapsto c, X \mapsto w1$)

      6. $\bot$ (by MP, from $col(a,w,w1)$, $equilateral(a,w,w1) \wedge triangle(a,w,w1)$ using axiom nnncolNegElim; instantiation: $A \mapsto a, B \mapsto w, C \mapsto w1$)

      7. Contradiction! (by QEDefq)

   8. Case $w1 \neq c$:

      9. $per(a,c,w1)$ (by MP, from $betS(a,c,w) \wedge cong(c,w,a,c)$, $betS(a,c,w) \wedge cong(c,w,a,c)$, $equilateral(a,w,w1) \wedge triangle(a,w,w1)$, $w1 \neq c$ using axiom defrightangle2; instantiation: $A \mapsto a, B \mapsto c, C \mapsto w1, X \mapsto w$)

      10. Proved by assumption! (by QEDas)

11. Proved by case split! (by QEDcs, by $w1 = c, w1 \neq c$)

Note that Larus' proofs omits applications of „simple axioms", axioms that are universal implications from one atomic formula to another. The above proof explicitly uses only the following lemmas (not counting those implied by equality axioms and those introducing $\bot$): *lemma_extension*, *proposition_01*, *defrightangle2*. Hence, we need visual interpretation of these. In order to obtain them, we can run the prover or provide the visual interpretation ourselves. For instance, the visual counterpart for the famous Euclid's proposition 1 (on existence of a equilateral triangle on a given segment), can be like the following:

```
procedure proposition_01 { A B X } {
  circle c1 A B
  drawcircle c1
  circle c2 B A
  drawcircle c2
  intersec2 X2 X c1 c2
  cmark X
}
```

In this visual interpretation, the point X is obtained (in the Cartesian model) as an intersection of two circles – one with the center A with B on it, and one the center B with A on it (the command `cmark X` annotates the point X by a small circle). Of course, one could have chosen the other intersection of two

---

[7]This LaTeX proof presentation is still very verbatim and it provides much more information than a traditional proof; for future work we are planning to produce a more natural output.

circles. Again, instead of providing this visual interpretation, we could run the prover on *proposition_01* and could get another function, expressed in terms of axioms (or, possibly, some other lemmas).

The prover also generates the conjecture which establishes existence of objects such that the premises of the main conjecture hold:

```
fof(proposition_11, axiom, ( ? [A,B,C] : (betS(A,C,B))))
```

This conjecture can be proved automatically or can be illustrated by hand, as here (within the file `proposition_11_exists.gcl`):

```
procedure proposition_11_exists { a b c  }  {
  point a 8 2
  point b 22 7
  towards c a b 0.7
  cmark_t a
  cmark_t b
  cmark_t c
}
```

The main conjecture is described by the following function, generated automatically by the prover (within the file `proposition_11.gcl`):

```
procedure proposition_11 { a b c w  }  {
  call lemma_extension { a c a c w }
  mark_t w
  call proposition_01 { a w w1 }
  mark_t w1
  % --- Illustration for branch 2
  call defrightangle2 { a c w1 w }
}
```

One point (*w*) is introduced by the application of *lemma_extension*, and another (*w*1) by the application of *proposition_01*. As we can see, the illustration follows the second, non-contradictory branch of the proof. The final proof step does not introduce new objects, but only establishes a property $ac \perp cw_1$ (which is stressed by drawing a small square at the vertex of the angle).

Finally, the main file, that will be processed by GCLC, is generated automatically and looks like the following:

```
% ----- Proof illustration -----
include proposition_11.gcl
include proposition_11_exists.gcl
include lemma_extension.gcl
include proposition_01.gcl
include defrightangle2.gcl
%--------------------------
call proposition_11_exists { a b c }
call proposition_11 { a b c w  }
```

After the „include" section with all needed files/functions, there is a call to the function which illustrates the objects such that the premises hold. After that, with the concrete points a, b, c (provided by the previous function), the main function is invoked. The illustration can be modified such that it
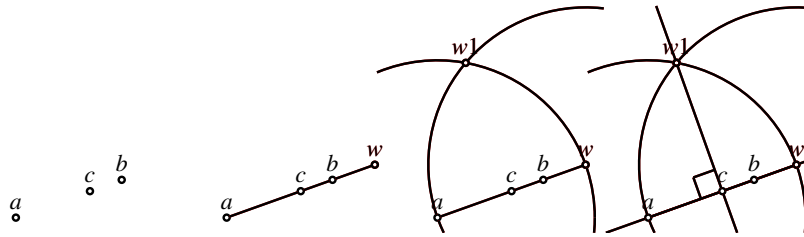
Figure 1: Four steps in illustration of the proposition 11

provides animation (within GCLC graphical environment, but it can be also exported as a sequence of images). A visual counterpart of each step is shown first in red, then in black and it remains like that. This is implemented using layers, as in the following addition to the main GCLC file, which ensures that all levels from i ($i = 1, 2, 3, \ldots$) are hidden (in our example, there are 6 layers):

```
animation_frames 7 1
point A0 0 0
point A1 1 0 7 0
distance dA A0 A1
hide_layers_from dA
```

Processing the above GCLC file gives the animation, i.e., the illustration that gets extended in several steps, as shown in Figure 1. Within GCLC environment, each step is shown first in red, then in black colour. (Figures can be exported to different formats, e.g. `TikZ`.)

7   Conclusions and Perspectives

We presented a new approach for automated generation of illustrations of proofs of geometry theorems. The approach is simple, as it is a small extension to our prover for coherent logic, Larus. The approach is modular, as all illustrations rely only on visual interpretations of axioms used. The approach is flexible, as one can provide by hand different visual counterparts of the axioms, but also of particular lemmas used within other proofs. The illustrations are generated in the form of the GCL language and they can also be viewed as animations, where proof steps are shown step-by-step.

When Larus finds a proof, it can generate the corresponding Coq proof and illustration. In the future, this work could be extended to deal with coherent logic proofs which are not necessarily obtained automatically by Larus, but constructed interactively within Coq.

We also plan to improve the translation of the proof to text form, to generate a more natural English text.

References

[1] Ahmed Amerkad, Yves Bertot, Loïc Pottier & Laurence Rideau (2001): *Mathematics and Proof Presentation in Pcoq*. In: *Workshop Proof Transformation and Presentation and Proof Complexities in connection with IJCAR 2001*, Siena.

[2] Jeremy Avigad, Edward Dean & John Mumma (2009): *A Formal System for Euclid's Elements*. The Review of Symbolic Logic 2, pp. 700–768, doi:10.1017/S1755020309990098.

[3] Michael Beeson, Julien Narboux & Freek Wiedijk (2019): *Proof-checking Euclid*. Annals of Mathematics and Artificial Intelligence 85(2-4), pp. 213–257, doi:10.1007/s10472-018-9606-x. Publisher: Springer.

[4] Yves Bertot, Frédérique Guilhot & Loïc Pottier (2004): *Visualizing Geometrical Statements with GeoView*. Proceedings of the Workshop User Interfaces for Theorem Provers 2003 103, pp. 49–65, doi:10.1016/j.entcs.2004.09.013.

[5] Yves Bertot & Laurent Thery (1998): *A Generic Approach to Building User Interfaces for Theorem Provers*. The Journal of Symbolic Computation 25, pp. 161–194, doi:10.1006/jsco.1997.0171.

[6] Marc Bezem & Thierry Coquand (2005): *Automating Coherent Logic*. In Geoff Sutcliffe & Andrei Voronkov, editors: *12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning — LPAR 2005*, Lecture Notes in Computer Science 3835, Springer-Verlag, pp. 246–260, doi:10.1007/11591191_18.

[7] Francisco Botana, Markus Hohenwarter, Predrag Janičić, Zoltán Kovács, Ivan Petrović, Tomás Recio & Simon Weitzhofer (2015): *Automated Theorem Proving in GeoGebra: Current Achievements*. Journal of Automated Reasoning 55(1), pp. 39–59, doi:10.1007/s10817-015-9326-4.

[8] Shang-Ching Chou, Xiao-Shan Gao & Ji Zhang (1996): *Automated Generation of Readable Proofs with Geometric Invariants, II. Theorem Proving With Full-Angles*. Journal of Automated Reasoning 17(13), pp. 349–370, doi:10.1007/BF00283134.

[9] Coq development team, The (2010): *The Coq proof assistant reference manual, Version 8.3*. LogiCal Project. Available at http://coq.inria.fr.

[10] Roy Dyckhoff & Sara Negri (2015): *Geometrization of first-order logic*. The Bulletin of Symbolic Logic 21, pp. 123–163, doi:10.1017/bsl.2015.7.

[11] M. Ganesalingam & W. T. Gowers (2013): *A fully automatic problem solver with human-style output*. CoRR abs/1309.4501.

[12] Xiao-Shan Gao & Qiang Lin (2004): *MMP/Geometer - A Software Package for Automated Geometric Reasoning*. In: *Proceedings of Automated Deduction in Geometry (ADG02)*, Lecture Notes in Computer Science 2930, Springer-Verlag, pp. 44–66, doi:10.1007/978-3-540-24616-9_4.

[13] Herbert Gelernter, J. R. Hansen & Donald Loveland (1960): *Empirical explorations of the geometry theorem machine*. In: *Papers presented at the May 3-5, 1960, western joint IRE-AIEE-ACM computer conference*, IRE-AIEE-ACM '60 (Western), ACM, San Francisco, California, pp. 143–149, doi:10.1145/1460361.1460381.

[14] Predrag Janičić (2006): *GCLC — A Tool for Constructive Euclidean Geometry and More Than That*. In Andrés Iglesias & Nobuki Takayama, editors: *Mathematical Software - ICMS 2006*, Lecture Notes in Computer Science 4151, Springer, pp. 58–73, doi:10.1007/11832225_6.

[15] Predrag Janičić (2009): *GCLC 9.0/WinGCLC 2009*. Manual for the GCLC Dynamic Geometry Software.

[16] Predrag Janičić (2010): *Geometry Constructions Language*. Journal of Automated Reasoning 44(1-2), pp. 3–24, doi:10.1007/s10817-009-9135-8.

[17] Predrag Janičić & Julien Narboux (2021): *Theorem Proving as Constraint Solving with Coherent Logic*. Submitted.

[18] Saunders MacLane & Ieke Moerdijk (1992): *Sheaves in geometry and logic: a first introduction to topos theory*. Springer-Verlag.

[19] Nathaniel Miller (2001): *A diagrammatic formal system for Euclidean geometry*. Ph.D. thesis, Cornell University.

[20] Julien Narboux (2007): *A Graphical User Interface for Formal Proofs in Geometry*. Journal of Automated Reasoning 39(2), pp. 161–180, doi:10.1007/s10817-007-9071-4.

[21] Julien Narboux & Viviane Durand-Guerrier (2021): *Combining pencil/paper proofs and formal proofs, a challenge for Artificial Intelligence and mathematics education*. In: *Mathematics Education in the Age of Artificial Intelligence: How Intelligence can serve mathematical human learning*, Springer. In press.

[22] Hans de Nivelle & Jia Meng (2006): *Geometric Resolution: A Proof Procedure Based on Finite Model Search*. In Ulrich Furbach & Natarajan Shankar, editors: *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, *Lecture Notes in Computer Science* 4130, Springer, pp. 303–317, doi:10.1007/11814771_28.

[23] Tuan Minh Pham & Yves Bertot (2012): *A Combination of a Dynamic Geometry Software With a Proof Assistant for Interactive Formal Proofs*. *Electron. Notes Theor. Comput. Sci.* 285, pp. 43–55, doi:10.1016/j.entcs.2012.06.005.

[24] Andrew Polonsky (2011): *Proofs, Types and Lambda Calculus*. Ph.D. thesis, University of Bergen.

[25] G. Sutcliffe (2009): *The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0*. *Journal of Automated Reasoning* 43(4), pp. 337–362, doi:10.1007/s10817-009-9143-8.

[26] Steven Vickers (1993): *Geometric Logic in Computer Science*. In: *Theory and Formal Methods*, Workshops in Computing, Springer, pp. 37–54, doi:10.1007/978-1-4471-3503-6_4.

[27] Dongming Wang (2003): *Automated Generation of Diagrams with Maple and Java*. In Michael Joswig & Nobuki Takayama, editors: *Algebra, Geometry and Software Systems*, Springer, Berlin, Heidelberg, pp. 277–287, doi:10.1007/978-3-662-05148-1_15.

[28] Sean Wilson & Jacques D. Fleuriot (2005): *Combining Dynamic Geometry, Automated Geometry Theorem Proving and Diagrammatic Proofs*. In: *ETAPS Satellite Workshop on User Interfaces for Theorem Provers (UITP)*, Springer, Edinburgh.

[29] Daniel Winterstein (2004): *Dr.Doodle: A Diagrammatic Theorem Prover*. In: *Proceedings of IJCAR 2004*, doi:10.1007/978-3-540-25984-8_24.

[30] Zheng Ye, Shang-Ching Chou & Xiao-Shan Gao (2010): *Visually Dynamic Presentation of Proofs in Plane Geometry*. *Journal of Automated Reasoning* 45(3), pp. 243–266, doi:10.1007/s10817-009-9163-4.

[31] Zheng Ye, Shang-Ching Chou & Xiao-Shan Gao (2010): *Visually Dynamic Presentation of Proofs in Plane Geometry, Part 1*. *J. Autom. Reason.* 45(3), pp. 213–241, doi:10.1007/s10817-009-9162-5.

[32] Zheng Ye, Shang-Ching Chou & Xiao-Shan Gao (2011): *An Introduction to Java Geometry Expert*. In: *Post-proceedings of Automated Deduction in Geometry (ADG 2008)*, *Lecture Notes in Computer Science* 6301, Springer-Verlag, pp. 189–195, doi:10.1007/978-3-642-21046-4_10.

A Example: Varignon's theorem

In this section, we provide as examples two different proofs of a statement in Euclidean geometry called Varignon's theorem.

Theorem 2 *th_varignon :* $\forall A \ \forall B \ \forall C \ \forall D \ \forall I \ \forall J \ \forall K \ \forall L \ (\neg col(B,D,A) \wedge \neg col(B,D,C) \wedge \neg col(A,C,B) \wedge \neg col(A,C,D) \wedge \neg col(I,J,K) \wedge B \neq D \wedge A \neq C \wedge midpoint(A,I,B) \wedge midpoint(B,J,C) \wedge midpoint(C,K,D) \wedge midpoint(A,L,D) \Rightarrow pG(I,J,K,L) \ )$

First Proof:

Consider arbitrary a,b,c,d,e,f,g,h such that: $\neg col(b,d,a)$, $\neg col(b,d,c)$, $\neg col(a,c,b)$, $\neg col(a,c,d)$, $\neg col(e,f,g)$, $b \neq d$, $a \neq c$, $midpoint(a,e,b)$, $midpoint(b,f,c)$, $midpoint(c,g,d)$, $midpoint(a,h,d)$. It should be proved that $pG(e,f,g,h)$.

  1. $par(b,d,f,g)$ (by MP, from $\neg col(b,d,c)$, $midpoint(c,g,d)$, $midpoint(b,f,c)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto b$, $B \mapsto d$, $C \mapsto c$, $P \mapsto g$, $Q \mapsto f$)

  2. $par(b,d,e,h)$ (by MP, from $\neg col(b,d,a)$, $midpoint(a,h,d)$, $midpoint(a,e,b)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto b$, $B \mapsto d$, $C \mapsto a$, $P \mapsto h$, $Q \mapsto e$)

  3. $par(a,c,e,f)$ (by MP, from $\neg col(a,c,b)$, $midpoint(b,f,c)$, $midpoint(a,e,b)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto b$, $P \mapsto f$, $Q \mapsto e$)

  4. $par(a,c,h,g)$ (by MP, from $\neg col(a,c,d)$, $midpoint(c,g,d)$, $midpoint(a,h,d)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto d$, $P \mapsto g$, $Q \mapsto h$)
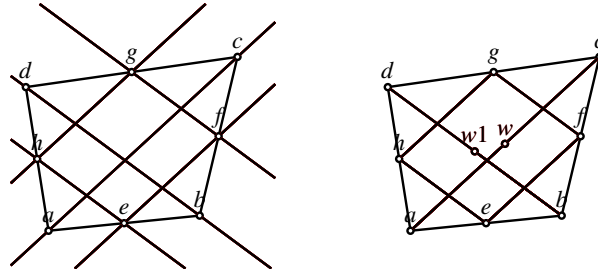
Figure 2: Illustrations for two proofs of Varignon's theorem

5. $par(e,f,g,h)$ (by MP, from $par(a,c,e,f)$, $par(a,c,h,g)$, $\neg col(e,f,g)$ using axiom lemma_par_trans; instantiation: $A \mapsto e$, $B \mapsto f$, $C \mapsto a$, $D \mapsto c$, $E \mapsto g$, $F \mapsto h$)

6. $par(f,g,e,h)$ (by MP, from $par(b,d,f,g)$, $par(b,d,e,h)$, $par(e,f,g,h)$ using axiom lemma_par_trans; instantiation: $A \mapsto f$, $B \mapsto g$, $C \mapsto d$, $D \mapsto b$, $E \mapsto e$, $F \mapsto h$)

7. $pG(e,f,g,h)$ (by MP, from $par(e,f,g,h)$, $par(e,f,g,h)$, $par(f,g,e,h)$ using axiom lemma_par2_pg; instantiation: $A \mapsto e$, $B \mapsto f$, $C \mapsto g$, $D \mapsto h$)

8. Proved by assumption! (by QEDas)

Second Proof:

Consider arbitrary a, b, c, d, e, f, g, h such that: $f \neq h$, $b \neq d$, $d \neq b$, $a \neq c$, $c \neq a$, $midpoint(a,e,b)$, $midpoint(b,f,c)$, $midpoint(c,g,d)$, $midpoint(a,h,d)$. It should be proved that $pG(e,f,g,h)$.

1. Let w be such that $midpoint(a,w,c)$ (by MP, from $a \neq c$ using axiom lemma_midpoint_existence; instantiation: $A \mapsto a$, $B \mapsto c$)

2. Let $w1$ be such that $midpoint(d,w1,b)$ (by MP, from $d \neq b$ using axiom lemma_midpoint_existence; instantiation: $A \mapsto d$, $B \mapsto b$)

3. $cong(a,w,g,h)$ (by MP, from $c \neq a$, $midpoint(a,w,c)$, $midpoint(c,g,d)$, $midpoint(a,h,d)$ using axiom lemma_triangle_mid_par_cong_1; instantiation: $A \mapsto d$, $B \mapsto a$, $C \mapsto c$, $P \mapsto w$, $Q \mapsto g$, $R \mapsto h$)

4. $cong(a,w,f,e)$ (by MP, from $c \neq a$, $midpoint(a,w,c)$, $midpoint(b,f,c)$, $midpoint(a,e,b)$ using axiom lemma_triangle_mid_par_cong_1; instantiation: $A \mapsto b$, $B \mapsto a$, $C \mapsto c$, $P \mapsto w$, $Q \mapsto f$, $R \mapsto e$)

5. $tP(c,a,h,g)$ (by MP, from $a \neq c$, $midpoint(a,w,c)$, $midpoint(a,h,d)$, $midpoint(c,g,d)$ using axiom lemma_triangle_mid_par_cong_1; instantiation: $A \mapsto d$, $B \mapsto c$, $C \mapsto a$, $P \mapsto w$, $Q \mapsto h$, $R \mapsto g$)

6. $tP(b,d,h,e)$ (by MP, from $d \neq b$, $midpoint(d,w1,b)$, $midpoint(a,h,d)$, $midpoint(a,e,b)$ using axiom lemma_triangle_mid_par_cong_1; instantiation: $A \mapsto a$, $B \mapsto b$, $C \mapsto d$, $P \mapsto w1$, $Q \mapsto h$, $R \mapsto e$)

7. $cong(b,w1,h,e)$ (by MP, from $d \neq b$, $midpoint(d,w1,b)$, $midpoint(a,h,d)$, $midpoint(a,e,b)$ using axiom lemma_triangle_mid_par_cong_1; instantiation: $A \mapsto a$, $B \mapsto b$, $C \mapsto d$, $P \mapsto w1$, $Q \mapsto h$, $R \mapsto e$)

8. $cong(h,e,h,e)$ (by MP, from $cong(b,w1,h,e)$, $cong(b,w1,h,e)$ using axiom lemma_congruencetransitive; instantiation: $A \mapsto h$, $B \mapsto e$, $C \mapsto b$, $D \mapsto w1$, $E \mapsto h$, $F \mapsto e$)

9. $cong(g,h,e,f)$ (by MP, from $cong(a,w,g,h)$, $cong(a,w,f,e)$ using axiom lemma_congruencetransitive; instantiation: $A \mapsto g$, $B \mapsto h$, $C \mapsto w$, $D \mapsto a$, $E \mapsto e$, $F \mapsto f$)

10. $cong(b,w1,g,f)$ (by MP, from $d \neq b$, $midpoint(d,w1,b)$, $midpoint(c,g,d)$, $midpoint(b,f,c)$ using axiom lemma_triangle_mid_par_cong_1; instantiation: $A \mapsto c$, $B \mapsto b$, $C \mapsto d$, $P \mapsto w1$, $Q \mapsto g$, $R \mapsto f$)

11. $cong(f,g,e,h)$ (by MP, from $cong(b,w1,g,f)$, $cong(b,w1,h,e)$ using axiom lemma_congruencetransitive; instantiation: $A \mapsto f$, $B \mapsto g$, $C \mapsto b$, $D \mapsto w1$, $E \mapsto e$, $F \mapsto h$)

12. $tP(b,d,g,f)$ (by MP, from $d \neq b$, $midpoint(d,w1,b)$, $midpoint(c,g,d)$, $midpoint(b,f,c)$ using axiom lemma_triangle_mid_par_cong_1; instantiation: $A \mapsto c$, $B \mapsto b$, $C \mapsto d$, $P \mapsto w1$, $Q \mapsto g$, $R \mapsto f$)

13. $tP(a,c,f,e)$ (by MP, from $c \neq a$, $midpoint(a,w,c)$, $midpoint(b,f,c)$, $midpoint(a,e,b)$ using axiom lemma_triangle_mid_par_cong_1; instantiation: $A \mapsto b$, $B \mapsto a$, $C \mapsto c$, $P \mapsto w$, $Q \mapsto f$, $R \mapsto e$)

14. $tP(g,h,f,e)$ (by MP, from $tP(c,a,h,g)$, $tP(a,c,f,e)$ using axiom lemma_tP_trans; instantiation: $A \mapsto g$, $B \mapsto h$, $C \mapsto a$, $D \mapsto c$, $E \mapsto f$, $F \mapsto e$)

15. $tP(f,g,e,h)$ (by MP, from $tP(b,d,g,f)$, $tP(b,d,h,e)$ using axiom lemma_tP_trans; instantiation: $A \mapsto f$, $B \mapsto g$, $C \mapsto b$, $D \mapsto d$, $E \mapsto e$, $F \mapsto h$)

16. $pG(e,f,g,h)$ (by MP, from $f \neq h$, $cong(g,h,e,f)$, $cong(f,g,e,h)$, $tP(f,g,e,h)$, $tP(g,h,f,e)$ using axiom lemma_par_par_cong_cong_parallelogram; instantiation: $A \mapsto e$, $B \mapsto f$, $C \mapsto g$, $D \mapsto h$)

17. Proved by assumption! (by QEDas)