A reflexive tactic for automated generation of proofs of incidence to an affine variety

Pierre Boutry, Julien Narboux, Pascal Schreck

Preprint version.

A reflexive tactic for automated generation of proofs of incidence to an affine variety

Pierre Boutry Julien Narboux Pascal Schreck

ICube, UMR 7357 CNRS, University of Strasbourg

{boutry,narboux,schreck}@unistra.fr

Abstract

This paper describes the formalization and implementation of a reflexive tactic for automated generation of proofs of incidence to an affine variety. Incidence proofs occur frequently in formal proofs of geometric statements. Nevertheless they are most of the time omitted in pen-and-paper proofs since they do not contribute to the understanding of the proof in which they appear. Our tactic allows us to automate proofs about incidence to an affine variety. Being based on a type class capturing the minimal set of properties needed to deal with incidence, the tactic is applicable to any theory verifying these properties. This type class is defined using dependent types to formalize predicates of parameterizable arity which represent the incidence to an affine variety.

Keywords Automation, computational reflection, geometry, incidence, formalization, Coq

1. Introduction

The following research was conducted as part of a larger project which focuses on the use of proof assistants in an educational framework to teach reasoning principles. We believe that geometry is a good subject to study in order to understand the proof process. Our long term goal consists in the development of an interactive software based on Coq in which students can write their own proofs. In the context of education, Hilbert's and Tarski's axiomatic developments possess the advantageous quality of not being based on set-theoretical notions. As a basis for these tools and libraries, Tarski's system of geometry was chosen for its well-known metamathematical properties. Yet the absence of set-theoretical notions has its drawback.

For instance, it induces incidence proofs to become particularly tedious. This issue also arises in Hilbert's axiomatic development while straight lines are nevertheless considered. To illustrate how often incidence proofs occur we made some statistics. We studied our formal development (GeoCoq) of Tarski's geometry which is mainly a formalization of [25]. Approximately one seventh of the lines of the proof script contains applications of lemmas about collinearity of points. And almost a third of the lemmas of our development have as hypothesis the collinearity of some given points. One should point out two facts which allowed us to lower the ratio of incidence proofs present in our development. Firstly most of the incidence proofs are produced using some automatic tactics, therefore their length is greatly reduced. Secondly we restricted ourselves to the formalization of two-dimensional euclidean geometry. Thus the greater part of the incidence proofs corresponds to proofs about collinearity.

The particularity of incidence proofs is that they are omitted in pen-and-paper proofs while they are subject to combinatorial explosion. These proofs are omitted as they do not contribute to the understanding of the proof in which they appear. This particularity calls for a procedure to automate these proofs. In this paper we describe the reflexive tactic we developed to deal with this issue. Our early version of the tactic was specifically conceived to handle the case of collinearity. We then realized that our approach could be generalized in order to deal with incidence to an affine variety. As well as automating the incidence proofs our tactic allowed us to achieve a higher readability of our proof scripts.

Geometry is a successful area of the field of automated theorem proving. Several efficient methods have been developed. The most popular ones are the Gröbner basis method [8], Wu's method [9, 31, 32], the area method of Chou, Gao and Zhang [10], geometric algebra [18]. It is to be noticed that a decision procedure for the theory we are using was given by Tarski [29]. Some of these methods have been formalized in Coq: the second author formalized the area method [16, 21], Pottier and Théry formalized the Gröbner basis method [14, 24, 30], Genevaux et al. extended this work to Wu's method [13], Fuchs and Théry formalized a procedure based on geometric algebra [12]. But in this paper, as we aim at applications in the education, we are not interested in obtaining the most powerful prover which automates the whole proof. On the contrary we want to automate only the proof steps which are usually implicit in a pen-and-paper proof. The basis for this work is the mechanization in Coq of Tarski's axiomatic development about geometry. This library called GeoCoq has already been described in our previous work [5–7, 22]. But for the sake of modularity we defined a type class capturing the minimal set of properties needed to apply our tactic. Our tactic is then applicable to any theory verifying these properties and is thus not restricted to Tarski's system of geometry.

Our work share some motivations with the work of Phil Scott and Jacques Fleuriot [26, 27]. They propose a framework to add domain-specific automation and apply it to the case study of Hilbert's geometry. Their approach consists in using the idle computation time to generate new facts using a given set of lemmas in a forward manner. Our method is different, it is specific for incidence problems. Hence it is more efficient because as we know the kind of data we manipulate, we can use a suitable data structure.

Automating the proof steps that are implicit in Euclid's proofs is done by Avigad et al. [1].

The rest of the paper is organized as follows. Section 2 describes the issue in the context of a simple example in elementary geometry. Section 3 presents a reflexive tactic to deal with the pseudo-transitivity of the collinearity predicate. In section 4, we generalize our approach to other predicates.

2. Illustration of the issue through a simple example

For the sake of clarity, we present the simple example of the midpoints theorem. We will present a proof which contains an incidence proof. This will allow us to illustrate the issue that arises

while doing an incidence proof. The statement of this theorem is the following.

THEOREM 1. *In a non-degenerate triangle ABC where P is the midpoint of segment BC and Q the midpoint of segment AC, the lines AB and PQ are strictly parallel* [1].

First we give the (slightly incorrect) informal proof which is often given in class.

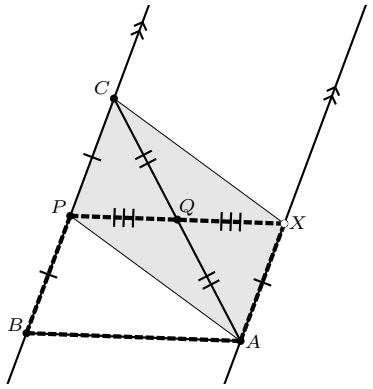

Figure 1. The midpoints theorem.

PROOF 1. *We first construct point X as the symmetric point of P with respect to Q. Point Q is therefore the midpoint of segment PX. From the assumptions we know that Q is also the midpoint of AC. Thus, the diagonals of the quadrilateral $APCX$ bisect in their midpoint and hence $APCX$ is a parallelogram. Now according to the fact that the opposite sides of a parallelogram are parallel and have the same length, we have that AX and CP (or BP) are parallel and $AX \equiv CP$. As we know that P is the midpoint of BC we also have $AX \equiv PB$. The quadrilateral $AXPB$ has two opposite sides which are parallel and of the same length, hence it is a parallelogram. Finally the opposite sides of a parallelogram are parallel, thus AB and PQ are parallel.* □

Formalizing this simple proof is not as trivial as it seems.

Firstly, for a good reason: actually, this proof is not correct because from the fact that the opposite sides of a quadrilateral $ABCD$ are parallel and of the same length we can only conclude that either $ABCD$ or $ABDC$ is a parallelogram. Proving that one is a parallelogram rather than the other is not trivial and should not be omitted. The possibility of overlooking the fact that in order to prove that $ABCD$ is a parallelogram we also need to prove that $ABDC$ cannot be a parallelogram motivates the use of a proof assistant.

Secondly, for some bad reasons: this proof contains proof steps which are implicit in a pen-and-paper proof that a proof assistant forces us to detail. Degenerate cases appear extremely often in geometry and correspond to cases of incidence of two geometric objects. These cases generally do not appear in pen-and-paper and contribute greatly to the difficulty of generating a proof of a geometric statement. Either the statement that we wish to prove holds in the degenerate case. Then one usually needs to prove the incidence of other pairs of geometric objects in order to prove the statement in the degenerate case. Or it does not and then an extra hypothesis is needed. This hypothesis is often referred to as a *non-degeneracy condition* and corresponds to the negation of the incidence of two geometric objects. This extra hypothesis will generate a proof obligation which will often be proven by proof of negation[2]. The contradiction will then be shown through an incidence proof.

Therefore handling degenerate cases will most of the time result in incidence proofs. The proof of midpoints theorem features such handling of degenerate cases. Indeed one cannot directly deduce that lines AB and PQ are parallel from the fact that $ABPX$ is a parallelogram. In fact this statement is only true when the parallelogram is non-flat. Thus one needs to prove that three of the four points defining the parallelogram are non-collinear. So let us prove A, P and X are non-collinear by proof of negation. Assuming that A, P and X are collinear one can obtain a contradiction by proving that A, B and C are collinear with following proof script.

```
apply col_permut231; apply col123_124__col234 with P;
[| |apply col_permut231]; auto.
apply col_permut231;apply col123_124__col134 with Q;auto.
apply col_permut231;apply col123_124__col134 with x;
[|apply col_permut321|apply col_permut132]; auto.
```

In this script there are six occurrences of lemmas dealing with permutation properties of the predicate Col designating the collinearity of an ordered triple of points. These lemmas correspond to

$$\forall ABC\sigma, \sigma \in S_{\{A,B,C\}} \Rightarrow \mathrm{Col}\ A\,B\,C \Rightarrow \mathrm{Col}\ \sigma(A)\,\sigma(B)\,\sigma(C)$$

where S_X denotes the symmetric group on a finite set X. To avoid the definition of the symmetric group we proved one lemma for each element of the group but the identity element.

Moreover there are three occurrences of lemmas handling the pseudo-transitivity of this same predicate. These lemmas correspond to

$$\forall PQABC, P \neq Q \Rightarrow \mathrm{Col}\ P\,Q\,A \Rightarrow \mathrm{Col}\ P\,Q\,B \Rightarrow \mathrm{Col}\ P\,Q\,C \Rightarrow \mathrm{Col}\ A\,B\,C$$

where A was chosen so that the first collinearity is trivially satisfied from

$$\forall AB, \mathrm{Col}\ A\,A\,B$$

which expresses that two points are always collinear. The lemma of pseudo-transitivity can be understood as the possibility of proving that three points are collinear if they all belong to the same line. The extra hypothesis is needed to ensure that the points really belong to a well-defined line.

Even if we believe that in the context of education it is important to mention that the parallelogram should be non-flat we would like the incidence proof corresponding to the proof of negation to be done automatically. Indeed besides corresponding to proof steps that should be implicit as they do not bring any understanding of the proof, its proof script is tedious to produce because of the combinatorics underlying the pseudo-transitivity of collinearity. In the next section we expose the reflexive tactic we developed to automatically prove collinearity properties.

[1] The common way of defining parallelism is to consider two lines as parallel if they belong to the same plane but do not meet. We distinguish between strict parallelism which corresponds to the previous definition and parallelism which add the possibility for the lines to be equal.

[2] We use the expression 'proof of negation' to describe a proof of $\neg A$ assuming A and obtaining a contradiction.

3. A reflexive tactic for dealing with permutation properties and pseudo-transitivity of collinearity

In order to simplify the proving process and to improve readability, we defined a tactic which can prove automatically collinearity properties which are consequences of this pseudo-transitivity.

Our first approaches to deal with this problem were to use the built-in automation of Coq (by creating a base of hints for the Coq tactic `eauto`) and then to write an *ad hoc* tactic in the tactic language of Coq. However this approach was not fulfilling our needs as it could not cope with difficult problems in a reasonable time. We therefore opted for a different approach. We chose to implement a reflexive tactic to handle this problem. First introduced in Coq by Samuel Boutin [4], reflexivity consists in replacing a tactic by an algorithm written in the Coq language and proving that the algorithm is sound. The reader interested in learning more about this now standard approach can also read the last chapter of the Coq'Art [3]. Using a reflective tactic allows us not only to save the user from doing the tedious work about the pseudo-transitivity of collinearity but also it hides these steps from the proof term. The algorithm used by this tactic is described in the following paragraph.

3.1 Algorithm

The algorithm is divided into three parts. The first one consists in the initialization phase: it computes the set of all the sets of points known collinear and the sets of pairs of points known distinct. The second part consists in updating our internal data structure to compute the sets of points on each line. Finally we check if three given points are collinear by testing if they belong to a single line. For our algorithm we need a set of sets of points $\mathcal{L}$ to represent the sets of points known to be collinear and a set of pairs of points $\mathcal{D}$ to represent the points known to be different.

The algorithm is as follows:
INPUT: 3 points A B C and the current hypotheses.

1. Initialize $\mathcal{L}$ so that it contains all sets of three points that are assumed to be collinear by the hypotheses in the context and $\mathcal{D}$ so that it contains all the pairs of points that are assumed to be different by the hypotheses in the context.
2. For every l_1 and l_2 in $\mathcal{L}$ such that there exists a pair (p_1, p_2) in $\mathcal{D}$ such that $p_1 \in l_1 \cap l_2$ and $p_2 \in l_1 \cap l_2$, replace $\mathcal{L}$ with $((\mathcal{L} \backslash l_1) \backslash l_2) \cup \{l_1 \cup l_2\}$[3] until there are no such l_1 and l_2.
3. If there is a set l in $\mathcal{L}$ such that $A \in l$, $B \in l$ and $C \in l$ then A, B and C are collinear.

Remark. Our tactic only captures basic properties of incidence, and is complete for only a small theory described below. Indeed, it can happen that some points A, B and C are collinear (if this fact follows from other geometric theorems) and our tactic fails in yielding a set $l \in \mathcal{L}$ such that $A \in l$, $B \in l$ and $C \in l$.

3.2 Implementation

We now give some technical details about the implementation in Coq of our algorithm.

3.2.1 Data-structures

We need to represent sets of sets of points. To represent points we need a decidable ordered type, hence we use the type of positive numbers as key. To represent finite sets we use the module `Msets` of the standard library. We could have used the library `Containers` [17] which is easier to use than `Msets` because it infers automatically the structures needed to build the finite sets but we have chosen to keep the standard `Msets` to make our development easier to install as it is included in the standard distribution. We selected the implementation using ordered lists. Notice that using AVLs is not interesting here since we rarely have more than thirty points.

3.2.2 The tactic

First, our tactic follows the first step of our previous algorithm in order to build the sets $\mathcal{L}$ and $\mathcal{D}$ by using an associative list so that the positives in our structures identify points. This initialization phase is implemented using the tactic language of Coq.

The second step is implemented as a Coq function defined using the `Function` package of Coq [2]. To convince Coq that the algorithm terminates we proved the fact that the cardinality of $\mathcal{L}$ decreases at each recursive call.

The third step is also implemented as a Coq function which searches for a triple of points in a same set contained in $\mathcal{L}$.

3.2.3 Proof of the soundness of our tactic

For the sake of modularity, we created a type class with the minimal set of properties that a theory needs to verify and we did all the proofs within the context of this type class. The type class mechanism allows us to state axiom systems and to use implicitly the proof that some theory is a model of this axiom system. Type classes are dependent records with some automation: Coq infers some implicit instances [28]. Our tactic is then applicable to any theory verifying these following four properties (our own development about Tarski's geometry but also, for example, the developments of Frédérique Guilhot [15], or Jean Duprat [11]) [4]:

```
Class Col_theory (CTpoint:Type)
             (CTCol:CTpoint->CTpoint->CTpoint->Prop):=
{
 CTcol_trivial : forall A B : CTpoint, CTCol A A B;
 CTcol_perm_1 : forall A B C : CTpoint, CTCol A B C ->
                                          CTCol B C A;
 CTcol_perm_2 : forall A B C : CTpoint, CTCol A B C ->
                                          CTCol A C B;
 CTcol3 : forall X Y A B C : CTpoint,
  X<>Y -> CTCol X Y A -> CTCol X Y B -> CTCol X Y C ->
                                          CTCol A B C
}.
```

We want to prove that the tactic produces a $\mathcal{L}$ that verifies the property "*any triple of points belonging to a set of $\mathcal{L}$ are provably collinear*". To do so we prove that the $\mathcal{L}$ produced by the first step of our algorithm verifies this property and that the second step of the algorithm preserves this property. The original $\mathcal{L}$ trivially verifies this property by construction. We denote by $\overline{x}$ the point represented by the positive integer x. Now assuming that we have l_1, l_2, p_1 and p_2 verifying $p_1 \in l_1 \cap l_2$, $p_2 \in l_1 \cap l_2$ and $(p_1, p_2) \in \mathcal{D}$. Assuming that the interpretation of any triple of points in l_1 are provably collinear and assuming the same for l_2 then for any p_3 in l_1, $Col\ \overline{p_1}\ \overline{p_2}\ \overline{p_3}$ holds and for any p_4 in l_2, $Col\ \overline{p_1}\ \overline{p_2}\ \overline{p_4}$ holds. By the lemma stated previously (`CTcol3`) the interpretation of any triple of points in $l_1 \cup l_2$ are provably collinear. This proves that the second step of our algorithm preserves the property stated above and at the end of the second step we will obtain a set $\mathcal{L}$ verifying this property.

In Coq, the function $x \mapsto \overline{x}$ is called `interp`. The functions to manipulate sets of sets of positives are prefixed by `SS`, the functions

[3] One should remember that we are manipulating sets of sets here.

[4] In order to capture that $\forall ABC\sigma, \sigma \in S_{\{A,B,C\}} \Rightarrow \mathrm{Col}\ A\,B\,C \Rightarrow \mathrm{Col}\ \sigma(A)\,\sigma(B)\,\sigma(C)$ it suffices that this proposition holds for the generators of $S_{\{A,B,C\}}$, namely $(A\ B)$ and $(A\ B\ C)$.

for sets of positives are prefixed by `S` and the functions for sets of pairs of positives are prefixed by `ST`.

We define a predicate expressing that our set of lines is correct; for every line, all points on this line are collinear:

```
Definition ss_ok (ss : SS.t)
                 (interp: positive -> COLTpoint) :=
  forall s, SS.mem s ss = true ->
  forall p1 p2 p3, S.mem p1 s &&
                   S.mem p2 s &&
                   S.mem p3 s = true ->
    CTCol (interp p1) (interp p2) (interp p3).
```

We also need a predicate expressing that our set of pairs of distinct points is correct; all pairs are distinct:

```
Definition sp_ok (sp : SP.t)
                 (interp: positive -> COLTpoint) :=
  forall p, SP.mem p sp = true ->
              interp (fstpp p) <> interp (sndpp p).
```

Finally, we prove that our main function `test_col` (which tests if 3 points belong to the same set $s \in \mathcal{L}$ after applying our algorithm on $\mathcal{L}$ and $\mathcal{D}$) is correct assuming that we start in a correct context:

```
Lemma test_col_ok : forall ss sp interp p1 p2 p3,
  ss_ok ss interp -> sp_ok sp interp ->
  test_col ss sp p1 p2 p3 = true ->
  CTCol (interp p1) (interp p2) (interp p3).
```

For the reification phase, we repeat the application of the following lemma, which states that if we know that two points A and B are distinct we can add them to the list of pairs of distinct points:

```
Lemma collect_diffs :
  forall (A B : COLTpoint)
         (H : A <> B)
         (pa pb sp : positive)
         (interp : positive -> COLTpoint),
  interp pa = A ->
  interp pb = B ->
  sp_ok sp interp -> sp_ok (SP.add (pa, pb) sp) interp.
```

We have a similar lemma to reify collinearity assumptions:

```
Lemma collect_cols :
  forall (A B C : COLTpoint)
         (HCol : CTCol A B C)
         (pa pb pc : positive) ss
         (interp :  positive -> COLTpoint),
  interp pa = A ->
  interp pb = B ->
  interp pc = C ->
  ss_ok ss interp ->
  ss_ok (SS.add (S.add pa (S.add pb
                       (S.add pc S.empty))) ss) interp.
```

3.3 Relation with equality of lines and rank functions

If we allow ourself the concept of line (either by defining it with Tarski's language as we have done in [6] or by using another language for geometry which includes lines such as Hilbert's axioms), then we can rewrite the pseudo-transitivity property of Col as an equality properties about lines:

$$A \neq B \wedge A \in l \wedge B \in l \wedge A \in m \wedge B \in m \Rightarrow l = m.$$

At first sight this property looks nicer than our properties about Col, but the problem with this formulation is that lines are always defined by pairs of distinct points. In practice using this kind of formulation would imply numerous case distinctions about equality of points.

There is a close link between the concept of rank that we formalized previously [19] and the properties studied in this section. The rank r of a subset S of elements of the matroid of points is the maximum size of an independent subset of S. Notice that the sub-modularity property of the rank function is a generalization of the pseudo-transitivity of Col:

$$r(l \cup m) + r(l \cap m) \leq r(l) + r(m).$$

Indeed, if l and m are lines then their rank are of 2, and if their intersection contains two distinct points then the rank of the intersection is at least of 2, hence all points in the union are collinear:

$$r(l \cup m) \leq 2 + 2 - 2 = 2.$$

4. Generalization

The algorithm presented in the previous section may seem to be very specific. In fact, it can be generalized to deal with other properties than pseudo-transitivity of collinearity. For example, the lemma to express the pseudo-transitivity of the coplanar predicate has the same form:

$$\begin{aligned}\forall ABCDPQR,\ \neg Col\ P\ Q\ R\ \Rightarrow\\ Coplanar\ P\ Q\ R\ A \Rightarrow Coplanar\ P\ Q\ R\ B \Rightarrow\\ Coplanar\ P\ Q\ R\ C \Rightarrow Coplanar\ P\ Q\ R\ D \Rightarrow\\ Coplanar\ A\ B\ C\ D.\end{aligned}$$

And the lemma to express the pseudo-transitivity of the concyclic predicate has the same form:

$$\begin{aligned}\forall ABCDPQ,\ \neg Col\ P\ Q\ R \Rightarrow\\ Concyclic\ P\ Q\ R\ A \Rightarrow Concyclic\ P\ Q\ R\ B \Rightarrow\\ Concyclic\ P\ Q\ R\ C \Rightarrow Concyclic\ P\ Q\ R\ D \Rightarrow\\ Concyclic\ A\ B\ C\ D.\end{aligned}$$

In fact, our tactic is generalizable to any incidence relationship with algebraic curves or affine varieties.

4.1 The tactic

We use an axiomatic approach to define the generalized tactic.

The generalization holds for any predicate `wd` of arity $n+2$ and `coinc` of arity $n+3$ for some n which verify the following axioms. Intuitively, `wd` predicate express the non degeneracy condition, and `coinc` the incidence relation.

We assume that the `wd` and `coinc` predicates are invariant by permutation [5]:

$$\forall X_1 X_2 \ldots X_{n+2}, wd\ X_1\ X_2\ \ldots\ X_{n+2} \Rightarrow wd\ X_2\ \ldots\ X_{n+2}\ X_1 \quad (1)$$

[5] Exactly as for `Col_theory`, to capture the permutation properties of `wd` and `coinc`, it suffices that the permutation properties hold for the generators of $S_{\{X_1,X_2,\ldots,X_{n+2}\}}$ and $S_{\{X_1,X_2,\ldots,X_{n+3}\}}$, respectively.

$$\forall X_1 X_2 \ldots X_{n+2}, wd\ X_1\ X_2\ X_3\ \ldots\ X_{n+2} \Rightarrow wd\ X_2\ X_1\ X_3\ \ldots\ X_{n+2} \quad (2)$$

$$\forall X_1 X_2 \ldots X_{n+3}, coinc\ X_1\ X_2\ \ldots\ X_{n+3} \Rightarrow coinc\ X_2\ \ldots\ X_{n+3}\ X_1 \quad (3)$$

$$\forall X_1 X_2 \ldots X_{n+3}, coinc\ X_1\ X_2\ X_3\ \ldots\ X_{n+3} \Rightarrow coinc\ X_2\ X_1\ X_3\ \ldots\ X_{n+3}. \quad (4)$$

Moreover, we admit that the `coinc` predicates trivially holds when two points are equal:

$$\forall A X_1 X_2 \ldots X_{n+1}, coinc\ A\ A\ X_1\ \ldots\ X_{n+1}. \quad (5)$$

Finally we need that the pseudo-transitivity property holds:

$$\forall X_1 X_2 \ldots X_{n+2} P_1 P_2 \ldots P_{n+3}, wd\ X_1\ \ldots\ X_{n+2} \wedge \bigwedge_{i=1}^{n+3} coinc\ X_1\ \ldots\ X_{n+2}\ P_i \Rightarrow coinc\ P_1\ \ldots\ P_{n+3}. \quad (6)$$

In Coq's syntax, we can express this axiom system, the dots are replaced by dependent types:

We define a class `Arity` which contains two fields:

- the type of the points we will consider,
- the natural number n such that $n+2$ is the arity of the `wd` (then $n+3$ will be the arity of `coinc`):

```
Class Arity :=
{
  COINCpoint : Type;
  n : nat
}.
```

`Coinc_predicates` inherits from `Arity` and contains one field for each predicate:

```
Class Coinc_predicates (Ar : Arity) :=
{
  wd : arity COINCpoint (S (S n));
  coinc : arity COINCpoint (S (S (S n)))
}.
```

The predicates are elements of the type `arity T n` representing predicates of type $\underbrace{T \rightarrow \ldots \rightarrow T}_{n \text{ times}} \rightarrow$ Prop. Its formal definition is the following:

```
Fixpoint arity (T:Type) (n:nat) :=
 match n with
 | 0 => Prop
 | S p => T -> arity T p
 end.
```

One can then define the type class corresponding to axioms (1-6):

```
Class Coinc_theory (Ar : Arity)
                   (COP : Coinc_predicates Ar) :=
{
  wd_perm_1 :
    forall A : COINCpoint,
    forall X : cartesianPower COINCpoint (S n),
               app_1_n wd A X ->
               app_n_1 wd X A;
  wd_perm_2 :
    forall A B : COINCpoint,
    forall X : cartesianPower COINCpoint n,
               app_2_n wd A B X ->
               app_2_n wd B A X;
  coinc_perm_1 :
    forall A : COINCpoint,
    forall X : cartesianPower COINCpoint (S (S n)),
               app_1_n coinc A X ->
               app_n_1 coinc X A;
  coinc_perm_2 :
    forall A B : COINCpoint,
    forall X : cartesianPower COINCpoint (S n),
               app_2_n coinc A B X ->
               app_2_n coinc B A X;
  coinc_bd :
    forall A : COINCpoint,
    forall X : cartesianPower COINCpoint (S n),
               app_2_n coinc A A X;
  coinc_n :
    forall COINC : cartesianPower COINCpoint (S(S(S n))),
    forall WD : cartesianPower COINCpoint (S (S n)),
                pred_conj coinc COINC WD ->
                app wd WD ->
                app coinc COINC
}.
```

The type T^n is denoted by `cartesianPower T n`. The functions `app_1_n P X Xn`, allow the application of a predicate `P` of arity $n+1$ to be applied to an `X` of some type T and an `Xn` of type T^n. The functions `app_n_1` and `app_2_n` are similar. We denote the n-ary conjunction by `pred_conj`.

The rest of the implementation is similar to the one presented in the previous section. The set of pairs of distinct points is generalized by a set of tuples ordered using lexicographic order. The proofs are more technical because we replace proof in a particular case by inductive proofs for an arbitrary arity. We have shown that the two generators of the group of permutations induce all permutation properties. Some lemmas are needed to deal with the dependent types, we omit the details.

4.2 Deriving instances in Tarski's geometry

In the context of Tarski's geometry, we derived three instances of the `Coinc_theory`.

4.2.1 Collinearity

It is straightforward to instantiate our theory to obtain a tactic for collinearity as in Sec. 3. The proofs can be performed within Tarski's neutral dimensionless geometry (without assuming any upper-dimension axiom nor parallel postulate).

4.2.2 Coplanarity

The definition of coplanarity we adopt corresponds to the lemma 9.33 in [25]. It states that four points are coplanar if (at least) two out of these four points form a line which intersect the line formed by the remaining two points (see Fig.2):

```
Definition Coplanar A B C D :=
  exists X, (Col A B X /\ Col C D X) \/
            (Col A C X /\ Col B D X) \/
            (Col A D X /\ Col B C X).
```

The permutation properties (3-4) and trivial cases (5) of the predicate `Coplanar` are easy to obtain but the pseudo-transitivity

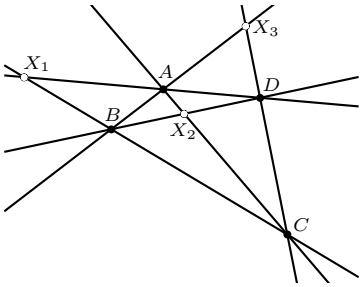

Figure 2. Definition of `Coplanar`.

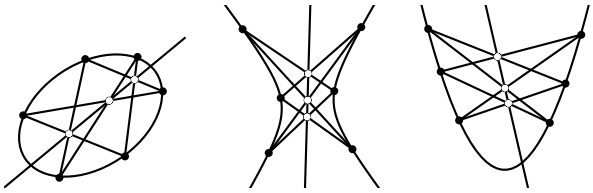

Figure 3. Pascal's hexagon theorem.

(6) requires the axiom of Pasch (both the inner and outer forms) and many case distinctions. The inner form of Pasch's axiom that we assume is a variant of the axiom Moritz Pasch introduced in [23] to repair the defects of Euclid. It intuitively says that if a line meets one side of a triangle and does not pass through the endpoints of that side, then it must meet one of the other sides of the triangle.

4.2.3 Concyclic

Our last instance allows to prove that points belong to the same circle. To define the concyclic predicate the first idea is to ask for the points to be coplanar and that there is a point (the center of the circle) which is at the same distance of the four points :

```
Definition Concyclic A B C D :=
 Coplanar A B C D /\
 exists O, Cong O A O B /\ Cong O A O C /\ Cong O A O D.
```

But in order to prove axiom (5), we need a more general definition because three points do not belong to a circle in case they are collinear. We will say that four points are concyclic-gen if either they are concyclic or if they all belong to the same line. Note that we do not assume the points to be distinct, so we need four collinearity assumptions to express this fact:

```
Definition Concyclic_gen A B C D :=
  Concyclic A B C D \/
  (Col A B C /\ Col A B D /\ Col A C D /\ Col B C D).
```

This generalized predicate allows us to instantiate the type class.

4.2.4 Other instances

It would be of interest to study the generalization of these instances to other cases such as conics, cubics or linear subspaces.

It is well-known that five points in general linear position define a plane conic. Pascal's hexagon theorem (see Fig.3) is a good mean to define conics geometrically. Assuming Pappus' theorem we have proved in [20] that the permutation properties of the predicate expressing that six points are coincident to a conic. It remains to show the pseudo-transitivity property.

5. Conclusions and Future Work

In this paper we described a generic reflexive tactic for solving some specific incidence properties which appear in high-school geometry. During this formalization we appreciated the versatility of the Calculus of Inductive Constructions (CIC) which allows to express easily functions of parameterizable arity.

Our tactic is generic in some sense, but also very specialized: it can solve a small class of goals. Yet, it would have been tedious to manually prove the goals which are solved automatically. Moreover, these sub-proofs are often hidden in an informal text since they are "obvious" and make the whole proof more difficult to read.

We believe that in order to obtain simple proofs, not only general progress about proof languages and automated deduction should be made but also some *ad hoc* techniques specialized in the domain of study should be used. For instance, in analysis, tactics to deal with proof involving epsilons could be used.

The tactic we obtain for dealing with co-planarity open also the path toward an extension of our formalization of geometry to higher dimension geometry.

Availability

The full Coq development can be found at the following url: http://geocoq.github.io/GeoCoq/

References

[1] J. Avigad, E. Dean, and J. Mumma. A formal system for euclids elements. *The Review of Symbolic Logic*, 2(04):700–768, 2009. URL http://arxiv.org/abs/0810.4315.

[2] G. Barthe, J. Forest, D. Pichardie, and V. Rusu. Defining and reasoning about recursive functions: a practical tool for the Coq proof assistant. In *Functional and Logic Programming (FLOPS'06)*, Fuji Susono, Japan, 2006. URL https://hal.inria.fr/inria-00564237.

[3] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development, Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[4] S. Boutin. Using reflection to build efficient and certified decision procedures. In M. Abadi and T. Ito, editors, *Theoretical Aspects of Computer Software*, volume 1281 of *Lecture Notes in Computer Science*, page 515529. Springer Berlin Heidelberg, 1997. ISBN 978-3-540-63388-4. . URL http://dx.doi.org/10.1007/BFb0014565.

[5] P. Boutry, J. Narboux, and P. Schreck. Parallel postulates and decidability of intersection of lines: a mechanized study within Tarski's system of geometry. submitted, July 2015. URL https://hal.inria.fr/hal-01178236.

[6] G. Braun and J. Narboux. From Tarski to Hilbert. In T. Ida and J. Fleuriot, editors, *Automated Deduction in Geometry 2012*, volume 7993, pages 89–109, Edinburgh, United Kingdom, Sept. 2012. Jacques Fleuriot, Springer. . URL https://hal.inria.fr/hal-00727117.

[7] G. Braun and J. Narboux. A synthetic proof of Pappus' theorem in Tarski's geometry. *Journal of Automated Reasoning*, 2015. URL https://hal.inria.fr/hal-01176508. accepted, revision pending.

[8] B. Buchberger and F. Winkler, editors. *Gröbner Bases and Applications*. Cambridge University Press, 1998.

[9] S.-C. Chou. *Mechanical Geometry Theorem Proving*. D. Reidel Publishing Company, 1988.

[10] S.-C. Chou, X.-S. Gao, and J.-Z. Zhang. *Machine Proofs in Geometry*. World Scientific, Singapore, 1994.

[11] J. Duprat. Fondements de géométrie euclidienne. 2010. URL https://hal.inria.fr/hal-00661537.

[12] L. Fuchs and L. Théry. A Formalization of Grassmann-Cayley Algebra in COQ and Its Application to Theorem Proving in Projective Geometry. In J. N. Pascal Schreck and J. Richter-Gebert, editors,

Automated Deduction in Geometry, ADG 2010, volume 6877 of *Lecture Notes in Computer Science*, pages 51–62, Munich, Germany, July 2010. Springer. . URL https://hal.archives-ouvertes.fr/hal-00657901.

[13] J.-D. Genevaux, J. Narboux, and P. Schreck. Formalization of Wu's simple method in Coq. In J.-P. Jouannaud and Z. Shao, editors, *CPP 2011 First International Conference on Certified Programs and Proofs*, volume 7086 of *Lecture Notes in Computer Science*, pages 71–86, Kenting, Taiwan, Dec. 2011. Springer-Verlag. . URL https://hal.inria.fr/inria-00618745.

[14] B. Grégoire, L. Pottier, and L. Théry. Proof Certificates for Algebra and Their Application to Automatic Geometry Theorem Proving. In T. Sturm and C. Zengler, editors, *Automated Deduction in Geometry*, volume 6301 of *Lecture Notes in Computer Science*, page 4259. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-21045-7. . URL http://dx.doi.org/10.1007/978-3-642-21046-4_3.

[15] F. Guilhot. Formalisation en coq et visualisation d'un cours de géométrie pour le lycée. *Technique et Science Informatiques*, 24(9): 1113–1138, 2005. . URL http://dx.doi.org/10.3166/tsi.24.1113-1138.

[16] P. Janicic, J. Narboux, and P. Quaresma. The Area Method : a Recapitulation. *Journal of Automated Reasoning*, 48(4):489–532, 2012. . URL https://hal.archives-ouvertes.fr/hal-00426563.

[17] S. Lescuyer. First-class containers in coq. *Stud. Inform. Univ.*, 9(1):87–127, 2011. URL http://studia.complexica.net/index.php?option=com_content&view=article&id=187%3Afirst-class-containers-in-coq-pp-87-127.

[18] H. Li. Automated Geometric Theorem Proving, Clifford Bracket Algebra and Clifford Expansions. In T. Qian, T. Hempfling, A. McIntosh, and F. Sommen, editors, *Advances in Analysis and Geometry*, Trends in Mathematics, page 345363. Birkhuser Basel, 2004. ISBN 978-3-0348-9589-7. . URL http://dx.doi.org/10.1007/978-3-0348-7838-8_17.

[19] N. Magaud, J. Narboux, and P. Schreck. Formalizing Desargues' theorem in Coq using ranks in Coq. In *24th Annual ACM Symposium on Applied Computing*, Proceedings of SAC 2009, pages 1110–1115, Honolulu, United States, Mar. 2009. Xiao-Shan Gao, Robert Joan-Arinyo, Dominique Michelucci, ACM. . URL https://hal.inria.fr/inria-00335719.

[20] N. Magaud, J. Narboux, and P. Schreck. A Case Study in Formalizing Projective Geometry in Coq: Desargues Theorem. *Computational Geometry*, 45(8):406–424, 2012. . URL https://hal.inria.fr/inria-00432810.

[21] J. Narboux. A Decision Procedure for Geometry in Coq. In K. Slind, A. Bunker, and G. C. Gopalakrishnan, editors, *Theorem Proving in Higher Order Logics 2004*, volume 3223, pages 225–240, Park City, USA, United States, July 2004. Springer. . URL https://hal.inria.fr/inria-00001035.

[22] J. Narboux. Mechanical Theorem Proving in Tarski's geometry. In F. B. Eugenio Roanes Lozano, editor, *Automated Deduction in Geometry 2006*, volume 4869, pages 139–156, Pontevedra, Spain, Aug. 2006. Francisco Botana, Springer. . URL https://hal.inria.fr/inria-00118812.

[23] M. Pasch. *Vorlesungen ber neuere Geometrie*. Springer, 1976.

[24] L. Pottier. Connecting Gröbner Bases Programs with Coq to do Proofs in Algebra, Geometry and Arithmetics. In G. Sutcliffe, P. Rudnicki, R. Schmidt, B. Konev, and S. Schulz, editors, *Knowledge Exchange: Automated Provers and Proof Assistants*, CEUR Workshop Proceedings, page 418, Doha, Qatar, 2008. URL http://hal.inria.fr/inria-00504727/en/.

[25] W. Schwabhauser, W. Szmielew, and A. Tarski. *Metamathematische Methoden in der Geometrie*. Springer-Verlag, Berlin, 1983.

[26] P. Scott. Mechanising Hilberts foundations of geometry in Isabelle, 2008. URL http://www.inf.ed.ac.uk/publications/thesis/online/IM080566.pdf. Master Thesis.

[27] P. Scott and J. Fleuriot. A Combinator Language for Theorem Discovery. In J. Jeuring, J. Campbell, J. Carette, G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge, editors, *Intelligent Computer Mathematics*, volume 7362 of *Lecture Notes in Computer Science*, page 371385. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-31373-8. . URL http://dx.doi.org/10.1007/978-3-642-31374-5_25.

[28] M. Sozeau and N. Oury. First-Class Type Classes. In O. Mohamed, C. Muoz, and S. Tahar, editors, *Theorem Proving in Higher Order Logics*, volume 5170 of *Lecture Notes in Computer Science*, page 278293. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-71065-3. . URL http://dx.doi.org/10.1007/978-3-540-71067-7_23.

[29] A. Tarski. What is Elementary Geometry? In P. S. L. Henkin and A. Tarski, editors, *The axiomatic Method, with special reference to Geometry and Physics*, pages 16–29, Amsterdam, 1959. North-Holland.

[30] L. Théry. A Machine-Checked Implementation of Buchberger's Algorithm. *Journal of Automated Reasoning*, 26(2):107137, 2001. ISSN 0168-7433. . URL http://dx.doi.org/10.1023/A%3A1026518331905.

[31] D. Wang. *Elimination Methods*. Springer, 2001.

[32] W.-T. Wu. On the Decision Problem and the Mechanization of Theorem Proving in Elementary Geometry. *Scientia Sinica*, 21:157–179, 1978.