Proceedings of ADG 2016

Julien Narboux, Pascal Schreck, Ileana Streinu

Julien Narboux
Pascal Schreck
Ileana Streinu (Eds.)

ADG 2016

Eleventh International Workshop on

Automated Deduction in Geometry

University of Strasbourg, France, June 27-29, 2016

Julien Narboux
Pascal Schreck
Ileana Streinu (Eds.)

Automated Deduction in Geometry

11th International Workshop, ADG 2016
Strasbourg, France, June 27-29, 2016

Preface

This volume contains the 14 papers and 3 invited talks presented at ADG 2016: Eleventh International Workshop on Automated Deduction in Geometry held on June 26-28, 2016 in Strasbourg, France.

ADG is a forum facilitating the exchange of ideas, presentation of new research results and demonstrations of software tools lying at the intersection of geometry and automated deduction. The selected papers, reviewed by an international Program Committee, cover diverse topics ranging from polynomial algebra, invariant and coordinate-free methods, synthetic and logic approaches, techniques for automated geometric reasoning from discrete mathematics, symbolic and numeric methods for geometric computation, geometric algorithms, geometric constraint solving, experimental studies with automated theorem provers, applications to mechanics, origami and geometric modeling.

The previous ten workshops were held in Coimbra 2014, Edinburgh 2012, Munich 2010, Shanghaï 2008, Pontevedra 2006, Gainesville 2004, Linz 2002, Zurich 2000, Beijing 1998, and Toulouse 1996.

The conference was organized by the Computer Graphics and Geometry Group of ICube Laboratory (CNRS / Université de Strasbourg).

We would like to thank the invited speakers and authors for their contributions. We are very grateful to the program committee members and referees for their expertise which ensures the high scientific standard of ADG. We also warmly thank all the local organizers for con-

tributing to the practical organisation so that the meeting can be held smoothly. We thank the IGG team and the ICube laboratory for helping us by sponsoring this conference. Support from EasyChair is also gratefully acknowledged.

June 13, 2016
Strasbourg

Julien Narboux
Pascal Schreck
Ileana Streinu

Program Committee

Michael Beeson	San Jose State University
Francisco Botana	University of Vigo
John Bowers	James Madison University
Xiaoyu Chen	Beihang University
Xiao-Shan Gao	Academia Sinica
Tetsuo Ida	University of Tsukuba
Filip Marić	University of Belgrade
Pascal Mathis	University of Strasbourg
Julien Narboux	University of Strasbourg
Pavel Pech	University of South Bohemia
Pedro Quaresma	University of Coimbra
Tomas Recio	Universidad de Cantabria
Pascal Schreck	University of Strasbourg
Meera Sitharam	University of Florida
Ileana Streinu	Smith College
Dongming Wang	Beihang University and CNRS
Bican Xia	Peking University

Additional Reviewers

Fadoua Ghourabi	Wenping Wang
Menghan Wang	Jeremy Youngquist

Organizers

Pierre Boutry	Pascal Mathis
David Braun	Julien Narboux
Gabriel Braun	Pascal Schreck
Nicolas Magaud	Dan Song

Table of Contents

Geometrisation of Geometry*

Predrag Janičić[1]

Faculty of Mathematics, University of Belgrade, Serbia

Abstract. Coherent logic (CL) is a fragment of first-order logic suitable for automation of proving process and also for formalization of various mathematical theories, including geometry. This paper gives an overview of several developments based on CL with geometry as the domain: automated theorem provers for CL, CL-based formalizations of geometry, CL-based proof representation, links between CL and geometry construction problems, links between CL and geometrical illustrations, etc.

1 Introduction

Automated deduction in geometry has been around for more than sixty years now and it still attracts a lot of attention for several reasons: it requires paradigmatic reasoning that is very difficult to automate, progress in automated reasoning in geometry often leads to ideas that influence other fields, there are practical applications in areas such as robotics, ... and, of course, it is very beautiful. Over the decades, there were several ingenious insights into geometrical reasoning and also links to algebra that led to new methods capable of solving many hard geometry problems [9,10,22]. Still, after all that years and methods, there are no computer programs capable of routinely solving all geometry problems explored in high-school Euclidean geometry. Like in other related domains, there are important issues and challenges concerning scope and efficiency, but in geometry it is also readability of solutions that matters. While algebraic theorem provers for geometry proved that they can efficiently solve a wide scope of geometry problems, they cannot provide readable, understandable solutions in terms of synthetic geometry. The same holds for resolution-based approaches. Some semi-synthetic approaches (such as the area method or the full-angle method) provide proofs that can be concise and intuitive, but not always (often their outputs involve enormously large expressions).

In this paper, coherent logic, denoted by CL, is discussed. A number of theories and theorems can be formulated directly and simply in CL, for instance group theory, ring theory, category theory, the theory of fields, lattice theory, a range of geometries, etc. Several authors independently point to CL or similar fragments of first order logic as suitable for expressing (sometimes – also automating) portions of standard mathematics, for instance, Ganesalingam and Gowers in the context of automated generation of readable proofs [18], Tarski in the context of his geometry [37], Avigad et.al. in the context of a

* Partly supported by the grant 174021 of the Ministry of Science of Serbia.

new diagram-based axiomatic foundations of geometry [1], etc. 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 can serve as a vehicle for producing, at least in some cases and to some extent, human-readable synthetic geometry proofs (in the style of forward reasoning) [4] with "structure of ordinary mathematical arguments better retained" [16]. Moreover, since it allows (limited) existential quantification, it allows building theorem provers with the scope that goes beyond universally quantified fragment, typical for most available proving methods in geometry. CL proofs can also be easily translated into input language of different proof assistants and in a natural language form.

2 Coherent and Geometric Logic

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

$$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}(B_1(\boldsymbol{x}, \boldsymbol{y}) \vee \ldots \vee \ B_m(\boldsymbol{x}, \boldsymbol{y}))$$

where universal closure is assumed, and where $0 \leq n$, $0 \leq m$, $\boldsymbol{x}$ denotes a sequence of variables $x_1, x_2, \ldots, x_k$ $(0 \leq k)$, A_i (for $1 \leq i \leq n$) denotes an atomic formula (involving zero or more variables from $\boldsymbol{x}$), $\boldsymbol{y}$ denotes a sequence of variables $y_1, y_2, \ldots, y_l$ $(0 \leq l)$, and B_j (for $1 \leq j \leq m$) denotes a conjunction of atomic formulae (involving zero or more of the variables from $\boldsymbol{x}$ and $\boldsymbol{y}$). 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, closed under derivability, axiomatised by coherent formulae.[1]

An example of a coherent geometry axiom is (the intuitive meaning is obvious): $point(A) \wedge point(B) \Rightarrow \exists p(line(p) \wedge incident(A, p) \wedge incident(B, p))$.

Every first-order theory has a coherent conservative extension [16,30], i.e., any first-order theory can be translated into coherent logic possibly with additional predicate symbols. This translation process is called "coherentisation" or, sometimes, "geometrisation" [15]. There are several effective ways for translating from FOL to CL (used for proving FOL formulae by coherent provers). Translations typically work by introducing new predicates symbols for subformulae of the input. In translations, one FOL formula may give several CL formulae. Translation of FOL formulae into CL involves elimination of negations: negations

[1] A coherent formula is also known as a "coherent axiom", "special coherent implication", "geometric axiom", "geometric sentence", "basic geometric sequent" [16]. A coherent theory is sometimes called a "geometric theory" [23]. However, much more widely used notion of "geometric formula" allows *infinitary disjuctions* (but only over finitely many variables) [38]. Coherent formulae involve only finitary disjunctions, so coherent logic can be seen as a special case of geometric logic, or as a first-order fragment of geometric logic.

can be kept in place and new predicates symbols for corresponding subformula have to be introduced, or negations can be pushed down to atomic formulae [30]. 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 $\forall \boldsymbol{x}(R(\boldsymbol{x}) \wedge \overline{R}(\boldsymbol{x}) \Rightarrow \bot)$, $\forall \boldsymbol{x}(R(\boldsymbol{x}) \vee \overline{R}(\boldsymbol{x}))$. In order to enable more efficient proving, some advanced translation techniques are used. Elimination of function symbols is also done by introducing additional predicate symbols.[2]

If a coherent formula can be classically proved from a set of coherent formulae, then it can be also intuitionistically proved from that set (this statement is known as the first-order Barr's Theorem [16]). However, translation from FOL to CL is not neccessarily constructive.

The problem of provability in coherent logic is semi-decidable. Coherent logic admits a simple proof system, a sequent-based variants is as follows [35]:

$$\frac{\Gamma, ax, A_1(\boldsymbol{a}), \ldots, A_n(\boldsymbol{a}), \underline{B_1(\boldsymbol{a},\boldsymbol{b}) \vee \ldots \vee B_m(\boldsymbol{a},\boldsymbol{b})} \vdash P}{\Gamma, ax, A_1(\boldsymbol{a}), \ldots, A_n(\boldsymbol{a}) \vdash P}\ \textit{emp}\ (\textit{extended mp})$$

$$\frac{\Gamma, \underline{B_1(\boldsymbol{c})} \vdash P \quad \ldots \quad \Gamma, \underline{B_n(\boldsymbol{c})} \vdash P}{\Gamma, B_1(\boldsymbol{c}) \vee \ldots \vee B_m(\boldsymbol{c}) \vdash P}\ \textit{cs}\ (\textit{case split})$$

$$\frac{}{\Gamma, \underline{B_i(\boldsymbol{a},\boldsymbol{b})} \vdash \exists \boldsymbol{y}(B_1(\boldsymbol{a},\boldsymbol{y}) \vee \ldots \vee\ B_m(\boldsymbol{a},\boldsymbol{y}))}\ \textit{as}\ (\textit{assumption})$$

$$\frac{}{\Gamma, \bot \vdash P}\ \textit{efq}\ (\textit{ex falso quodlibet})$$

In the rules given above, it is assumed: ax is a formula $A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}(B_1(\boldsymbol{x},\boldsymbol{y}) \vee \ldots \vee\ B_m(\boldsymbol{x},\boldsymbol{y}))$; $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ denote vectors of constants (possibly of length zero); in the rule *emp* (*extended modus ponens*), $\boldsymbol{b}$ are fresh constants; $\boldsymbol{x}$ and $\boldsymbol{y}$ denote vectors of variables (possibly of length zero); $A_i(\boldsymbol{x})$ ($B_i(\boldsymbol{x},\boldsymbol{y})$) have no free variables other than from $\boldsymbol{x}$ (and $\boldsymbol{y}$); $A_i(\boldsymbol{a})$ are ground atomic formulae; $B_i(\boldsymbol{a},\boldsymbol{b})$ and $B_i(\boldsymbol{c})$ are ground conjunctions of atomic formulae; $\underline{\Phi}$ denotes the list of conjuncts in Φ if Φ is conjunction, and otherwise Φ itself. In the proving process, the rules are read from bottom to top, i.e., by a rule application one gets the contents (new subgoals) above the line.

For a set of coherent axioms AX and the statement $A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}(B_1(\boldsymbol{x},\boldsymbol{y}) \vee \ldots \vee\ B_m(\boldsymbol{x},\boldsymbol{y}))$ to be proved, within the above proof system one has to derive the following sequent (where $\boldsymbol{a}$ denotes a vector of new simbols of constants): $AX, A_1(\boldsymbol{a}) \wedge \ldots \wedge A_n(\boldsymbol{a}) \vdash \exists \boldsymbol{y}(B_1(\boldsymbol{a},\boldsymbol{y}) \vee \ldots \vee\ B_m(\boldsymbol{a},\boldsymbol{y}))$.

Any coherent logic proof can be represented in the following simple way (*emp* is used zero or more time, *cs* involves at least two other *proof* objects):

$$\textit{proof}\ ::=\ \textit{emp}^*\ \left(\textit{cs}\left(\textit{proof}^{\geq 2}\right)\ |\ \textit{as}\ |\ \textit{efq}\right)$$

[2] Elimination of function symbols is sometimes called *anti Skolemization* [13].

3 Automated Theorem Proving for CL

There are several semi-decision proving procedures for coherent logic and there are several implemented automated provers. To our knowledge, the first CL automated theorem prover was developed in Prolog by Janičić and Kordić [21] and was used for one fixed axiomatization of Euclidean geometry – an axiom system closely related to Borsuk's one [7]. This prover, based on forward chaining and iterative deepening, was later reimplemented in C++ to give a more efficient and generic theorem prover ArgoCLP [36] that produces both natural language proofs (formatted in LaTeX or in HTML) and object level proofs in the Isabelle form [29]. Bezem developed in Prolog a CL prover based on depth-first search that can generate proof objects in Coq [4]. This prover was used for proving Hessenberg's theorem of projective plane geometry (that states that Pappus axiom implies Desargues axiom), by proving a number of lemmas [5]. Fisher developed a prover GeologUI with graphical interface [17]. Berghofer developed (using shallow embedding) an internal prover for CL in ML to be used within the system Isabelle. None of these provers uses backjumps or lemma learning. De Nivelle implemented a theorem prover Geo for logic close to coherent logic, that uses a mechanism for learning lemmas of somewhat restricted form [13]. All of these systems perform only ground reasoning. A prover Calypso [28] supports lemma learning and introduces non-ground reasoning in automated proving for CL. For lemma learning, the latter two systems use ideas from CDCL SAT solving [6].

The prover ArgoCLP can be used for proving geometry statements in several contexts (more details are given in the following sections). In generated proofs, the prover brings back the eliminated negation symbols. The proofs that ArgoCLP generates are automatically simplified by elimination of redundant steps and using the *reductio ad absurdum* form [24]. An example of a proof of a theorem from Hilbert's geometry [19] is given below (a corresponding proof for the system Isabelle can also be generated).

Theorem: Assuming that $\alpha \neq \beta$, the line p is incident to the plane α, the line p is incident to the plane β, the point A is incident to the plane α, and the point A is incident to the plane β, show that the point A is incident to the line p.

Proof:

Let us prove that the point A is incident to the line p by reductio ad absurdum.

1. Assume that the point A is not incident to the line p.

2. There exist a point B and a point C such that the point B is incident to the line p, $B \neq C$ and the point C is incident to the line p (by axiom ax_I3a).

3. From the facts that the line p is incident to the plane α, and the point B is incident to the line p, it holds that the point B is incident to the plane α (by axiom ax_D11).

4. From the facts that the line p is incident to the plane β, and the point B is incident to the line p, it holds that the point B is incident to the plane β (by axiom ax_D11).

5. From the facts that $B \neq C$, the point B is incident to the line p, the point C is incident to the line p, and the point A is not incident to the line p, it holds that the points B, C and A are not collinear (by axiom ax_D1a).

6. From the facts that the line p is incident to the plane α, and the point C is incident to the line p, it holds that the point C is incident to the plane α (by axiom ax_D11).

7. From the facts that the line p is incident to the plane β, and the point C is incident to the line p, it holds that the point C is incident to the plane β (by axiom ax_D11).

8. From the facts that the points B, C and A are not collinear, it holds that the points A, B and C are not collinear (by axiom ax_ncol_231).

9. From the facts that the points A, B and C are not collinear, the point A is incident to the plane α, the point B is incident to the plane α, the point C is incident to the plane α, the point A is incident to the plane β, the point B is incident to the plane β, and the point C is incident to the plane β, it holds that $\alpha = \beta$ (by axiom ax_I5).

10. From the facts that $\alpha = \beta$, and $\alpha \neq \beta$ we get a contradiction.

Contradiction.

Therefore, it holds that the point A is incident to the line p.

This proves the conjecture.

Theorem proved in 10 steps and in 4.56 s.

4 CL Vernacular

Automated provers for coherent logic such as ArgoCLP export proofs in some custom format(s). However, it would be beneficial if there is an output format supported by various CL provers that can be further used for generating proofs in different target languages. Following this motivation, a new proof representation and a corresponding format for coherent logic were developed [35].

De Bruijn used a syntagm *mathematical vernacular*[3] in 1980's within his formalism proposed for trying to "put a substantial part of the mathematical vernacular into the formal system" [12]. Several authors later modified or extended de Bruijn's framework. Wiedijk follows de Bruijn's motivation [41], but he also notices: "It turns out that in a significant number of systems (proof assistants) one encounters languages that look almost the same. Apparently there is a canonical style of presenting mathematics that people discover independently: something like a *natural* mathematical vernacular. Because this language apparently is something that people arrive at independently, we might call it *the* mathematical vernacular." The language discussed by Wiedijk is actually closely related to a proof language of coherent logic. CL can naturally express a number of mathematical theories, including different geometries. A proof representation

[3] *Vernacular* is the everyday, ordinary language (in contrast to the official, literary language) of the people of some country or region.

called "coherent logic vernacular" [35] can link different proof formats and presentations. CL vernacular is simple, yet expressive proof representation with only a few inference rules, shown in Section 2, supported.

The proposed proof representation is accompanied by a corresponding XML format, specified by a DTD `Vernacular.dtd`. As an illustration, here are some fragments of this DTD file:

```
...
<!--******** Theory **************-->
<!ELEMENT theory (theory_name, signature, axiom*) >
<!ELEMENT theory_name (#PCDATA)>
<!ELEMENT signature (type*, relation_symbol*, constant*) >
<!ELEMENT relation_symbol (type*)>
<!ATTLIST relation_symbol name CDATA #REQUIRED>
<!ELEMENT type (#PCDATA)>
<!ELEMENT axiom (cl_formula)>
<!ATTLIST axiom name CDATA #REQUIRED>
...
```

The above fragment describes the notion of theory. A file describing a theory could be shared among several files with theorems and proofs. The following fragment describes the notion of a theorem and a proof:

```
...
<!--******** Theorem **************-->
<!ELEMENT theorem (theorem_name, cl_formula, proof+)>
<!ELEMENT theorem_name (#PCDATA)>
<!ELEMENT conjecture (name, cl_formula)>

<!--******** Proof **************-->
<!ELEMENT proof (proof_step*, proof_closing, proof_name?)>
<!ELEMENT proof_name EMPTY>
<!ATTLIST proof_name name CDATA #REQUIRED>

<!--******** Proof steps **************-->
<!ELEMENT proof_step (indentation,extended_modus_ponens)>
<!ELEMENT proof_closing (indentation, (case_split|efq|from),
  (goal_reached_contradiction|goal_reached_thesis))>
...
```

The XML format is supported by a suite of XSL transformations for generating formal proofs for Isabelle/Isar and Coq, as well as proofs expressed in a natural language form (formatted in LaTeX or in HTML). The XML documents themselves can be read by a human, but much better alternative is using translation to human readable proofs in natural language. Proofs (for Coq and Isabelle/Isar) that are produced from XML documents are also fairly readable. The developed XSLT style-sheets are rather simple and short — each is only around 500 lines long. This shows that transformations for other target languages (other theorem provers, like Mizar and HOL light, LaTeX with other

natural languages, MathML, OMDoc or TPTP) can easily be constructed, thus enabling wide access to a single source of mathematical contents.

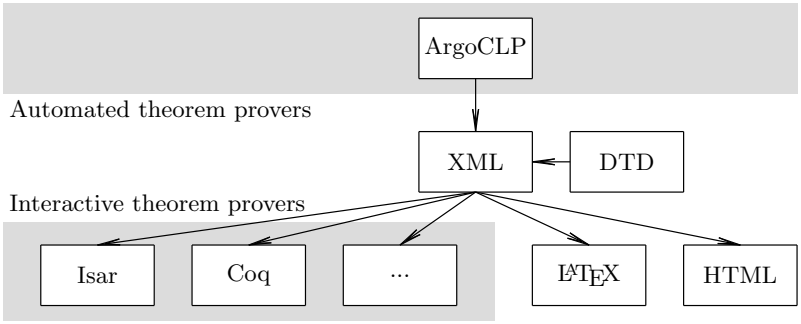


Fig. 1. Architecture of the presented framework

The automated theorem prover for coherent logic ArgoCLP can export proofs in the form of the XML files that conforms to this DTD (while reads an input theory and the conjecture given in the TPTP[4] form). The overall architecture of the framework is shown in Figure 1.

As an illustration, below are given proofs of one theorem (4.19) from Tarski's book on geometry [34], in natural language and in the Coq form (generated from XML representation produced by ArgoCLP) ($(A, B) \cong (C, D)$ is an infix notation for $cong(A, B, C, D)$ and it denotes that the pairs of points (A, B) and (C, D) are congruent, $bet(A, B, C)$ denotes that the point B is between the points A and C, $col(A, B, C)$ denotes that the points A, B and C are collinear).

Theorem: Assuming that $bet(A, B, C)$ and $AB \cong AD$ and $CB \cong CD$ it holds that $B = D$.

Proof:

1. It holds that $bet(B, A, A)$ (using th_3_1).
2. From the fact(s) $bet(A, B, C)$ it holds that $col(C, A, B)$ (using $ax\_4\_10\_3$).
3. From the fact(s) $AB \cong AD$ it holds that $AD \cong AB$ (using th_2_2).
4. It holds that $A = B$ or $A \neq B$.
   5. Assume that: $A = B$.
      6. From the fact(s) $AD \cong AB$ and $A = B$ it holds that $AD \cong AA$.
      7. From the fact(s) $AD \cong AA$ it holds that $A = D$ (using ax_3).
      8. From the fact(s) $A = B$ and $A = D$ it holds that $B = D$.
      9. The conclusion follows from the fact(s) $B = D$.
   10. Assume that: $A \neq B$.
      11. It holds that $A = C$ or $A \neq C$.
         12. Assume that: $A = C$.

[4] `http://www.cs.miami.edu/~tptp/`

13. From the fact(s) $bet(A, B, C)$ and $A = C$ it holds that $bet(A, B, A)$.

14. From the fact(s) $bet(A, B, A)$ and $bet(B, A, A)$ it holds that $A = B$ (using th_3_4).

15. From the fact(s) $A \neq B$ and $A = B$ we get contradiction.

16. Assume that: $A \neq C$.

17. From the fact(s) $A \neq C$ it holds that $C \neq A$.

18. From the fact(s) $C \neq A$ and $col(C, A, B)$ and $CB \cong CD$ and $AB \cong AD$ it holds that $B = D$ (using th_4_18).

19. The conclusion follows from the fact(s) $B = D$.

20. The conclusion follows in all cases.

21. The conclusion follows in all cases.

QED

```
Theorem th_4_19 : ∀ (A:point) (B:point) (C:point) (D:point), (bet A B C ∧ cong
A B A D ∧ cong C B C D) → B = D.
Proof.
 intros.
 assert (bet B A A) by applying (th_3_1 B A ) .
 assert (col C A B) by applying (ax_4_10_3 A B C ) .
 assert (cong A D A B) by applying (th_2_2 A B A D ) .
 assert (A = B ∨ A ≠ B) by applying (ax_g1 A B ) .
 by cases on (A = B ∨ A ≠ B).
 - {
    assert (cong A D A A) by (substitution).
    assert (A = D) by applying (ax_3 A D A ) .
    assert (B = D) by (substitution).
    conclude.
    }
 - {
    assert (A = C ∨ A ≠ C) by applying (ax_g1 A C ) .
    by cases on (A = C ∨ A ≠ C).
    - {
       assert (bet A B A) by (substitution).
       assert (A = B) by applying (th_3_4 A B A ) .
       assert (False) by (substitution).
       contradict.
       }
    - {
       assert (C ≠ A) by (substitution).
       assert (B = D) by applying (th_4_18 C A B D ) .
       conclude.
       }
    }
Qed.
```

5 CL-based Formalizations of Geometry

Large portions of geometry can be expressed in coherent logic, or can be relatively easily transformed into coherent logic. This is supported by a study [14] of the book on foundations of geometry *Metamathematische Methoden in der Geometrie*, by Wolfram Schwabhäuser, Wanda Szmielew, and Alfred Tarski [34]. This book has been a subject of several automation and formalization projects, using automated theorem proving [2,3,31] or interactive theorem proving [8,27]. Geometry in this book is expressed in terms of first-order logic with equality, without sorts (the only primitive objects are *points*) and with two primitive predicate symbols written (in prefix form) as *cong* (for congruence) and *bet* (for betweeness). There are only eleven axioms (it is assumed that all axioms are universally closed):

Axiom A1: $cong(A, B, B, A)$
Axiom A2: $cong(A, B, P, Q) \wedge cong(A, B, R, S) \Rightarrow cong(P, Q, R, S)$
Axiom A3: $cong(A, B, C, C) \Rightarrow A = B$
Axiom A4: $\exists X\ (bet(Q, A, X) \wedge cong(A, X, B, C))$
Axiom A5: $A \neq B \wedge bet(A, B, C) \wedge bet(A', B', C') \wedge cong(A, B, A', B') \wedge cong(B, C, B', C') \wedge cong(A, D, A', D') \wedge cong(B, D, B', D') \Rightarrow cong(C, D, C', D')$
Axiom A6: $bet(A, B, A) \Rightarrow A = B$
Axiom A7: $bet(A, P, C) \wedge bet(B, Q, C) \Rightarrow \exists X\ (bet(P, X, B) \wedge bet(Q, X, A))$
Axiom A8: $\exists A\ \exists B\ \exists C\ (\neg bet(A, B, C) \wedge \neg bet(B, C, A) \wedge \neg bet(C, A, B))$
Axiom A9: $P \neq Q \wedge cong(A, P, A, Q) \wedge cong(B, P, B, Q) \wedge cong(C, P, C, Q) \Rightarrow (bet(A, B, C) \vee bet(B, C, A) \vee bet(C, A, B))$
Axiom A10: $bet(A, D, T) \wedge bet(B, D, C) \wedge A \neq D \Rightarrow \exists X\ \exists Y\ (bet(A, B, X) \wedge bet(A, C, Y) \wedge bet(X, T, Y))$
Axiom A11 (continuity): $\forall \Phi\ \forall \Psi\ \exists A\ \forall X\ \forall Y\ ((X \in \Phi \wedge Y \in \Psi \Rightarrow bet(A, X, Y)) \Rightarrow \exists B\ \forall X\ \forall Y\ (X \in \Phi \wedge Y \in \Psi \Rightarrow bet(X, B, Y))$

One goal of the study was analysis of how many theorems from such book can be proved by coherent logic prover ArgoCLP supported by resolution theorem provers [14]. The process of coherentisation of the theorems was straightforward (and included eliminating sets that were used in the book only as a syntactic sugar, for the sake of clarity and conciseness). From the original 179 theorems, the process gave 238 coherent formulae (while 5 schematic theorems involving n-tuples were considered only for $n = 2$).

Given all the axioms and the theorems from the book in coherent form and written in the TPTP format, the proving process went as follows.

- Given a theorem to be proved, all axioms and theorems that precede it (in the book) are passed to resolution provers (Vampire [32], E [33], or SPASS [39]).
- If one or more resolution provers proves the conjecture, the smallest list of used axioms/theorems (returned by the resolution prover) is used for proving the conjecture again, in the same manner. This process is repeated until the list of used axioms/theorems remains unchanged between two consecutive iterations.

– With the obtained list of axioms, ArgoCLP prover is invoked, and (if successful) the proof is exported in the CL vernacular XML format.

ArgoCLP supported by resolution provers proved (in the above way) 37% of the theorems from chapters 1 to 12 of the book completely automatically. This performance was reached in the scenario that mimics humans doing mathematics: even if one cannot prove some theorem, he/she proceeds with further theorems and uses even those theorems that were unproven. In a more strict formalization scenario, in which only theorems that were already automatically proved could be used, the framework proved 17% of the theorems.

One of the outputs of the proving framework within the above study was a digital version of the book, with all axioms, definitions, theorems, and generated proofs filled-in, all in the natural language form.

6 Coherent Logic and Construction Problems

Almost all automated theorem provers for geometry focus on universally quantified statements. However, statements of the form $\forall\exists$ naturally arise in geometry. One source of such statements are construction problems. Popular geometry theorem provers (like those based on Wu's method) can be used for showing correctness of a solution for a construction problem, however, they cannot provide a full answer: they can (at least for some constructions) prove statements such as *if some points meet some conditions, then it holds that....* However, it is not answered whether such points exist, and if yes – under what conditions.

For a construction problem, roughly said, the task is to *prove constructively* a theorem of the following form (where $\boldsymbol{x}$ and $\boldsymbol{y}$ denote vectors of objects – points, lines, rays, etc):

$$\forall \boldsymbol{x} \exists \boldsymbol{y}\ \Psi(\boldsymbol{x}; \boldsymbol{y})$$

The above subsumes two claims: that the problem is solvable (say, by ruler and compass) and that a particular construction (that witnesses $\exists \boldsymbol{y}\ \Psi(\boldsymbol{x}; \boldsymbol{y})$) is correct. There could be given some constraints imposed on the given objects $\boldsymbol{x}$ — some construction problems do not have solutions and some problems have solutions only under some additional conditions, not known in advance. So, one typically has to discover[5] $\Phi(\boldsymbol{x})$ (for the given $\Psi(\boldsymbol{x}; \boldsymbol{y})$) and to prove:

$$\forall \boldsymbol{x} (\Phi(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}\ \Psi(\boldsymbol{x}; \boldsymbol{y}))$$

The above claims that solution exists under some conditions. But one may claim even more:

$$\forall \boldsymbol{x} (\Phi(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}\ \Psi(\boldsymbol{x}; \boldsymbol{y})\ \wedge\ \neg\Phi(\boldsymbol{x}) \Rightarrow \neg\exists \boldsymbol{y}\ \Psi(\boldsymbol{x}; \boldsymbol{y}))$$

[5] In solving specific classes of construction problems, some goal conditions may be assumed. For instance, in solving triangle construction problems, an implicit goal condition is that the constructed points A, B, and C are not collinear.

The above gives a complete characterization of solvability: it states that solution exists under some conditions $\Phi(\boldsymbol{x})$ and solution does not exist otherwise.

Let us, for example, consider the contruction problem 4 from Wernick's list [40]: *Given points A, B, and G, construct a triangle ABC, such that G is the centroid of ABC.* A careful analysis leads to the theorem that gives a clear characterization of solvability:

$$\forall A, B, G\ (\neg collinear(A, B, G) \Leftrightarrow$$
$$\exists C.(\neg collinear(A, B, C) \wedge centroid(G, A, B, C)))$$

which can be trivially transformed into two coherent formulae.

The system ArgoTriCS [26] is able to automatically find constructions for almost all solvable Wernick's and Connelly's [11] problems. It uses algebraic provers for proving correctness of solutions (under assumption that the constructed object do exist) and the prover ArgoCLP for showing statements that give characterizations of solvability [25]. The prover ArgoCLP is not powerful enough to prove needed theorems and it uses facts used by ArgoTriCS in the generated solution for the construction problem.

7 Coherent Logic and Geometric Illustrations

In geometry and in the whole of mathematics, illustrations are often very valuable, but almost always just an informal content, provided to support intuition and understandability of proofs. Links between proofs and associated illustration are typically very loose: proofs do not rely on illustrations, illustrations are not derived from proofs. However, proofs in coherent logic can be used, in some cases, for automated generation of illustrations.[6] The idea is not to instruct a CL prover to generate illustrations, but to use proofs themselves (represented, say, as discussed in Section 4) and generate illustrations from them directly.

In each substantial step of a CL proof, one formula (axiom or a lemma) of the following form is used:

$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}(B_1(\boldsymbol{x}, \boldsymbol{y}) \vee \ldots \vee\ B_m(\boldsymbol{x}, \boldsymbol{y}))$.

Each axiom (or potentially used lemma) with $m > 0$ (i.e., each axiom that introduce news objects) need to have an associated illustration rule. For instantiated $\boldsymbol{x}$ interpreted in some model (e.g., Cartesian plane), there should be a rule for determining $\boldsymbol{y}$. In some cases, such $\boldsymbol{y}$ are determined uniquely, and in some cases there are degrees of freedom. These rules should be formulated in a language that can serve as a specification language for illustrations. One such language is geometry language GCLC [20]. For example, an axiom *for any two points A and B, there is a point C such that* $bet(A, B, C)$ can be modelled in GCLC in the following way:

```
random r
expression r' {1+r}
towards C A B r'
```

[6] This idea has not been implemented yet.

where `random` choses a non-negative pseudorandom real number `r`, the second line gives a number `r'` such that `r'` ≥ 1, and the third line determines `C` (a concrete point in Cartesian plane) given two (concrete points) `A` and `B`. When the case split rule is applied, only one case is illustrated (so, one proof has only one illustration). This process is straightforward and yields illustration for each proof (with all axioms properly processed). However, there is still one initial challenge. If the conjecture being proved has the form: $\forall \boldsymbol{x}(\Phi(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}\, \Psi(\boldsymbol{x}; \boldsymbol{y}))$ in order to be illustrated, one must have initial objects $\boldsymbol{x}$ meeting conditions $\Phi(\boldsymbol{x})$. So, the first step is to prove that $\Phi(\boldsymbol{x})$ is consistent (if it is not, the statement is trivially valid), i.e., to prove $\exists \boldsymbol{x}\, \Phi(\boldsymbol{x})$. A constructive proof of this conjecture will give one model for $\Phi(\boldsymbol{x})$ and will serve as a basis for an illustration for the main proof. Note that this step may not be easy and it actually involves solving a geometry construction problem.

8 Conclusions

Coherent logic has several applications in automated deduction in geometry: in producing human-readable proofs, in producing machine verifiable proofs, in proving statemens of the form $\forall\exists$, in producing illustrations automatically from proofs, etc. However, there are still potentials to be used. For instance, theorem provers for coherent logic have been so far used for fully automated proving of only low-level conjectures (with proofs close to the axiomatic level). It would be interesting to construct suitable sets of geometry lemmas that can be used in proving higher-level conjectures.

Acknowledgements The author is grateful to Marc Bezem, Vesna Marinković, Mladen Nikolić, and Sana Stojanović Đurđević for their feedback on earlier versions of this paper.

References

1. Jeremy Avigad, Edward Dean, and John Mumma. A formal system for Euclid's Elements. *The Review of Symbolic Logic*, 2009.
2. Michael Beeson. Proof and computation in geometry. In *Automated Deduction in Geometry - 9th International Workshop*, volume 7993 of *Lecture Notes in Computer Science*, pages 1–30. Springer, 2013.
3. Michael Beeson and Larry Wos. OTTER proofs in tarskian geometry. In *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*, volume 8562 of *Lecture Notes in Computer Science*, pages 495–510. Springer, 2014.
4. Marc Bezem and Thierry Coquand. Automating coherent logic. In Geoff Sutcliffe and Andrei Voronkov, editors, *12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning — LPAR 2005*, volume 3835 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.

5. Marc Bezem and Dimitri Hendriks. On the Mechanization of the Proof of Hessenberg's Theorem in Coherent Logic. *Journal of Automated Reasoning*, 40(1), 2008.
6. Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.
7. Karol Borsuk and Wanda Szmielew. *Foundations of Geometry*. Norht-Holland Publishing Company, Amsterdam, 1960.
8. Gabriel Braun and Julien Narboux. From Tarski to Hilbert. In Tetsuo Ida and Jacques Fleuriot, editors, *Automated Deduction in Geometry 2012*, Edinburgh, United Kingdom, September 2012.
9. Shang-Ching Chou. *Mechanical Geometry Theorem Proving*. D.Reidel Publishing Company, Dordrecht, 1988.
10. Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. *Machine Proofs in Geometry*. World Scientific, Singapore, 1994.
11. Harold Connelly. An Extension of Triangle Constructions from Located Points. *Forum Geometricorum*, 9:109–112, 2009.
12. Nicolaas Govert de Bruijn. The mathematical vernacular, a language for mathematics with typed sets. In Dybjer et al., editor, *Proceedings of the Workshop on Programming Languages*, 1987.
13. Hans de Nivelle and Jia Meng. Geometric resolution: A proof procedure based on finite model search. In *Automated Reasoning, Third International Joint Conference, IJCAR*, volume 4130 of *Lecture Notes in Computer Science*, pages 303–317. Springer, 2006.
14. Sana Stojanović Djurdjević, Julien Narboux, and Predrag Janičić. Automated generation of machine verifiable and readable proofs: A case study of tarski's geometry. *Annals of Mathematics and Artificial Intelligence*, 74(3-4):249–269, 2015.
15. Roy Dyckhoff. Coherentisation of first-order logic. In *Automated Reasoning with Analytic Tableaux and Related Methods - 24th International Conference, TABLEAUX 2015*, volume 9323 of *Lecture Notes in Computer Science*, pages 3–5. Springer, 2015.
16. Roy Dyckhoff and Sara Negri. Geometrization of first-order logic. *The Bulletin of Symbolic Logic*, 21:123–163, 2015.
17. John Fisher and Marc Bezem. Skolem machines and geometric logic. In Cliff B. Jones, Zhiming Liu, and Jim Woodcock, editors, *4th International Colloquium on Theoretical Aspects of Computing — ICTAC 2007*, volume 4711 of *Lecture Notes in Computer Science*. Springer-Verlag, 2007.
18. Mohan Ganesalingam and William Timothy Gowers. A fully automatic problem solver with human-style output. *CoRR*, abs/1309.4501, 2013.
19. David Hilbert. *Grundlagen der Geometrie*. Leipzig, 1899.
20. Predrag Janičić. Geometry Constructions Language. *Journal of Automated Reasoning*, 44(1-2):3–24, 2010.
21. Predrag Janičić and Stevan Kordić. EUCLID — the geometry theorem prover. *FILOMAT*, 9(3):723–732, 1995.
22. Predrag Janičić, Julien Narboux, and Pedro Quaresma. The area method: a recapitulation. *Journal of Automated Reasoning*, 48(4):489–532, 2012.
23. Saunders MacLane and Ieke Moerdijk. *Sheaves in geometry and logic: a first introduction to topos theor*. Springer-Verlag, 1992.
24. Vesna Marinkovic. Proof simplification in the framework of coherent logic. *Computing and Informatics*, 34(2):337–366, 2015.

25. Vesna Marinkovic, Predrag Janicic, and Pascal Schreck. Computer theorem proving for verifiable solving of geometric construction problems. In *Automated Deduction in Geometry - 10th International Workshop, Revised Selected Papers*, volume 9201, pages 72–93. Springer, 2015.
26. Vesna Marinković and Predrag Janičić. Towards Understanding Triangle Construction Problems. In J. Jeuring et al., editor, *Intelligent Computer Mathematics - CICM 2012*, volume 7362 of *Lecture Notes in Computer Science*. Springer, 2012.
27. Julien Narboux. Mechanical theorem proving in Tarski's geometry. In *Proceedings of Automatic Deduction in Geometry 06*, volume 4869 of *Lecture Notes in Artificial Intelligence*, pages 139–156. Springer-Verlag, 2007.
28. Mladen Nikolić and Predrag Janičić. CDCL-based abstract state transition system for coherent logic. In J. et.al. Jeuring, editor, *Intelligent Computer Mathematics - CICM 2012*, volume 7362 of *Lecture Notes in Computer Science*. Springer, 2012.
29. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle HOL: a Proof Assistant for Higher-Order Logic*. Springer, 2005. URL: `http://www.cl.cam.ac.uk/research/hvg/Isabelle/dist/Isabelle/doc`.
30. Andrew Polonsky. *Proofs, Types and Lambda Calculus*. PhD thesis, University of Bergen, 2011.
31. Art Quaife. Automated development of tarski's geometry. *Journal of Automated Reasoning*, 5(1):97–118, 1989.
32. Alexandre Riazanov and Andrei Voronkov. The design and implementation of vampire. *AI Communications*, 15(2-3):91–110, 2002.
33. Stephan Schulz. E - a brainiac theorem prover. *AI Communications*, 15(2-3):111–126, 2002.
34. Wolfram Schwabhuser, Wanda Szmielew, and Alfred Tarski. *Metamathematische Methoden in der Geometrie*. Springer-Verlag, Berlin, 1983.
35. Sana Stojanovic, Julien Narboux, Marc Bezem, and Predrag Janicic. A vernacular for coherent logic. In *Intelligent Computer Mathematics*, volume 8543 of *Lecture Notes in Computer Science*, pages 388–403. Springer, 2014.
36. Sana Stojanović, Vesna Pavlović, and Predrag Janičić. A coherent logic based geometry theorem prover capable of producing formal and readable proofs. In Pascal Schreck, Julien Narboux, and Jürgen Richter-Gebert, editors, *Automated Deduction in Geometry*, volume 6877 of *Lecture Notes in Computer Science*. Springer, 2011.
37. Alfred Tarski and Steven Givant. Tarski's system of geometry. *The Bulletin of Symbolic Logic*, 5(2), June 1999.
38. Steven Vickers. Geometric logic in computer science. In *Theory and Formal Methods*, Workshops in Computing, pages 37–54. Springer, 1993.
39. Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. Spass version 3.5. In *Automated Deduction - CADE-22 Proceedings*, volume 5663 of *Lecture Notes in Computer Science*, pages 140–145. Springer, 2009.
40. William Wernick. Triangle constructions vith three located points. *Mathematics Magazine*, 55(4):227–230, 1982.
41. Freek Wiedijk. Mathematical Vernacular. Unpublished note. `http://www.cs.ru.nl/~freek/notes/mv.pdf`, 2000.

Solving With Or Without Equations

Dominique Michelucci

LE2I UMR6306, CNRS, Arts et Métiers,
Bourgogne Franche-Comté University, Dijon, France

1 Equations versus algorithms, back and forth

The pentahedron problem (§2) shows the proximity between Geometric Theorem Proving (GTP) and Geometric Constraint Solving (GCS). However, the two fields separate, due to specificities of GCS (§3), which prefers algorithms to equations. Yet GCS still benefits from symbolic tools, like DAG (§4), and dual numbers (§5). Finally, §6 conjectures that algorithms can be converted to systems of equations.

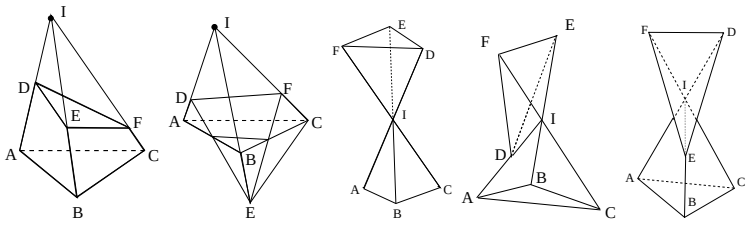


Fig. 1. Pentahedra. 6 vertices, 5 faces, 9 edges. The lengths of all edges is known.

2 The Pentahedron problem

The pentahedron problem Fig.1 [1] is to find compatible coordinates for its 6 vertices $ABCDEF$. The planarity of the 3 quadrilateral faces provides 3 constraints, and the specified lengths of the 9 edges provide 9 other constraints. This problem is well-constrained, up to 3D location and orientation. It is easy to solve for the triangle ABC and to pin it in the Oxy plane. Then it remains a polynomial system in 9 equations and 9 unknowns, the 3D coordinates of D, E, F. This system can be solved, slowly, with an interval solver. A much better formulation remarks that lines AD, BE, CF must be concurrent. Either they meet at a common point I, or they are parallel. In the first case, the 3 lengths of ID, IE, IF can be used as 3 unknowns x_{ID}, x_{IE}, x_{IF} of a smaller, and intrinsic

(coordinate-free) system of 3 equations and 3 unknowns: the law of cosines gives the 3 equations, one per quadrilateral face; *e.g.,* let $\alpha = \widehat{DIE} = \widehat{AIB}$; then

$$\cos\alpha = \frac{x_{ID}^2 + x_{IE}^2 - l_{DE}^2}{2x_{ID}x_{IE}} = \frac{(x_{ID} + l_{AD})^2 + (x_{IE} + l_{BE})^2 - l_{AB}^2}{2(x_{ID} + l_{AD})(x_{IE} + l_{BE})}$$

gives the equation in x_{ID}, x_{IE} for the quadrilateral face $ABED$. This system is solved 42 times faster than the previous one. In the second case, the point I is at infinity and a simple geometric construction shows that there are always pentahedra with parallel edges AD, BE, CF, except when some triangular or tetrahedric inequality is violated. Finally, there are 6 spurious roots, where the pentahedron is flat, so edges AD, BE, CF need not be concurrent.

This problem illustrates many common issues to GCS and GTP: what is the dimension of the manifold solution? Are there points at infinity? What is the best way to pose equations? Are there any degenerate solutions, and what is the topological dimension of the degenerate manifold? Indeed, in GTP, non degeneracy conditions (the triangle must not be flat, vertices must be distinct, etc) have to be specified in order to prove theorems. This example also shows that GCS and GTP are close while all constraints are incidence, or distance, or angle constraints between flats: points, lines, planes. But the latter are not sufficient in CADCAM.

3 Specificities of GCS for CADCAM

The first specificity in GCS is the inaccuracy issue. The nullity of a number, the equality of two numbers are no more decidable. The computation of the rank of a set of vectors, or of a Jacobian, is no more guaranteed. The distinction between $x > 0$ and $x \geq 0$ becomes irrelevant. The equivalence $x \neq 0 \Leftrightarrow \exists y \,|\, xy - 1 = 0$ used in Gröbner bases becomes irrelevant as well.

Another specificity is the need for optimization and algorithms.

For example, there are many orthogonal projections of a point on a non linear curve or surface but for distance constraints, only one is relevant. First order conditions, like KKT (Karush-Kuhn-Tucker), are necessary but not sufficient to fully characterize solutions. Solving KKT equations provides a superset of the roots, and spurious ones (saddle points, local optima) must be cancelled with some algorithm.

When computing the orthogonal projection of a point p to a composite object (*e.g.,* the union of a line and a conic), often used in CADCAM, the relevant system of equations depends on the location of p. Again, some optimization problem occurs. For the orthogonal projection on a part of an object, like a segment, optimization can be avoided, but an algorithm and some if-then-else are more convenient than equations.

CADCAM systematically uses piecewise polynomials (box splines, bsplines, etc). They are not polynomials and standard tools of Computer Algebra (Gröbner bases, Wu-Ritt method, resultants, GCD, fundamental theorem of algebra, Sturm's theorem, etc) no more apply. Idem for NURBS and piecewise rational functions.

Finally, Computer Graphics and CADCAM use algorithmic shapes, called features or parametric objects, like staircases, gears and sprockets, etc. The number of steps in a staircase is an integer (thus diophantine equations occur) and depend on parameter values of length and height: the number of unknowns and equations depend on parameter values. In passing, there is some similarity with Steiner's porism, or Poncelet's porism, in GTP.

Worse, subdivision curves and subdivision surface have invaded Computer Graphics: designers interactively define a coarse mesh, and a procedure rounds vertices and edges. Most of the time, there is no equation for the limit surface.

Sometimes, equations are available but too huge to be symbolically expanded, *e.g.,* $\det(M(X)) = 0$. Of course, a numerical algorithm can still be used to compute $\det(M(V))$ for a given numerical vector V. Another example is given by intersection curves between rational surfaces: they are not rational but all geometric modelers approximate them with rational curves.

4 DAGs

Gouaty et al [2] solve such geometric constraints for CADCAM: equations are replaced with algorithms. Constraints are represented with DAGs (Directed Acyclic Graph). DAG is a popular data structure in Dynamic Geometry softwares (where they are called Straight Line Programs) and in Computer Algebra. In CADCAM, DAGs involve spline or NURBS functions, algorithms (for rounding, for orthogonal projection), subdivision surfaces and other algorithmic shapes. They are no more convertible into polynomials, and it is no more possible to compute the DAG of the derivative of a given DAG. But these DAG keep some interesting features: they still can be evaluated for given values of parameters, thus it is still possible to solve; DAG can be interactively specified and modified by users or designers who are not computer scientists, thus users can still pose their problems; probabilistic tests for nullity or equality (up to some tolerance) are still possible; and finally, exact computations (up to floating point precision) of derivatives are still possible, after all, with dual numbers. This is interesting because derivatives computed with finite differences are inaccurate, which hampers the convergence of numeric solvers close to the solution.

5 Dual numbers

The idea is to attach an infinitesimal number ϵ_i to each unknown x_i, with the rule $\epsilon_i^2 = \epsilon_i \epsilon_j = 0$. The addition is straightforward. The product, for one ϵ, is given by:

$$\begin{array}{ccccc} (a + b\,\epsilon) & \times & (a' + b'\,\epsilon) & = & aa' + (ab' + ba')\,\epsilon \\ \downarrow & & \downarrow & & \downarrow \\ \begin{pmatrix} a & 0 \\ b & a \end{pmatrix} & \times & \begin{pmatrix} a' & 0 \\ b' & a' \end{pmatrix} & = & \begin{pmatrix} aa' & 0 \\ ba' + ab' & aa' \end{pmatrix} \end{array} \tag{1}$$

and it is generalizable to many ϵ_i. The bijection between dual numbers and matrices is an isomorphism: the matrice of the opposite (inverse) of a dual number is the opposite (inverse) of the matrice of the dual number. Other rules are:

$$\frac{1}{a+b\,\epsilon} = \frac{1}{a} - \frac{b}{a^2}\,\epsilon \text{ when } a \neq 0 \tag{2}$$

thus $b\epsilon$ has no inverse (the associated matrice is not invertible). This rule is a special case of:

$$(a+b\epsilon)^k = a^k + ka^{k-1}b\,\epsilon \tag{3}$$

If P is a polynomial, then $P(x_v + \epsilon)$ where x_v is a floating-point number, gives $P(x_v)$ and the derivative $P'(x_v)$:

$$\begin{aligned} P(x_v + \epsilon) &= a(x_v+\epsilon)^3 + b(x_v+\epsilon)^2 + c(x_v+\epsilon) + d \\ &= a(x_v^3 + 3x_v^2\,\epsilon) + b(x_v^2 + 2x_v\,\epsilon) + c(x_v + \epsilon) + d \\ &= (ax_v^3 + bx_v^2 + cx_v + d) + (3ax_v^2 + 2bx_v + c)\,\epsilon \\ &= P(x_v) + P'(x_v)\,\epsilon \end{aligned} \tag{4}$$

It extends to multivariate polynomials: either we have only one ϵ and two evaluations are needed:

$$\begin{aligned} Q(x_v+\epsilon, y_v) &= Q(x_v, y_v) + Q'_x(x_v, y_v)\epsilon \\ Q(x_v, y_v+\epsilon) &= Q(x_v, y_v) + Q'_y(x_v, y_v)\epsilon \end{aligned} \tag{5}$$

or each variable is attached its own ϵ and one evaluation suffices:

$$Q(x_v+\epsilon_x, y_v+\epsilon_y) = Q(x_v, y_v) + Q'_x(x_v, y_v)\epsilon_x + Q'_y(x_v, y_v)\epsilon_y$$

Dual numbers extend to non polynomial functions:

$$\begin{aligned} \exp(a+b\,\epsilon) &= e^a + be^a\,\epsilon \\ \cos(a+b\epsilon) &= \cos(a) - b\sin(a)\,\epsilon \\ \sin(a+b\epsilon) &= \sin(a) + b\cos(a)\,\epsilon \\ \tan(a+b\epsilon) &= \tan(a) + b(1+\tan^2(a))\,\epsilon \\ |a+b\epsilon| &= |a| + (\mathrm{sgn}(a)b + (1-\mathrm{sgn}(a)^2)|b|)\,\epsilon \end{aligned}$$

Dual numbers permit to compute the derivative of $D(X) = \det(M(X))$, for square matrices $M(X)$, even if entries of M are piecewise polynomials, or algorithms: just replace floating point numbers with dual numbers and then use any standard numerical method (Gauss pivot, LUP). There are also formulas.

Lemma: $\det(I + \epsilon M) = 1 + \mathrm{Trace}(M)\,\epsilon$, where $M \in \mathbb{R}^{n,n}$. Proof:

$$\det(I+\epsilon M) = (1+M_{11}\epsilon)(1+M_{22}\epsilon)\ldots(1+M_{nn}\epsilon) + R = 1 + \mathrm{Trace}(M)\,\epsilon + R$$

where R represents other perfect matchings in $I+\epsilon M$. But other matchings use at least two off-diagonal entries in $I+\epsilon M$, thus are multiples of ϵ^2, thus are zero.

When A is inversible, $\det(M(x+\epsilon)) = \det(A+\epsilon B)$ is:

$$\begin{aligned} \det(A+\epsilon B) &= \det(A(I+\epsilon A^{-1}B)) \\ &= \det(A)\det(I+\epsilon A^{-1}B) \\ &= \det(A)(1+\mathrm{Trace}(A^{-1}B)\,\epsilon) \end{aligned} \tag{6}$$

When A is not inversible, we use its SVD : $A = U\Sigma V^t$ (with Σ diagonal and U, V unitary):

$$\begin{aligned} \det(A+\epsilon B) &= \det(U\Sigma V^t + \epsilon B) \\ &= \det(U(\Sigma V^t + \epsilon U^t B)) \\ &= \det(U(\Sigma + \epsilon U^t B V)V^t) \\ &= \det(\Sigma + \epsilon U^t B V) \end{aligned} \tag{7}$$

equals the product of diagonal entries of $\Sigma + \epsilon U^t B V$. It is 0 when there are at least two null singular values in Σ. Otherwise it is

$$(\sigma_1 + k_1\epsilon)\ldots(\sigma_{n-1} + k_{n-1}\epsilon)(0 + k_n\epsilon) = 0 + \sigma_1 \ldots \sigma_{n-1} k_n\, \epsilon \tag{8}$$

The extension to many ϵ is lengthy but easy.

Dual numbers provide exact (up to floating point precision) derivatives even when equations are not available and are replaced with algorithms. Thus they make possible to use Newton method for solving, Euler method for following an homotopy curve, BFGS method for optimizing.

It is possible to compute Taylor expansions beyond degree 1 (using $\epsilon^4 = 0$), which eases Runge Kutta method for homotopy. It has a cost, reducible with the sparsity of ϵ expansions. ϵ expansions are sortable [3] with compatible orders used in Gröbner bases.

In passing, an algebraic construction ϕ starting from $\mathbb{R}$ gives the quaternions, which represent 3D rotations. If ϕ is applied to $\mathbb{R} + \epsilon\mathbb{R}$, it gives biquaternions, aka dual quaternions, which represent both 3D rotations and translations.

6 From algorithms to systems of equations

Algorithms are more convenient than equations to express constraints. But maybe algorithms can be automatically converted into systems of equations, and algorithms are just a convenience to pose equations?

Let $a \in \mathbb{R}$, and $s = \mathrm{sgn}(a)$ be the sign of a: $a = 0 \Rightarrow s = 0$, $a \neq 0 \Rightarrow s = |a|/a$. Then the system of equations below is such that $S(a,s) = 0 \Leftrightarrow s = \mathrm{sgn}(a)$.

$$\begin{cases} 0 = s^3 - s \quad \Leftrightarrow s \in \{0, -1, 1\} \\ 0 = a - sy^2 \Leftrightarrow y^2 = |a| \text{ except when } a = 0 \\ 0 = y^2 z - 1 \Leftrightarrow y^2 \neq 0 \text{ Remark that } 0 = yz - 1 \text{ also works} \end{cases}$$

The reader can check that when $a > 0$, there is only one real solution $y^2 = |a| = a$, $s = 1$, $z = 1/a$. When $a < 0$, there is only one real solution $y^2 = |a| = -a$, $s = -1$, $z = -1/a$. Finally, if $a = 0$, then $s = 0$ and y^2 is free; for uniqueness, add the equation: $(1 - s^2)(y - 1) = 0$. It changes nothing if $a \neq 0$.

Otherwise, if $a = 0$, the only real solution is $s = 0, y = z = 1$. Then it is easy to build systems $S(a, R)$ for defining $|a|$, the positive (negative) part a^+ (a^-), etc:

$$\begin{array}{l} R = |a| = sa \\ R = a^+ = \max(0, a) = (a + |a|)/2 = (a + sa)/2 \\ R = a^- = \min(0, a) = a - \max(0, a) = (a - |a|)/2 = (a - sa)/2 \\ R = \max(a, b) = (a + b)/2 + |b - a|/2 \\ R = \min(a, b) = (a + b)/2 - |b - a|/2 \end{array}$$

Then we can convert the if-then-else instruction: $F =$ (if $x > 0$ then P else if $x < 0$ then N else Z) into equations:

$$F = \mathrm{sgn}(x^+)P + \mathrm{sgn}(x^-)N + (1 - (\mathrm{sgn}(x^+) + \mathrm{sgn}(x^-))Z$$

For translating the arithmetic constraint $x \in \mathbb{Z}$ into equations, we can use the equation $\sin(\pi x) = 0$, which indeed describes $\mathbb{Z}$, but it is not algebraic. An algebraic system is: $x = x_0 + 2x_1 + \ldots 2^n x_n$ and $x_i(1 - x_i) = 0$ for all $i \in [[0, n]]$. The system has logarithmic size in 2^n, but it describes only integers in $[[0, 2^n - 1]]$. The naive representation: $x(x - 1) \ldots (x - 2^n - 1) = 0$ is exponential size.

After functional programming, assignments and iterations are useless. It is sufficient to consider fixpoints : $F(F(...(X)))$, where F is some algorithm. Assume the program $Y = F(X)$ is represented with some system of equations: $S(X, Y) = 0$. Then fixpoints of F are solutions of the system $S(X, X) = 0$. The latter clearly shows that the sizes of X and Y must be equal. It is untrue for some algorithm F, *e.g.,* for subdivision curves, the size of Y is twice the size of X.

We only sketched this conversion: we did not treat the conversion of function calls, or of data structures like Lisp pairs. Assume this conversion is possible under mild assumption. Then Computer Algebra applies to resulting polynomial systems, *e.g.,* ideals and radicals concepts become relevant for piecewise polynomials after all. We can deduce equations from algorithms, *e.g.,* for the distance between a point and a segment, or for canceling spurious roots. Thus algorithms are just a convenient way to pose equations. Is it possible to find (Gröbner bases of) polynomial preconditions for the algorithm to work, and to fail?

References

1. Barki, H., Cane, J.M., Garnier, L., Michelucci, D., Foufou, S.: Solving the pentahedron problem. Computer-Aided Design 58, 200–209 (2015)
2. Gouaty, G., Fang, L., Michelucci, D., Daniel, M., Pernot, J.P., Raffin, R., Lanquetin, S., Neveu, M.: Variational geometric modeling with black box constraints and dags. Computer-Aided Design 75, 1–12 (2016)
3. Michelucci, D.: An epsilon-arithmetic for removing degeneracies. In: 12th Symposium on Computer Arithmetic. p. 230. IEEE (1995)

Dependence of Axioms for Weak Geometries Proved Syntactically

Victor Pambuccian

School of Mathematical and Natural Sciences (MC 2352)
Arizona State University - West Campus
P. O. Box 37100
Phoenix, AZ 85069-7100
U.S.A. pamb@asu.edu

Abstract. This talk will focus on two separate topics. The first one concerns the geometry of point-reflections, and the various axioms that one can add to an axiom system for point-reflections valid in absolute geometry to get a hierarchy of geometries that are intermediate between absolute an affine geometry. The language in which the axioms are expressed is a one-sorted one, with variable to be interpreted as 'points', with two binary operation symbols, σ — with $\sigma(ab)$ standing for the reflection of b in a — and μ — with $\mu(ab)$ standing for the midpoint of ab. All statements considered are universal. The dependencies and independencies involved were all established with the aid of Tipi, an aggregate of automatic theorem provers designed by Jesse Alama. The results were published in [1]. Several open problems remain.

The second part is a walk through results known to be true, regarding statements that are known to be equivalent to established axioms, but for which we have only algebraic proofs, or even proofs using Tarski's principle and other deep model-theoretical results. The challenge would be to find syntactic proofs by means of automatic theorem provers. Examples are: Szmielew's [7] proof that the circle axiom implies the Pasch axiom in a certain axiom system for semi-ordered Euclidean geometry, the fact that the Erdős-Mordell theorem is equivalent to the statement that the angle sum in a triangle is not larger than two right angles, proved in [4], the fact that the distances from a point to the vertices of an equilateral triangle satisfy the generalized triangle inequality is equivalent to the same statement that the angle sum in a triangle is not larger than two right angles, proved in [2] and [6], the equivalence of Lagrange's axiom with Bachmann's *Lotschnittaxiom* (stating that a quadrilateral with three right angles closes), proved in [5], or the equivalence of the *Lotschnittaxiom* with the universal statement "In an isosceles triangle with base angles of 45°, the altitude to the base is smaller than the base" (proved in [3]).

References

1. Alama, J., Pambuccian, V.: From absolute to affine geometry in terms of point-reflections, midpoints, and collinearity, *Note di Matematica*, 36: 11–24 (2016).

2. Barbilian, D. : Exkurs über die Dreiecke, *Bulletin Mathématique de la Société des Sciences Mathématiques de Roumanie* 38: 3–62 (1936).
3. Pambuccian, V.: Zum Stufenaufbau des Parallelenaxioms, *Journal of Geometry*, 51: 79-88 (1994).
4. Pambuccian, V.: The Erdős-Mordell inequality is equivalent to non-positive curvature *Journal of Geometry*, 88: 134 139 (2008).
5. Pambuccian, V.: On the equivalence of Lagrange's axiom to the *Lotschnittaxiom*, *Journal of Geometry*, 95: 165–171 (2009).
6. Pambuccian, V.: On a paper of Dan Barbilian, *Note di Matematica*, 29: 29–31 (2009).
7. Szmielew, W.: The Pasch axiom as a consequence of the circle axiom, *Bulletin de l'Académie Polonaise des Sciences. Série des Sciences Mathématiques, Astronomiques et Physiques*, 18: 751–758 (1970).

Implementing Automatic Discovery in GeoGebra

Extended Abstract

Miguel A. Abánades[1], Francisco Botana[2], Zoltán Kovács[3], Tomás Recio[4] and Csilla Sólyom-Gecse[5]

[1] Depto. de Economía Financiera y Contabilidad e Idioma Moderno
Universidad Rey Juan Carlos, Madrid
miguelangel.abanades@urjc.es
[2] Depto. de Matemática Aplicada I, EE Forestal, Campus A Xunqueira
Universidade de Vigo, Pontevedra
fbotana@uvigo.es
[3] Private University College of Education of the Diocese of Linz, Austria
zoltan@geogebra.org
[4] Depto. de Matemáticas, Estadística y Computación
Universidad de Cantabria, Santander
tomas.recio@unican.es
[5] Babeş-Bolyai University, Cluj-Napoca, Romania
solyom-csilla@yahoo.com

Abstract. A prototype for the automatic discovery and derivation of elementary geometry statements, based on the implementation of computational algebraic geometry methods onto the dynamic geometry program GeoGebra, is presented. The emphasis is placed on the diverse mathematical and educational challenges posed by the—apparently straightforward—application of some well known algorithms onto the GeoGebra framework.

Keywords: Automatic Theorem Proving, Automatic Theorem Discovery, Automatic Theorem Deduction, Computational Algebraic Geometry, Elementary Geometry, Dynamic Geometry Software, GeoGebra

1 Automatic Discovery in GeoGebra

In this paper we address the automatic discovery and derivation of elementary geometry statements onto the dynamic geometry program GeoGebra, by implementing the computational algebraic geometry methods described in [3]. We refer the reader to such work for technical details about the main concepts and algorithms. We remark that the considered methods are constrained to work under an algebraically closed field of coordinates, e.g. the field of complex numbers; the validity of the output on the real plane is subject to future development.

Since September 2014, with the launching of version 5.0, GeoGebra includes proving capabilities based on symbolic algebraic computations through the command `Prove[`*<Boolean Expression>*`]`[6], where a Boolean expression is a GeoGe-

[6] See `https://www.geogebra.org/wiki/en/Prove_Command` for documentation.

bra statement with possible logical values true/false (e.g. `AreCollinear(E,F,G)` for points E, F, G in a diagram). Alternatively, in GeoGebra, one can check whether a Boolean expression is true or not simply by typing the said expression in the command line. In this case, GeoGebra uses the numerical coordinates of all the basic elements in the construction to compute the value of the given Boolean expression. However, the `Prove` command assigns symbolic coordinates (i.e. variables $x, y, u, v, \ldots$) to the free and semi-free points in the construction, and uses symbolic methods to determine whether the proposed statement is true or false in general. This way `Prove[`*<Boolean Expression>*`]` returns whether the given Boolean expression is true or false for any numerical instance of the construction. For example, if we start with a triangle ABC and the intersection point O of the altitudes from points A, B and type `Prove[ArePerpendicular[Line(O,C),Line(A,B)]]` we obtain true, meaning that **for any triangle**, its three altitudes share one point (more details in [1]).

The new feature we are reporting here, that of automatic discovery in GeoGebra, requires that the user first constructs a geometric diagram with GeoGebra's *drawing tools* or *drawing commands*. Although theoretically all *algebraic* constructions (i.e. those composed of elements that can be expressed by polynomial equations) can serve as initial data for GeoGebra's discovery tool, technical reasons, mainly related to computational time limitations, restrict the applicability of the tool for complicated constructions. Moreover, it is important to note that non-algebraic elements, such as the graph of a sine function, fall out of the scope of the method, algebraic in nature.

After constructing a geometric diagram the user needs to type the command `LocusEquation[`*<Boolean Expression>, <FreePoint>* `]`[7] with two parameters: the sought thesis T (which must be a Boolean expression) and a free point P 'supporting' the discovery. Remark this is a drastically new functionality, available in GeoGebra, version 5.0.213.0, since March 2016, of the preexisting command `LocusEquation`. We have decided to extend the features of the command rather than adding a new name to the long list of GeoGebra commands.

The Boolean expression defining the discovery plays the part of the *extra condition* that we require our diagram to satisfy. The coordinates of the free point P, second parameter of the command `LocusEquation`, are those the sought extra hypothesis will deal with. That is, the symbolic coordinates of P will be the variables of the polynomials conforming the necessary conditions obtained as a result of the discovery process. As a result, `LocusEquation[T,P]` will produce a set V (providing its implicit equation) such that "if T is true then $P \in V$". Notice that this set can be empty (if the result of the algorithmic elimination yields the polynomial equation $1 = 0$, or it can be the whole plane, if the equation is $0 = 0$.

[7] See `https://www.geogebra.org/manual/en/LocusEquation_Command`.

For instance, let the constructed geometrical figure consist of the following steps:

Step	Textual description	Tool	Icon	Command
1.	Let A be a free point.	**Point**		`A=(1,2)`
2.	Let B be a free point.	**Point**		`B=(3,4)`
3.	Let C be the midpoint of segment AB.	**Midpoint or Center**		`C=Midpoint[A,B]`
4.	Let a be the segment AC.	**Segment**		`a=Segment[A,C]`
5.	Let D be a free point.	**Point**		`D=(5,6)`
6.	Let b be the segment BD.	**Segment**		`b=Segment[B,D]`

Now (as step 7) the user types `LocusEquation[a==b,D]` to learn how to choose point D in order for the construction to satisfy the extra condition $a = b$. Clearly, the output must be the circle with center B and radius a. That is, if the sought thesis $a = b$ holds, then D has to be an element of this circle. In this particular case the implicit curve $c : -x^2 + 6x - y^2 + 8y = 23$ is drawn by GeoGebra, which can be rewritten as $(x-3)^2 + (y-4)^2 = \sqrt{2}^2$.

It is crucial to highlight that points A and B are fixed in this example when obtaining the implicit locus, but D is considered a free point. This means that the numeric coordinates of A and B were used during the computations, but the initial numeric coordinates of D were ignored, and the symbolic coordinates (x, y) for D were used instead. That is, GeoGebra computed the (polynomial) conditions that the symbolic coordinates of point $D = (x, y)$ have to satisfy in order for the expression $a = b$ to be true for the particular case when $A = (1, 2)$ and $B = (3, 4)$.

The implicit curve element newly obtained as a result of the discovery process behaves interactively as any GeoGebra element. When changing the coordinates of A or B, say making $B = (0, -1)$, the implicit curve will be recomputed and $c : -2x^2 - 2y^2 - 4y = -3$ will be obtained (which can be rewritten as $(x-0)^2 + (y+1)^2 = \left(\sqrt{10}/2\right)^2$). Since the computation is very fast—enough for GeoGebra to compute the implicit curve in a large number of cases while the user drags point B from $(3, 4)$ to $(0, -1)$—the visual result is that the drawn curves are circles in all particular cases. Thus the user can visually conclude that the implicit locus is *always* a circle, independently of the input fixed points A and B (see Fig. 1).

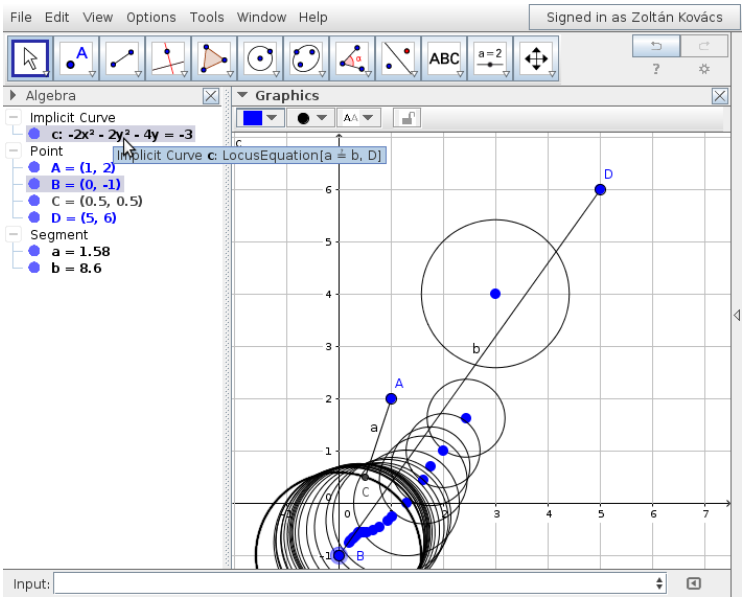


Fig. 1. Dragging point B from $(3, 4)$ to $(0, -1)$ and conjecturing that the implicit locus is always a circle. (Here tracing was enabled for the point B and also for the implicit locus c.)

As mentioned above, the `LocusEquation[T,P]` command produces extra necessary conditions, say H', over P for T to be true for the initial construction. To formally check that the found necessary conditions are also sufficient, the user needs to use the `Prove` command with T as thesis for the geometric diagram modified so that P satisfies H'. For instance, in the example above, we have obtained '$H' = D$ is in the circle with center B and radius a', so to prove the conjecture '$a = b$ if H'' we would type `Prove[a==b]` for the construction modified so that D is defined to be in the circle with center B and radius a.

Technically speaking, the implementation of `LocusEquation` uses GeoGebra's theorem proving subsystem to translate the geometric construction into a complex algebraic geometry model and also to obtain an algebraic equation as a representation of the algebraic variety, hence all tools with proof capabilities can also be used to obtain an implicit locus [4].

1.1 Examples

`https://www.geogebra.org/book/title/id/mbXQuvUV` points to a GeoGebraBook where we describe some examples of use of the `LocusEquation` command in GeoGebra. They show how we can (re-)discover and generalize geometric results while pointing out some precautions that one has to keep in mind while using this automated geometric discovery tool.

Example *Discovering the Simson-Wallace theorem* in the above GeoGebraBook shows how `LocusEquation` can be used to discover highly non-trivial geometric results. In particular, the classical Wallace-Simson theorem is (re-)discovered.

In example *A generalization of the right triangle altitude theorem*, a natural generalization of the known right triangle altitude theorem is obtained for non-right triangles.

The examples in *Dependency of locus sets on the initial construction 1–2* emphasize the key idea of the dual symbolic-numeric nature of the protocol behind `LocusEquation`. They show how the numeric nature of the protocol materializes in a subtle dependence of the produced necessary conditions on the initial (numeric) coordinates of the basic elements in the construction; in such a way that for some initial construction we might find that the sought locus is a line, while for other initial configuration it is a circle. An example is also provided in this section that deals with the important special cases in which a property is never/always true. It shows how the answer by GeoGebra in these cases takes the form of a trivially false/true equation.

Being `LocusEquation` based on algebraic methods, it can sometimes produce sensible conditions from an algebraic point of view that are geometrically meaningless. That is the case of most of the instances associated to the locus set in example *Degenerate cases and necessary not-sufficient conditions*, producing geometrically degenerate situations due to the coincidence of two points.

2 Further developments

We think that featuring automatic discovery tools, in the generalized sense we have described in section 1, combined with automatic proving, in a widely accesible Dynamic Geometry programa such as GeoGebra, is an interesting feature, deserving some consideration in the realm of mathematics education, for instance, in the context of intelligent tutoring. The previous list of examples shows, on the one hand, the power of the implemented tool; and, on the other hand, the subtleties and limitations that are, sometimes, involved in its use. What follows are some reflections on further developments we are currently working on.

2.1 Extending the number of discovery points

Firstly, while the proposed protocol for discovery is naturally aimed to search for necessary conditions involving more than one point, the `LocusEquation` command only accepts exactly one point, thus severely limiting the scope of discovery. If necessary conditions on a construction are to be discovered, and there is exactly one point involved in the discovery, the result, if found, returns an implicit curve, which can be plotted, thus graphically suggesting the conjecture. Nevertheless, it is easy to pose questions where there are several (and not necessarily free) points a priori unknown on which the discovery is to be performed. Consider, for instance, a triangle ABC and the midpoints D and E of sides

AB and AC, respectively. If one asks for the locus of C such that side AC is perpendicular to line DE, GeoGebra returns the circle with diameter AB, that coincides with the circle circumscribed to triangle ABC, when C lies in the locus. But, relaxing point D to just lie on side AB, the `LocusEquation` command does not give any answer. Point D is not a free point in the construction, thus its coordinates are not replaced by their actual numeric values. One of its coordinates is a constant, and the other one can have any value. So, the algebraic machinery would return 0 as discovery result, meaning that vertex C can be placed in any part of the plane in order to satisfy the required perpendicularity. The set of points to be discovered should include D, violating the actual syntax of `LocusEquation` command. In such a case, that is, trying to discover conditions on C and D in order to get the perpendicularity, a possible command could be `Discover[ArePerpendicular[AC,DE],{C,D}]`. The algebraic output for this multi-point discovery, being $A(0,0)$ and $B(1,0)$, consists of a pair of polynomials, where $C(x_1, x_2)$ and $D(x_3, x_4)$:

$$x_4,$$

$$x_1^2 + x_2^2 - 2x_1x_3.$$

Thus, point D lies on side AB, as it was defined, and point C lies on a circle centered at D and passing through A (see Fig. 2). Nevertheless, currently GeoGebra has no means to translate this algebraic result into a meaningful assertion.

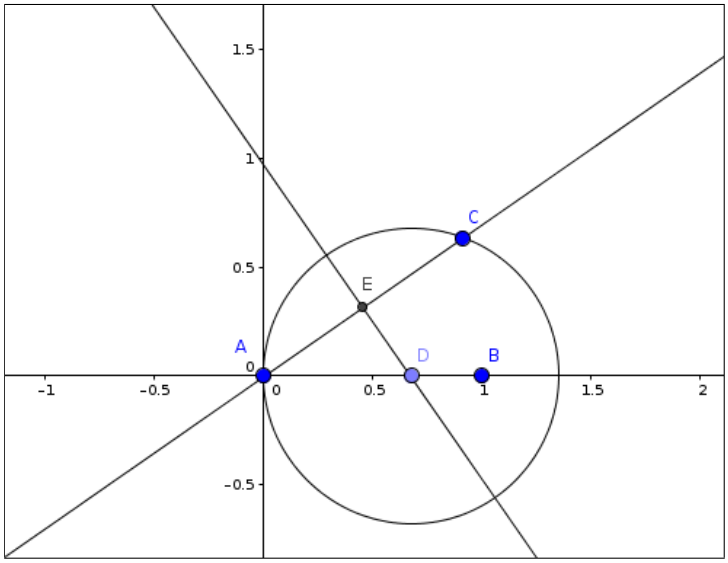


Fig. 2. What is the locus of C such that line AC is perpendicular to line DE?

This example shows that, in some sense, what we are considering as an extension of the `LocusEquation` command, is the possibility of pointing out several coordinates for discovery (such as one for the semi-free point D and two coordinates for C), not just the two of the single point P, whose locus we are searching for. Moreover, we should also include, in the extended discovery features, the possibility to find the locus, not only of points, but also of some geometric objects such as lengths, distances, areas, etc. I.e. we could be asking for the polynomial equation that a length or a distance must satisfy if some thesis is to become true.

In particular this could be very useful for *automatic derivation*, since it can be considered as a discovery in a context in which there is not a new thesis, but we would like to obtain conclusions from the construction in terms of some specific variables. Say, to discover the relation between the lengths of the sides of a triangle and its area, a sort of "locus" of the lengths and the area.

2.2 Handling directly entered algebraic inputs

The theorem proving subsystem is designed to work with geometric inputs like parallel lines, perpendiculars, intersection of circles, and so on, but not to handle algebraic descriptions like $y = x^2$ to define a parabola. This means that a directly entered input equation of a geometric curve cannot be handled at the moment. On the other hand, an algebraic equation is considered to be the natural way for its *output*, so that the result of the `LocusEquation` command is internally always an algebraic equation (which is visualized by another subsystem in GeoGebra), independently from computing which type of locus (explicit or implicit).

As a result, unfortunately, the algebraic output cannot be used as *another input* for the theorem proving subsystem at the moment. This means that, from the perspective of automated proving, the user cannot directly investigate the implicit curve created by the LocusEquation command. It requires to make a new construction, including the conjectural locus, and then use the `Prove` command. To simplify this issue, the theorem proving subsystem has to be extended to make it possible to handle direct algebraic inputs as well. See `https://jira.geogebra.org/browse/TRAC-3596` for more on this.

2.3 3D loci

A further step is to generalize locus computations to work in 3 dimensions. There are no theoretical issues concerning this, but computationally we cannot expect fast results for the moment. In general the number of coordinates are 50% more than for the 2D case, and this will increase the number of variables by 50%. Considering that the underlying computer algebra system uses Gröbner bases for manipulating the polynomials, and in worst case it is doubly exponential in the number of variables, we cannot be optimistic for the general case. Nevertheless, technically it is still possible to generalize the current 2D implementation to be 3D ready to harness GeoGebra's attracting features to visualize 3D objects.

2.4 Enlarging Boolean expressions

A more promising possibility for further improvement is to enable using disjunctions for Boolean expressions as the first parameter of the `LocusEquation` command. Clearly, for an arbitrary disjunction A or B, `LocusEquation[A||B,...]` should be the same as the union of `LocusEquation[A,...]` and `LocusEquation[B,...]`, and algebraically the product of the two locus equation polynomials will describe the polynomial of the union. On the other hand, a similar strategy for conjunctions may be technically more difficult, because a symbolically reliable display of the intersection of two algebraic curves is a more challenging task.

2.5 Unification parts of the source code

GeoGebra supports symbolic computation of *explicit* locus equations since version 4.2 (December 2012) by the result of the collaborative work of some of the authors and other researchers (see [2]). Independently from that research, another module of GeoGebra was started in 2011 by the joint work of the authors and other developers, namely the theorem proving subsystem including the `Prove` command. It was published in version 5.0 (September 2014) and is being improved continuously since then. The *implicit* locus equation command—presented in this paper—uses the same module.

The two subsystems were developed separately and have no common parts. Meanwhile the first module was not improved any longer, but recently it turned out that unifying the codebases of the two projects is achievable. Such a unification would be fruitful to avoid double implementation of translating geometry statements into algebraic equations, so that user experiments may be extended also for the explicit locus equations by harnessing many advanced features of the second module, including direct definition of conic sections or geometric inversion.

Fortunately, the GeoGebra Team decided to officially support this work from 2016 and the codebase has been started to be merged. See `https://jira.geogebra.org/browse/GGB-641` for details. We expect fruitful results shortly.

Acknowledgement

First, second and fourth authors are partially supported by the Spanish Research Project 'Construcciones algebro-geométricas: fundamentos, algoritmos y aplicaciones' (MTM2014-54141-P).

References

1. F. Botana, M. Hohenwarter, P. Janičić, Z. Kovács, I. Petrović, T. Recio, S. Weitzhofer: Automated theorem proving in GeoGebra: Current achievements, *Journal of Automated Reasoning* 55: 39–59, 2015.

2. F. Botana, Z. Kovács: A Singular web service for geometric computations, *Annals of Mathematics and Artificial Intelligence* 74(3-4): 359–370, 2015.
3. G. Dalzotto and T. Recio: On protocols for the automated discovery of theorems in elementary geometry, *Journal of Automated Reasoning* 43(2): 203–236, 2009.
4. Z. Kovács, C. Sólyom-Gecse: GeoGebra Tools with Proof Capabilities. `http://arxiv.org/abs/1603.01228`. March, 2016.

Geodesic Star Unfolding

Md. Ashraful Alam[1] and Ileana Streinu[1,2]

[1] Computer Science Department, University of Massachusetts Amherst, Amherst, MA 01003, USA
[2] Computer Science Department, Smith College, Northampton, MA 01063, USA

Abstract. If we draw a non-crossing tree on the surface of a convex polyhedral surface in R^3 such that it touches all the vertices of the polyhedron, and then cut along the tree edges, the resulting surface can be unfolded onto the Euclidean plane, possibly with overlap. A well studied case, called *star unfolding*, cuts along the shortest paths from a source point to all vertices. This is one of the few instances that guarantees a non-overlapping unfolding of the flat polygonal surface resulting from the cutting. In this paper, we introduce *geodesic star unfolding*, a generalization that allows cuts along arbitrary geodesics from the source point to the polyhedral vertices, rather than just the shortest paths. We discuss a number of geometric properties that transfer from the shortest-path version, and others (such as non-overlapping) that do not. Some of these properties were identified via experimentation with an interactive Mathematica implementation.

1 Introduction

In this paper we introduce a new type of polyhedral unfolding, called *geodesic star unfolding*, and initiate the study of its properties. We focus on the comparison to the well studied case when the geodesics are all shortest paths. Several distinguishing properties are presented via examples obtained by experimentation with an interactive Mathematica implementation.

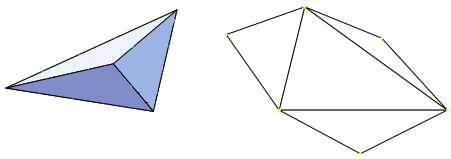


Fig. 1. A classical example of an edge unfolding obtained by cutting the tetrahedron along a spanning tree of its 1-skeleton.

Unfolding. Let P be a convex polyhedron in R^3 on whose surface we draw a collection of non-crossing geodesic arcs forming a tree and touching all the vertices of the polyhedron (this is called a *cut tree*). If we cut the surface along the tree edges, we obtain an intrinsically flat surface with boundary which is topologically a disk. An *unfolding* is obtained by immersing the cut surface into the 2D plane such that it is locally non-overlapping. The boundary becomes a planar polygon which, in general, may not be simple (i.e. may be self-intersecting) due to global self-overlaps of the immersed unfolded surface.

Non-overlapping unfoldings. Different types of unfoldings are distinguished by the choice of the cut tree. The *edge unfolding*, illustrated in Fig. 1 for a tetrahedron, uses a spanning tree of the 1-skeleton of the polyhedron. An edge unfolding is often non-overlapping, as in Fig. 1, but this does not always happen; self-overlapping examples are presented in [7, 10]. A long standing open question is whether *every* convex polyhedron has *some* edge unfolding to a simple, non-overlapping polygon. Experimental evidence points to a positive answer, yet a proof or counter-example remain elusive. However, there exist specific kinds of unfoldings which are always non-overlapping: (shortest-path) *star unfolding* and *source unfolding* [5]. In this paper we introduce a generalization of star unfolding; the ultimate goal is to identify critical properties that could help prove global non-self-overlap in general unfoldings.

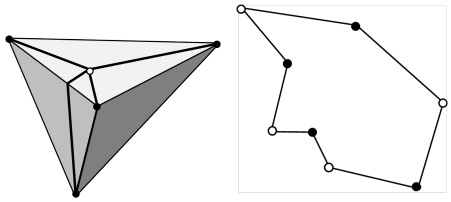


Fig. 2. A tetrahedron with the shortest paths from a source vertex, and the corresponding sp-star unfolding polygon. Source and polyhedral vertices are colored in white, resp. black.

Shortest-path star unfolding. This type of unfolding is obtained by choosing a point s (the *source vertex*) on the surface of the polyhedron P and cutting along the shortest paths (on the surface) from s to all vertices $v_1, v_2, \ldots, v_n$ of P, as in Fig. 2. Since the cut tree is a star, this is known in the literature as a *star unfolding* of P. To distinguish it from the generalization introduced in this paper, we will refer to it as the *shortest-path star unfolding*, shortly *sp-star unfolding*. It is also known as *Alexandrov unfolding* [8], and it was first mentioned in [4]. A detailed exposition can be found in [6].The non-overlapping property of the shortest path star unfolding was studied in [5] and a number of applications appear in [1]. Building on these results, we gave in [2] a complete characterization of those polygons which arise as sp-star unfoldings.

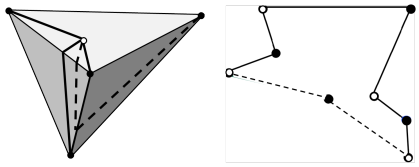


Fig. 3. One of the shortest paths on the tetrahedron from Fig. 2 is replaced by a geodesic. The dotted line indicates the geodesic which is not a shortest path.

Geodesic star unfolding. We generalize the sp-star unfolding by cutting along a star-tree whose edges are non-crossing geodesics (i.e. not necessarily the *shortest* geodesics) from the source vertex. We refer to this more general setting as a *geodesic star unfolding* (shortly, star unfolding in this paper) and use the term *geodesic star unfolding polygon* to refer to its polygonal boundary.

Contributions. Since geodesic star unfoldings are relaxed versions of sp-star unfoldings, one might anticipate some properties similar to those of the sp-star unfolding, as well as differences. In particular, *are they always non-self-*

overlapping? As a first contribution, we characterize the polygons arising as geodesic star unfoldings. We obtain self-overlapping counter-examples through experimentats using an interactive Mathematica implementation of geodesic and sp-star unfoldings, based on their corresponding characterizations.

We also address the following *reconstruction problem:* given an ordered set of points on the plane, construct a geodesic star unfolding of a convex polyhdron whose source vertices are placed at the given points. Such a reconstruction algorithm can be used to generate polyhedral metrics (in the sense of Alexandrov) without going through 3D, i.e. not by generating polyhedra and somehow cutting their surfaces, but by directly generating a 2D polygon which corresponds to an unfolding of (a unique) 3D convex polyhedron. As a second contribution, we identify a family of points which are guaranteed to support geodesic star unfoldings.

2 Preliminaries

Polygons and polyhedra. For our purposes, a *2D polygon* is a region in R^2 bounded by a simple (non self-intersecting) polygonal cycle joining an ordered sequence of points $p_1, \cdots, p_n$. The polygon is *convex* when it coincides with the convex hull of its vertices, i.e. when a line segment connecting any of its two points lies fully inside its region; otherwise, it is non-convex. A 3D polygon is flat if it lies in a plane.

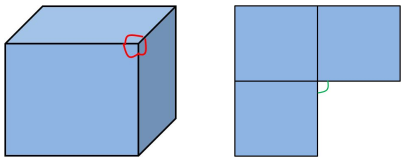


Fig. 4. The face angles of one vertex of the cube. The curvature at this point is 2π minus the sum of the face angles. When the surface is cut at this vertex and flattened in 2D, this curvature is the angle deficit at the corresponding interior vertex of the unfolding polygon.

A *3D polyhedron* is the region in R^3 bounded by a finite number of flat polygons called *faces* such that: (a) if two faces intersect, then it is only at a common edge or a vertex, (b) every edge of every face is an edge of exactly one other face, and (c) faces surrounding each vertex form a simple cycle. A polyhedron is *convex* if a line segment connecting any of its two points lies entirely in its interior. Convex polygons, resp. polyhedra can also be characterized as bounded regions obtained by intersecting a finite number of half-planes, resp. half-spaces. The boundary of a polyhedron is a surface of some genus. In this paper we work only with convex polyhedra P.

The *Gaussian curvature* (shortly, *curvature*) of a point p on a polyhedral surface is the angle deficit at p, i.e. 2π minus the sum of the face angles incident to it (Fig. 4). The vertices of a convex polyhedron have positive curvature, and the points on the faces and edges have zero curvature. The *total Gaussian curvature* is the sum of all vertex curvatures. For a surface with spherical topology, this is always 4π.

Polyhedral metric and Alexandrov's Theorem. A *polyhedral metric* is a collection of (one or more) planar polygonal pieces together with rules for glueing them, along pieces of their boundaries, into an intrinsic surface with the topology of a sphere. The glueing may result in a finite set of points of non-zero intrinsic (Gaussian) curvature, called the vertices of the surface. The metric is convex if the sum of the surface angles at each vertex is at most 2π, i.e. if the curvature is positive. The following classical result provides the connection with convex polyhedra.

Theorem 1. (A.D. Alexandrov's theorem [3]) *For every convex polyhedral metric, there exists a unique convex polyhedron (up to translation and symmetry) realizing this metric.*

As a side note, we remark that the edges of the polygonal pieces in the polyhedral metric need not be related in any way to the edges of the realization as a convex polyhedron, and that there exist infinitely many polyhedral metrics that give rise to the same convex polyhedron.

Shortest paths and geodesics. A *shortest* path between two points on a convex polyhedral surface P is the shortest of all possible curves between these two points, measured on the surface of P. Various properties of shortest paths on polyhedral surfaces have been identified in [11, 9].

A *geodesic path* between two points on the surface is a locally shortest path. In this paper we consider only non-self-intersecting (simple) geodesics and refer to them, shortly, as *geodesics*. Given two points on the polyhedral surface, there may be more than one (even an infinity of) geodesic paths between them. The shortest paths, by definition, are geodesics. However, in this paper we need to distinguish those which are not shortest paths: in short, we refer to them simply as *geodesics*. For a convex polyhedral surface, a geodesic segment may contain a point of strictly positive curvature (i.e. a polyhedral vertex) only as an endpoint (but not strictly interior to the segment). In particular, a geodesic segment is a straight line segment on some unfolding of (possibly, a cover of) the surface.

Geodesic star unfolding. We consider only unfoldings which are obtained by cutting a polyhedral surface along a tree whose edges are *disjoint* (non-crossing, simple) geodesic segments. These tree edges may be shortest geodesics, as is the case for all edge or sp-star unfoldings. The *geodesic star unfolding* (shortly, *star unfolding*) is obtained by cutting along geodesic edges connected as a star, but where *some or all* of these edges are *not shortest paths.* The common vertex s of all the edges is called the *source vertex*. The polygon obtained via a star unfolding (Fig. 2) has $2n$ vertices: n of them correspond to polyhedral vertices of P and n to copies of the source vertex (*source images*). These two types of vertices appear alternately on the boundary of the unfolding, meaning each source image has one polyhedral vertex preceeding and one following it, in cyclic order. The angle at each source image is a *source angle*. The sum of all source angles is 2π when the source vertex is on a polyhedral face or an edge and strictly

less than 2π when it is placed on a polyhedral vertex (by an amount given by the curvature of the polyhedral vertex). Given a polyhedron, a source vertex and a geodesic star, we can angularly sort the star edges around the source vertex. This induces a circular ordering of the polyhedral vertices. If we join by shortest paths the polyhedral vertices, in this order, we obtain a geodesic polygon on the surface of the polyhedron. In the case of an sp-star unfolding, this polygon is non-self-intersecting and bounds a region called the *core* of the sp-star. In the corresponding sp-star unfolding, the core unfolds as the inner polygonal region obtained by joining the polyhedral vertices in the order in which they appear on the boundary of the unfolding polygon.

Voronoi diagram. A set P_n of n points (called sites) in the Euclidean plane induces a *Voronoi* diagram $V_D(P_n)$. This is a planar subdivision of n *Voronoi regions* (one for each site), *Voronoi edges* between them and *Voronoi vertices* where the Voronoi edges meet, defined as follows. An arbitrary point $x \in R^2$ lies in the Voronoi region of site p_i if the Euclidean distance from x to p_i is smaller than the distance to any other point $p_j \in P_n, j \neq i$. There are two kinds of Voronoi edges: segments, bounded at both ends by Voronoi vertices, and rays, having just one Voronoi vertex at one end (the *closed end*) but unbounded at the other end (the *open end*).

Ridge tree. Given a source point s on the surface of a convex polyhedron P, the *ridge tree* is the set of all the points $r \in P$ for which there exist at least two distinct shortest paths from r to s on the surface. It is known that its closure is a tree which touches all the vertices of the polyhedron. Another remarkable property (see [5]) relates the sp-star unfolding to the ridge tree (from the same source vertex), as follows. Take the images of the source vertex in the sp-star unfolding and compute their Voronoi diagram. Then delete the part that lies outside the sp-star unfolding polygon: what remains is a tree which, on the (folded) 3D polyhedron surface, is precisely the ridge tree for source s.

3 Characterization of geodesic star unfoldings

In this section we prove a very simple characterization of (geodesic) star unfoldings, and remind the reader of additional features of sp-star unfoldings. The basic properties of a geodesic star unfolding mentioned in the Introduction lead to the following definition:

Su-Polygon. An *su-polygon* (short for *abstract star-unfolding polygon*[3]) is a disk-like zero-curvature (intrinsically flat) surface, whose boundary satisfies the following properties (known to hold by any geodesic star unfolding polygon): (a) it has $2n$ vertices, alternately labeled as s_i and v_i and referred to as s-, resp. v-vertices. The angle (interior to the surface) at an s-vertex is refered to as an s-angle; (b) the sum of all the s-angles is 2π, and (c) the v-vertices are placed

[3] The modifier *"abstract"* is used to emphasize that, *a priori*, such polygons are not guaranteed to arise from star unfoldings of 3D convex polyhedra.

on the perpendicular bisectors of the two neighboring s-vertices. In particular, this last property implies that the two edges incident to a v-vertex have equal lengths. Intuitively, the s-vertices, v-vertices and s-angles are analogous to source images, polyhedral vertices and source angles for the sp-star unfolding polygons.

We remark that the definition allows for the self-overlap of the unfolded surface, i.e. the su-polygon is not necessarily a simple polygon. Su-polygons are special cases of polyhedral metrics, under the rule that the two equal sized edges incident to a v-vertex are glued together.

Since the boundary edges of an sp-star-unfolding polygon must be the shortest paths on the corresponding polyhedral surface, not all su-polygons arise as sp-star-unfoldings. However, every su-polygon arises from a geodesic star unfolding of some convex polyhedron with respect to some source vertex. This follows immediately from the following straightforward lemma by applying Alexandrov's Theorem:

Lemma 1. *An su-polygon is a convex polyhedral metric.*

Therefore, an su-polygon can be folded into a convex polyhedron. After folding, its boundary edges correspond to the cut edges on the polyhedral surface. Since these cut edges are straight lines, they are also geodesics (some of which may be the shortest) from the source to the polyhedral vertices. Therefore, we have the following characterization:

Theorem 2. *A polygon (not necessarily simple) is the geodesic star unfolding of some convex polyhedron from a source vertex if and only if it is an su-polygon.*

4 Comparison of Geodesic and Sp-Star Unfoldings

Although geodesic star unfoldings share some basic properties with sp-star unfoldings, there are several important differences. We present now a series of properties that hold for sp-star unfoldings but not for arbitrary geodesic ones. We need the following characterization from [2]:

Theorem 3. (Characterization of sp-star unfoldings) *A simple su-polygon is an sp-star unfolding polygon if and only if (a) the perpendicular bisectors of the pairs of consecutive s-vertices are all present in the Voronoi diagram of all the s-vertices; and (b) any v-vertex lies precisely on the Voronoi edge corresponding to its two s-vertex neighbors.*

Source angles. In an sp-star unfolding all source angles are convex (smaller than π). This is often the case for geodesic non-sp star unfoldings, but not always:

Lemma 2. *There exist geodesic non-sp star unfoldings with one reflex (larger than π) source angle.*

Proof. An example of a geodesic star unfolding polygon with one source angle larger than π is shown in Fig. 3. Since the angle sum at the source vertex is at most 2π, at most one source angle can be reflex.

Placement of polyhedral vertices. The characterization of sp-star unfolding polygons [2] states that if v_i is the v-vertex between two s-vertices s_i and s_{i+1}, then the Voronoi diagram of the source vertices contains an edge between these two sites, and, moreover, v_i lies precisely on this Voronoi edge (segment or ray). This property depends on the "shortest path" assumption and does not extend to arbitrary geodesic star unfoldings.

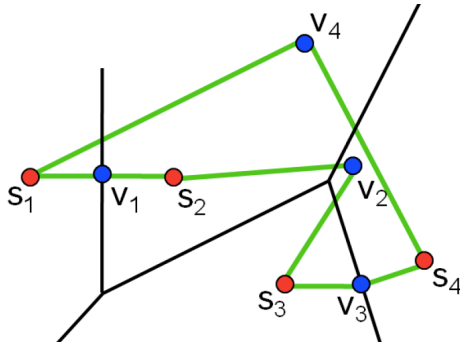


Fig. 5. An example of a geodesic star unfolding with v-vertices that do not lie on Voronoi edges: the polyhedral vertex v_2 is on the extension of a Voronoi segment and the bisector on which v_4 lies is not present in the Voronoi diagram.

Lemma 3. *There exist geodesic star unfolding polygons where some perpendicular bisectors on which the polyhedral vertices lie are not part of the Voronoi diagram.*

Lemma 4. *There exist geodesic star unfolding polygons where some polyhedral vertex lies not on its corresponding Voronoi segments, but on an extension of the underlying bisector.*

Proof. Both cases are illustrated in Fig. 5. □

Ridge tree. In the case of an sp-star unfolding, we have seen that the ridge tree is part of the Voronoi diagram of the source images. The above counter-examples indicate that this may not be the case for arbitrary geodesic star unfoldings.

Self-overlapping geodesic star unfolding. It is known that sp-star unfoldings are non-self-overlapping. On the other side:

Lemma 5. *There exist self-overlapping geodesic star unfolding polygons.*

Proof. The example in Fig. 6 has been obtained by experimenting with an interactive implementation of su-polygons in Mathematica. □

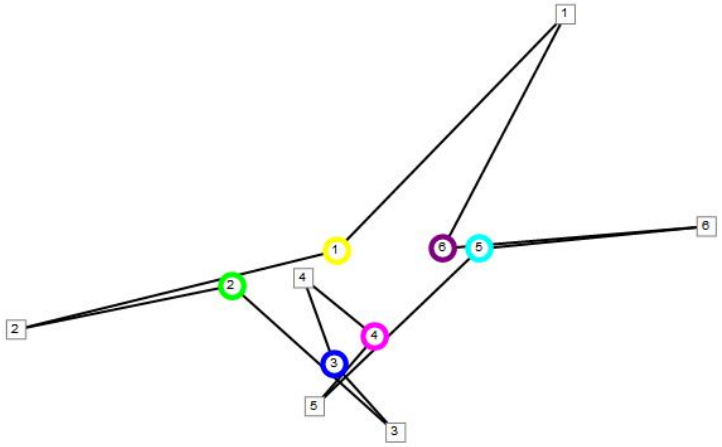


Fig. 6. An example of a self-overlapping geodesic star unfolding polygon. Polyhedral, resp. source vertices are shown with colored circles, resp. white squares.

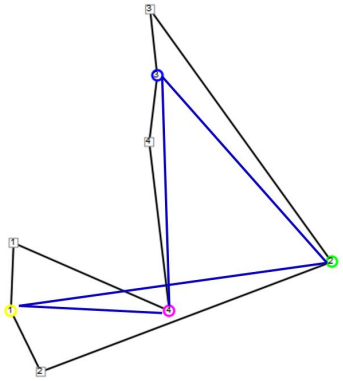


Fig. 7. An example of a geodesic star unfolding where the core polygon, shown in blue, is self-intersecting.

Self-intersecting core. The core of an sp-star unfolding joins the v-vertices and is a simple polygon [6]. Furthermore, in this case the core, as a polygonal region, contains the ridge tree inside. On the other hand:

Lemma 6. *There exist convex polyhedra, source vertices and non-crossing geodesics from the source to the polyhedral vertices such that the core of the resulting geodesic star unfolding polygon is self-intersecting.*

Proof. An example, obtained by experimentation with our interactive Mathematica implementation of su-polygons, is shown in Fig. 7. □

5 Reconstruction of geodesic star unfolding polygons

Based on the characterization given in Theorem 3, we presented in [2] a linear time algorithm that constructs sp-star unfolding polygons with prescribed combinatorics, as captured in the (topologically embedded) ridge tree. It turns out that our algorithm always produces an sp-star unfolding polygon whose s-vertices lie in *convex* position. We also know that this property, of having the source vertices in convex position, does not hold for all sp-star unfoldings polygons. On the other hand, if we are given an ordered point set in convex position,

is it true that it supports some sp-star unfolding? This question was answered in the negative in [2]. Here, we investigate the extent to which such reconstruction questions can be answered for geodesic star unfoldings, where we encounter new challenges.

Challenges. We have shown in [2] that not all ordered sets of s-vertices support sp-star unfolding polygons, and in particular not all that are in convex position. For this purpose, we introduced the concept of a *flap polygon*, which satisfies the same properties as a sp-star unfolding polygon except that the source vertex angle sum may be arbitrary (in particular, larger than 2π). We defined the *extreme flap polygon*, whose v-vertices are placed at Voronoi vertices in a specific manner that minimizes the source angle sum; when these vertices are moved away from the Voronoi vertex on incident Voronoi edges, this angle sum increases. For certain source point sets, the angle sum of the extreme flap polygon exceeds 2π, hence they do not support any sp-star unfolding polygon.

However, this constraint is not prohibitive for arbitrary geodesic star unfoldings: here, the polyhedral vertices are not constrained to lie on corresponding Voronoi segments. Hence if the sum of the s-angles of an extreme flap polygon is larger than 2π, we can push some v-vertices inwards until the s-angles sum up to 2π. This causes some v-vertices to be placed on the *line extension* of their corresponding Voronoi segments. Theorems 3 and 2 then ensure that the resulting polygon is a geodesic star unfolding.

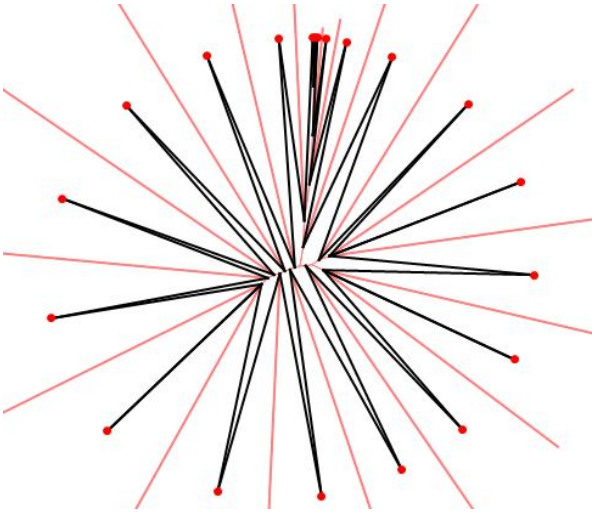


Fig. 8. An example of a extreme flap polygon whose s-angle sum is larger than 2π. It is difficult to push the v-vertices inwards (on the line extensions of the corresponding Voronoi edges) to achieve a sum of the s-angles equal to 2π, while maintaining a valid geodesic star unfolding. Note that in this example, the s-vertices are in convex position and hence all the consecutive bisectors are part of the Voronoi diagram.

A problem with this approach is that we do not know how far we can push each v-vertex inwards; the whole process is a trial-and-error heuristic. There are no geometric properties that will guarantee that we can push a set of v-vertices inwards enough to make the sum of s-angles to 2π without crossing through other vertices or edges. Such a challenging example is presented in Fig. 8. This complicates the development of an algorithm for the construction of arbitrary geodesic star unfoldings on given point sets. A natural question thus arises:

Are there point sets guaranteed to support sp-star, resp. geodesic unfoldings?

In particular, point sets in convex position are good candidates. However, we have shown in [2] that there exist cyclically ordered point sets in convex position which do not support sp-star unfoldings. But do they always support geodesic star unfoldings? The example in Fig. 8 shows that this may not be easy to decide.

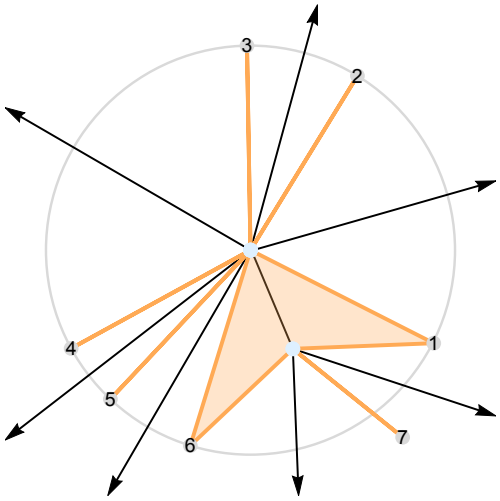


Fig. 9. An "almost circular" point set of type "out" ($i = 7$), its Voronoi diagram (black) and extreme su-polygon (orange).

Point sets guaranteed to support sp-star unfoldings. However, there exist special set of points, designated as s-vertices, such that the v-vertices can freely move on the extension of the Voronoi segments while seeking positions satisfying the angle sum condition. One such case is when all s-vertices lie on a circle. The bisectors of all pairs of consecutive s-vertices are part of the Voronoi diagram, and they all meet exactly at the center of the circle (i.e. the Voronoi diagram of these points has n line segments which meet at the center of the circle). The sum of the s-angles of the extreme flap polygon is zero, hence any displacement of a v-vertex on the corresponding Voronoi segment leads to a valid su-polygon.

Almost circular point sets. There are no other known general families of points that guarantee sp-star unfoldings. We show now that a small relaxation of circularity leads to guaranteed sp, resp. geodesic star unfoldings. An ordered point set $s_1, \cdots, s_n$ is called "almost circular" if all points lie on a circle C of center c, in this order, except for one point, say s_i, which lies somewhere inside the wedge centered at c and bounded by the rays (c, s_{i-1}) and (c, s_{i+1}), and the wedge is convex. The set is said to be of type "out" when s_i lies outside (Fig. 9), and of type "in" when s_i lies inside the circle C (Fig. 10).

Theorem 4. *An "almost circular" point set of type "out" always supports an sp-star unfolding, and one of type "in" always supports a geodesic star unfolding.*

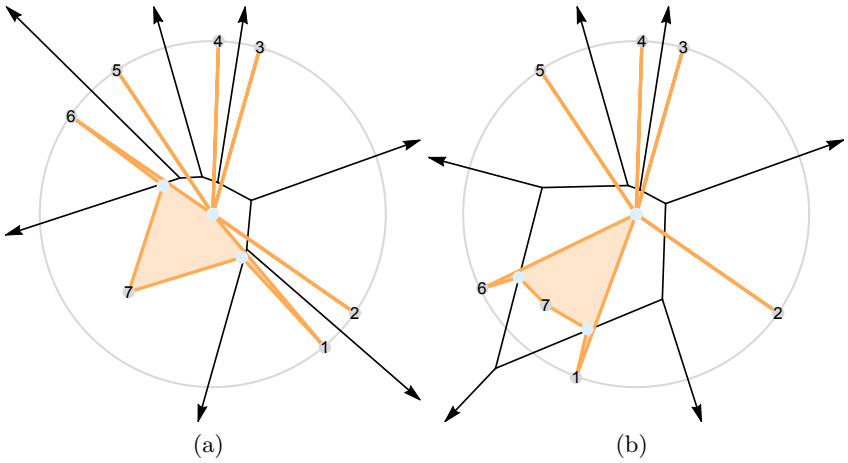


Fig. 10. Two "almost circular" point sets ($i = 7$) of type "in": (a) convex and (b) non-convex. The Voronoi diagram is shown in black. In orange is shown the *adapted* geodesic su-polygon, with all the source vertices at the center of the circles, except for the two incident with the special source vertex (shown slightly apart from the extreme position, for clarity).

Proof. (Sketch) Let c be the center of the circle and assume, without loss of generality, that the points are labeled consecutively such that let $s_1, \cdots, s_n$ lie on the circle, except s_i (the "special" source vertex) which lies somewhere inside the wedge (s_{i-1}, c, s_{i+1}). We place the v-vertices on the corresponding Voronoi segments and build the flap polygon by joining the s- and v-vertices alternately. Let us label the vertices of su-polygon as $s_1, v_1, s_2, v_2, \ldots, s_i, v_i, \ldots, s_n, v_n$. We analyze three cases, illustrated in Figs. 9 and 10.

Case "Out": s_i lies outside the circle C. Independent on whether the whole point set is convex or not, the extreme su-polygon has the same shape, illustrated in Fig. 9, with exactly two strictly positive source angles. Since the total curvature at the source vertex is positive, the point set supports an sp-star unfolding polygon.

Case "In": s_i lies inside the circle C. The entire point set may be convex or not, depending on whether the special point lies outside or inside the convex hull of the others. In the first case, the Voronoi diagram has only unbounded faces, otherwise the Voronoi region for the special source vertex is bounded (Fig. 10).

In both cases, we define an adapted extreme su-polygon, where all v-vertices are moved to the center of the circle except v_{i-1} and v_i. The lines $s_{i-1}c$ and $s_{i+1}c$ will intersect the Voronoi segments corresponding to v_{i-1} and v_i respectively. We place v_{i-1} and v_i, respectively, on those intersection points, as illustrated in Fig. 10 (where these vertices are shown slightly apart from the extreme position, for clarity). In this configuration, the s-angles at all s-vertices except s_i are zero, and thus the total angle sum at the s-vertices is no more than 2π

(positive curvature). All v-vertices except v_{i-1} and v_i are on the extension of their corresponding Voronoi segments, while v_{i-1} and v_i lie on the Voronoi segments. Now we can now move the v-vertices that are placed on the center of the circle, outwards (away from the center of the circle in the direction towards the Voronoi edge to which they are associated) and the s-angle sum increases. This allows us to obtain a geodesic su-polygon by moving a little so that all s-vertex angles are strictly positive.

This concludes the proof that almost circular point sets support geodesic star-unfoldings. □

Conclusion. We have examined some properties of geodesic star unfoldings that distinguish them from shortest-path star unfoldings. In particular, we have shown that they may not always lead to simple (non-overlapping) unfoldings. We have also discussed specific challenges for reconstructing geodesic star unfoldings on given point sets and presented a method to construct the geodesic star unfolding for "almost circular" points.

Acknowledgement. This research was supported by the NSF grant CCF-1319366 of the second author.

References

1. P. K. Agarwal, B. Aronov, J. O'Rourke, and C. A. Schevon. Star unfolding of a polytope with applications. *SIAM J. Computing*, 26:1689–1713, 1997.
2. M. A. Alam and I. Streinu. Star unfolding polygons. In F. Botana and P. Quaresma, editors, *Lecture Notes in Artificial Intelligence 9201, Automated Deduction in Geometry (ADG'14, Coimbra, Portugal, July 9-11, 2014)*, volume 9201 of *Lecture Notes in Artificial Intelligence*, pages 1–20. Springer Verlag, 2015. ISBN 978-3-319-21362-0.
3. A. D. Alexandrov. *Convex Polyhedra*. Springer Monographs in Mathematics. Springer Verlag, Berlin Heidelberg, 2005. English Translation of Russian edition, Gosudarstv. Izdat.Tekhn.-Teor.Lit., Moscow-Leningrad, 1950.
4. A. D. Alexandrov. *Intrinsic Geometry of Convex Surfaces*. Chapman and Hill/CRC, Taylor and Francis Group, Boca Raton, Florida, 2006. Selected Works II, translated from Russian by S. S. Kutateladze.
5. B. Aronov and J. O'Rourke. Non-overlap of the star unfolding. *Discrete and Computational Geometry*, 8:219–250, 1992.
6. E. D. Demaine and J. O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, and Polyhedra*. Cambridge University Press, 2007.
7. K. Fukuda. Strange unfoldings of convex polytopes. `http://www.ifor.math.ethz.ch/~fukuda/unfold_home/unfold_open.html`, 1997.
8. E. Miller and I. Pak. Metric combinatorics of convex polyhedra: Cut loci and nonoverlapping unfoldings. *Discrete Comput. Geom.*, 39(1):339–388, 2008.
9. D. Mount. On finding shortest path on convex polyhedra. Technical Report 1495, University of Maryland, 1985.
10. W. Schlickenrieder. Nets of polyhedra. Technical report, Technische Universitat Berlin, 1997.
11. M. Sharir and A. Schorr. On shortest path in polyhedral spaces. *SIAM Journal on Computing*, 15(1):193–215, February 1986.

Geometric Deformations of Sodalite Frameworks

Ciprian Borcea[1] and Ileana Streinu[2]

[1] Department of Mathematics, Rider University, Lawrenceville, NJ 08648, USA
[2] Computer Science Department, Smith College, Northampton, MA 01063, USA

Abstract. In mathematical crystallography and computational materials science, it is important to infer flexibility properties of framework materials from their geometric representation. We study combinatorial, geometric and kinematic properties for frameworks modeled on sodalite.

Keywords: periodic framework, sodalite, geometric flexibility.

1 Introduction

In this paper we study the deformation space of a periodic framework modeled after the ideal sodalite.

The sodalite framework is the prototype of numerous crystal structures. In fact, more than 900 crystals of this family have been identified [7]. This structure is remarkable not only for crystallography and materials science. It has direct connections with tiling and sphere packing problems and has natural generalizations in arbitrary dimension [11].

A one-parameter geometric deformation of sodalite was first observed by Pauling [12] in 1930, and used as a phase transition model [10, 13, 5].

We show that, besides this classical 'Pauling tilt scenario', the deformation space of sodalite includes a six-dimensional component, which can be described in fairly intuitive terms.

Our study relies on the mathematical foundations of a deformation theory for periodic frameworks introduced in [2]. In general, the deformation space is a semi-algebraic set made of real solutions of a finite system of polynomial equations. For applications, such as displacive phase transitions in crystalline materials, it is highly desirable to obtain explicit and detailed descriptions of the deformation space [6, 3, 4].

We start with the presentation of the three-dimensional structure of the ideal sodalite framework in a realization with congruent regular tetrahedra and maximal crystallographic symmetry [10, 9, 4]. We express the *periodic graph* underlying the framework via 6-rings of tetrahedra with marked periods. This formulation leads naturally into considerations of symmetry preservation in deformations, with central symmetry at the forefront. The existence of a six-dimensional deformation component follows naturally from this formulation. We also investigate dihedral symmetry and use it to explain the one-dimensional 'classical Pauling tilt'.

2 A placement with maximal symmetry

We use Figure 1 to illustrate the essential aspects of the initial placement of the sodalite framework: a 6-ring of regular tetrahedra (a) is placed in a specific manner relative to a cube of side length 2 centered at the origin (c). Three pairs of parallel vectors between specific vertices of the ring (b) yield the generators of the periodicity lattice.

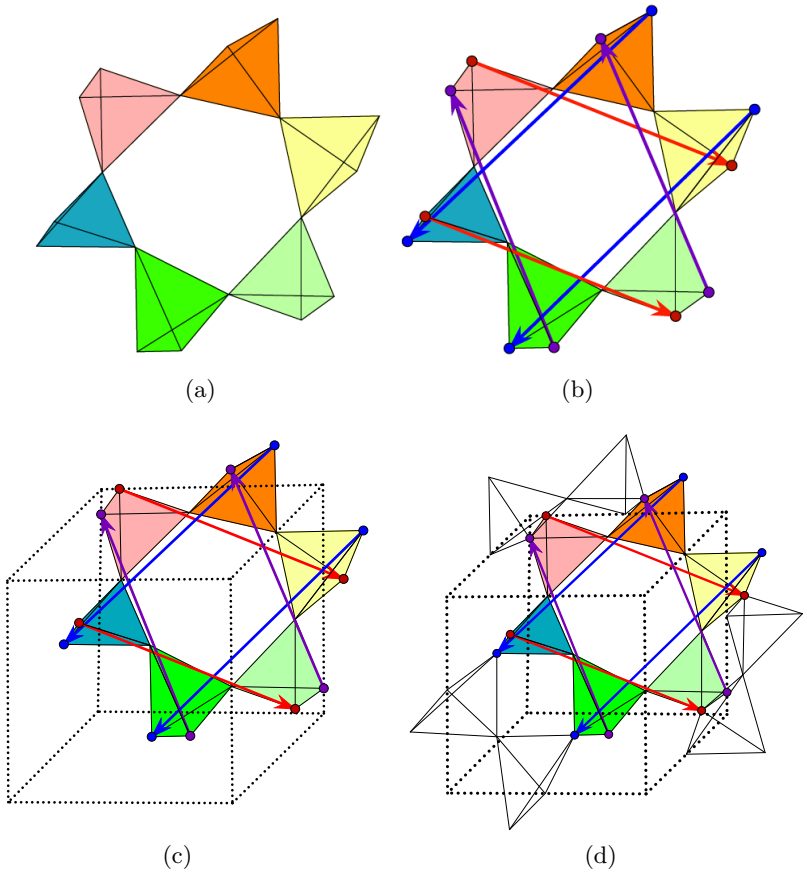


Fig. 1. An ideal sodalite placement with cubical symmetry. A 6-ring of this framework is highlighted, with three pairs of generators of the periodicity lattice. In each pair the vectors must be equal. All tetrahedra are congruent and regular.

Specifically, we consider the origin of the Cartesian system of coordinates at the center of the cube depicted in Fig.1 (c), with standard basis vectors e_1, e_2, e_3 reaching to the centers of the frontal, right side and top face respectively. Thus, the edge length for the cube is 2. The edge length for the regular tetrahedra of the framework is determined by the condition that each tetrahedron has a pair of opposite (and hence orthogonal) edges parallel to a pair of standard directions e_i, e_j. This requires an edge lengths of $2a$, with $a = \sqrt{2} - 1$.

The $4 \times 6 = 24$ tetrahedra around the cube. We give the coordinates for the four vertices $v_0, ..., v_3$ of the tetrahedron touching the cube at $v_0 = e_1 + e_3$. They are:

$$v_0 = (1, 0, 1),\ v_1 = (1, 0, 2\sqrt{2} - 1), \qquad (1)$$
$$v_2 = (\sqrt{2} - 1, \sqrt{2} - 1, \sqrt{2}),\ v_3 = (\sqrt{2} - 1, 1 - \sqrt{2}, \sqrt{2})$$

The coordinates for the vertices of the other tetrahedra are obtained by symmetry. The symmetries of the cube are represented by permutations and sign changes of the three coordinates and form a group of order $3! \cdot 2^3 = 48$. This group has a double-transitive action on the 24 framework tetrahedra around the cube.

Periodicity lattice. The translational symmetries of the whole framework are generated by the three (pairs of equal) vectors shown in Figure 1(b). In coordinates:

$$\lambda_1 = \sqrt{2}(1, -1, -1),\ \lambda_2 = \sqrt{2}(-1, 1, -1),\ \lambda_3 = \sqrt{2}(-1, -1, 1) \qquad (2)$$

with the natural labeling suggested by the order three rotational symmetry around the diagonal direction $e_1 + e_2 + e_3$. We note that the periods

$$\lambda_2 + \lambda_3 = -2\sqrt{2}e_1,\ \lambda_3 + \lambda_1 = -2\sqrt{2}e_2,\ \lambda_1 + \lambda_2 = -2\sqrt{2}e_3 \qquad (3)$$

are *mutually orthogonal* and generate a *sublattice of index 2* in the full lattice of periods generated by (2).

The sodalite cage and the Kelvin polyhedron. The 24 framework tetrahedra wrapped around a cube form a so-called *sodalite cage*. We note that the barycenter of the tetrahedron in (1) has coordinates $b = (\frac{1}{\sqrt{2}}, 0, \sqrt{2})$ and the barycenters of the 6-ring of tetrahedra highlighted in Figure 1 form the vertices of a planar regular hexagon of edge one, centered at $c = \frac{1}{\sqrt{2}}(1, 1, 1)$.

Thus, the 24 barycenters of the sodalite cage tetrahedra give the vertices of a centrally symmetric polyhedron with 8 hexagonal faces and 6 square faces. Under translation by the periodicity lattice Λ generated by (2), this polyhedron *tiles* the three dimensional space R^3, as illustrated in Figure 2. Translations identifying opposite faces are periods and generate the periodicity lattice Λ. The polyhedron is the *Voronoi cell* at the origin for the lattice Λ. It may be seen as a truncated regular octahedron (with edge length 3 at our chosen scale). It is also called the Kelvin polyhedron, due to a famous conjecture formulated by Kelvin

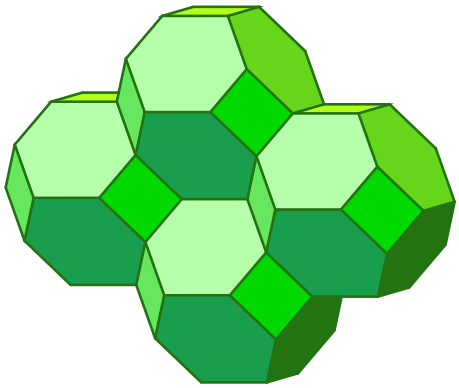


Fig. 2. Kelvin polyhedra tiling space.

[14] and disproved by a counter-example of Weaire and Phelan [15]. The name *permutohedra* is also used for a family which includes this polyhedron.

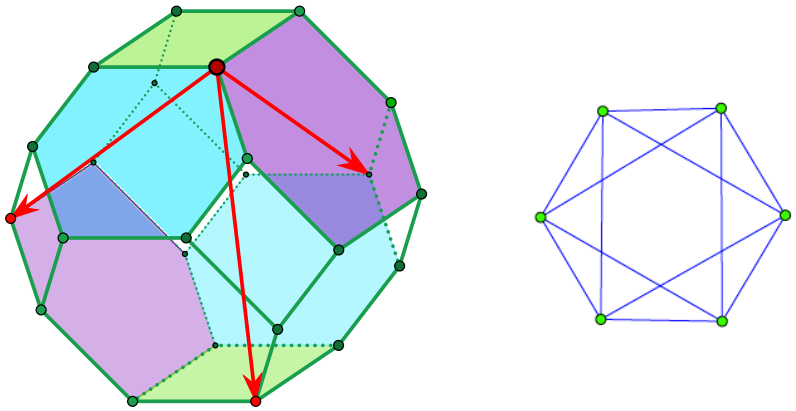


Fig. 3. (Left) Periodicity identifications in the Kelvin polytope. Opposite faces are identified by translation. (Right) The quotient graph of the Kelvin polytope skeleton.

Figure 3 illustrates the result of periodicity identifications in the vertex-and-edge skeleton of a sodalite cage. The quotient graph pattern is also encoded in a 6-ring (with marked periods), as discussed below.

The 6-rings. A 6-ring of tetrahedra, as in Figure 1, contains all the information needed for generating the whole periodic framework. It contains vertex representatives for all the vertex orbits and edge representatives for all the edge orbits under periodicity. The 6 pairs of vertices which have to be further identified by periodicity provide the generators of the periodicity lattice Λ. It is important to note that (up to sign) we have exactly three generators, since the 6 periods, as

free vectors, come as three pairs of equal vectors. This is a consequence of the *central symmetry* of the 6-ring. Indeed, $c = \frac{1}{\sqrt{2}}(1,1,1)$ is actually the center of symmetry for the 6-ring from Figure 1.

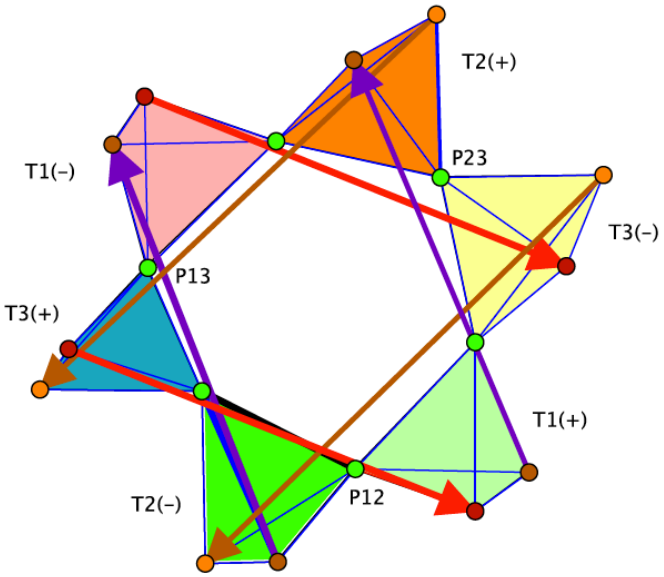


Fig. 4. A 6-ring of the sodalite framework with three pairs of generators of the periodicity lattice. In each pair the vectors must be equal.

In order to have a convenient representation of the symmetries of the cube (given by permutations and sign changes of the coordinates) we **label** the tetrahedra in the 6-ring as follows: $T_1^-, T_3^+, T_2^-, T_1^+, T_3^-, T_2^+$, with T_3^+, T_2^- on the frontal face of the depicted cube. Then coordinate sign changes will correspond to (upper) sign changes and permutations of coordinates will correspond to permutations of lower indices and multiplicative signature effect on upper signs. For example, the transposition of the first two coordinates corresponds to the order two product of transpositions $(T_1^-, T_2^+)(T_3^-, T_3^+)(T_2^-, T_1^+)$. Figure 4 summarizes this description.

We denote $P_{13} = T_1^- \cap T_3^+$, $Q_{23} = T_3^+ \cap T_2^-$, with similar labeling around the **spatial hexagon** $P_{13}Q_{23}P_{12}Q_{13}P_{23}Q_{12}$. Note that, in the initial placement, the triangles $P_{12}P_{23}P_{13}$ and $Q_{12}Q_{23}Q_{13}$ are centrally symmetric with respect to $c = \frac{1}{\sqrt{2}}(1,1,1)$, but are not in the same plane. If we ignore the conditions on the three pairs of period vectors, the 6-ring, as a finite linkage has twelve degrees of freedom: the spatial hexagon has a six-dimensional deformation space and for a fixed configuration of the hexagon, each tetrahedron may rotate around the corresponding edge.

3 Deformations

We use the previously described placement for a single 6-ring to investigate the periodic deformations allowed by the ideal sodalite structure. Specifically, we show below that the deformation space has a six-dimensional component related to the preservation of central symmetry for the 6-ring. This ample geometrical flexibility supports the experimentally observed versatility of sodalite [5].

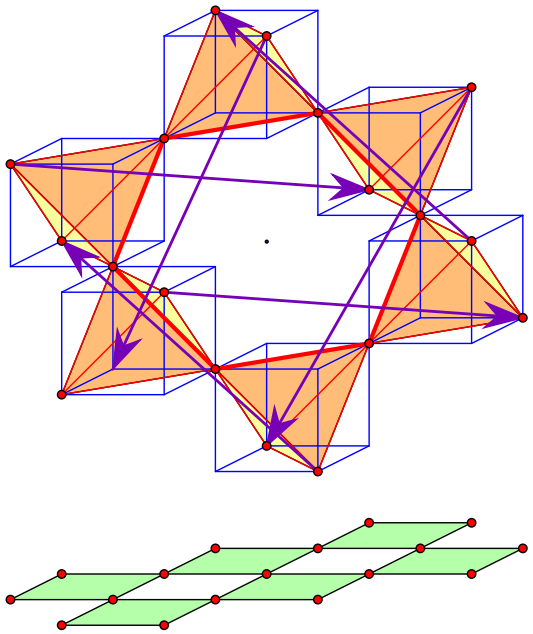


Fig. 5. A 6-ring in a periodic deformation of the ideal sodalite framework, with three pairs of generators of the periodicity lattice. The ring has central symmetry but not periodicity preserving reflections. Cubes and their projections serve only suggestive purposes for spatial positioning.

Deformations preserving central symmetry. This six-dimensional component allows a very direct and intuitive description. Since we want to preserve the central symmetry of the 6-ring, indicating the configuration of three linked tetrahedra, say T_1^-, T_2^+, T_3^-, will be enough. By equivalence under rigid motions we may assume T_2^+ fixed. Then, the positions of T_1^- and T_3^- can be parametrized by $SO(3) \times SO(3)$. The center of symmetry will be at the midpoint of $P_{13}Q_{13}$. The completion of the 6-ring by central symmetry will have *ipso facto* equal vectors in the three pairs of periods. Thus, this component of the deformation space is parametrized by $SO(3) \times SO(3)$ minus the subvariety where the three generators become dependent. Topologically, the rotation group $SO(3)$ is the projective space $P_3(R)$, hence our six-dimensional component is parametrized

by an open and dense subset of $P_3(R) \times P_3(R)$. One configuration is illustrated in Figure 5 and shows that reflection symmetries of the initial 6-ring are lost.

Deformations preserving dihedral symmetry D_3. We now investigate deformations which preserve the symmetries induced by permutations of coordinates in the initial placement. This group of order six may be conceived as a dihedral group D_3 generated by reflections in three planes with a common axis and forming dihedral angles of $\pi/3$ or $2\pi/3$. When restricting our attention to our chosen 6-ring, the group is simply transitive on the six tetrahedra and their barycenters form the vertices of a planar regular hexagon.

Suppose we want to preserve the reflection symmetry given by the transposition of the first two coordinates in the initial placement. The reflection plane goes through P_{12} and Q_{12} and the two marked periods from T_2^- to T_1^- and respectively T_1+ to T_2^+ are parallel to this plane and one is the reflection of the other.

Thus, in order to retain this feature when deforming the 6-ring, we look only at the three linked tetrahedra T_1^-, T_3^+, T_2^- and adopt as reflection plane the plane through $P_{12}Q_{12}$ which runs parallel to the period from T_2^- to T_1^-. When we complete the 6-ring by reflection in this plane, we have to satisfy only one of the remaining periodicity constraints, since the other one will then be fulfilled as well by reflection.

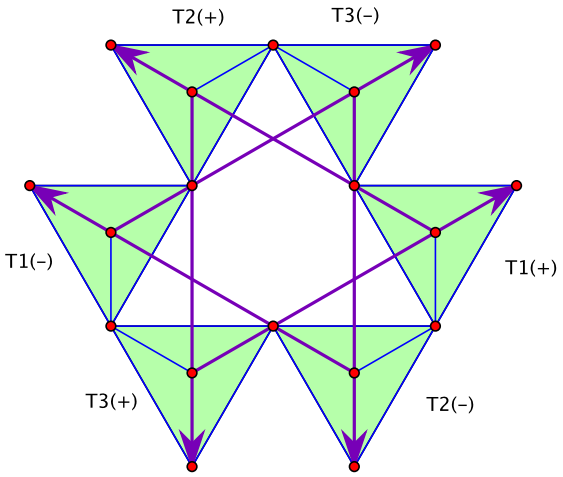


Fig. 6. A tetrahedrite 6-ring resulting from a periodic deformation of the ideal sodalite framework. The regular tetrahedra are pointing towards the viewer. The 6-ring has D_3 dihedral symmetry but no central symmetry.

Similar considerations apply for the other two transpositions of coordinates. In Figure 6 we show a deformation which has preserved all three reflections, but

central symmetry for the 6-ring has been lost. The illustrated structure is that of another mineral called *tetrahedrite*. According to [1], this relationship between sodalite and tetrahedrite was noticed only at a later stage in the mineralogy literature, by A.S. Povarennykh. We now compare deformations which 'break' the central symmetry of the 6-ring and those which maintain it.

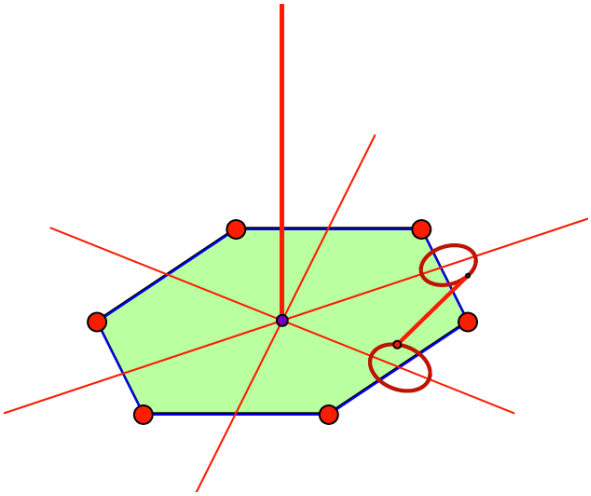


Fig. 7. The regular hexagon of barycenters in a 6-ring with D_3 dihedral symmetry. Circumscribed spheres for linked tetrahedra meet in circles on the reflection planes. A possible edge position is shown.

Given the observed fact that barycenters must form a planar regular hexagon, we may start with the configuration shown in Figure 7. The connecting vertices of the tetrahedra in the 6-ring must be in the reflection planes, indicated by their common axis and intersections with the plane of barycenters. More precisely, they belong to circles of intersection of circumscribed spheres of consecutive tetrahedra. Since one edge determines by symmetry the entire 6-ring, we see that we have a one-parameter family of possible configurations for the 6-ring with given hexagon of barycenters. When we ask for fulfillment of the periodicity conditions, the selection of the edge is subject to an additional condition expressing the parallelism of one of the periods to the corresponding reflection plane.

These considerations show that deformations preserving a dihedral D_3 symmetry consist of curves. One type breaks the central symmetry of the 6-ring: this is the 'Pauling tilt scenario' [12, 13, 10, 8, 5]. As mentioned above, the tetrahedrite structure can be reached via this classical tilt. What is distinctive in this scenario is that sodalite cages maintain their shape, although decreasing in size relative to the initial case.

However, central symmetry may be maintained as follows. Since we have now double transitivity on the six tetrahedra, all diameters in the hexagon of barycenters must go through midpoints of opposite edges in the corresponding tetrahedra. This condition is satisfied when we choose the edge discussed above to be met perpendicularly and in the middle by the appropriate diameter. A deformation result of this procedure is illustrated in Figure 8.

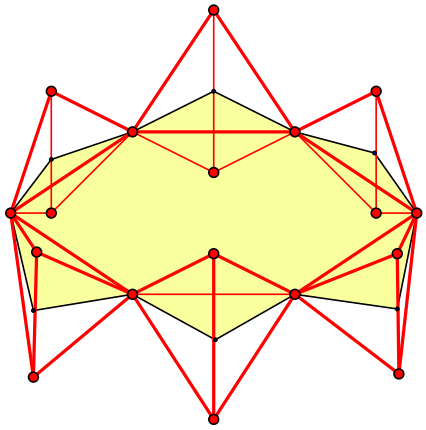


Fig. 8. A deformation result when preserving both D_3 and central symmetry. The highlighted plane is the perpendicular bisecting plane of all distant edges.

4 Conclusion

The sodalite framework belongs to the class of tectosimplicial periodic structures made of vertex sharing simplices. In dimension three, a basic count of infinitesimal deformations yields at least three infinitesimal degrees of freedom for this type of frameworks [2], Theorem 4.2. However, if more than three infinitesimal degrees of freedom are present, as is the case for the ideal sodalite framework, infinitesimal considerations are not sufficient for obtaining actual local deformations. In this paper, we have relied on direct and intuitive geometric features to show that for ideal sodalite, in addition to the one-parameter deformation known as the 'Pauling tilt', there is a six-dimensional deformation component. It is conceivable that certain crystalline materials with distorted sodalite cages may be more closely related to sodalite on this account.

Acknowledgements. The first author acknowledges partial support through NSF award no. 1319389 and the second author acknowledges partial support through NSF award no. 1319366. Both authors are partially supported through NIH Grant 1R01GM109456. All statements, findings or conclusions contained in this paper are those of the authors and do not necessarily reflect the position or policy of the US Government. No official endorsement should be inferred.

References

1. Baur, W.H. and Fischer, R.X.: *A historical note on the sodalite framework: The contribution of Frans Maurits Jaeger*, Microporous and Mesoporous Materials **116** (2008), 1-3.
2. Borcea, C.S. and Streinu, I.: *Periodic frameworks and flexibility*, Proc. Roy. Soc. A **466** (2010), 2633-2649.
3. Borcea, C.S. and Streinu, I.: "Deformations of crystal frameworks", arXiv:1110.4661 (2011).
4. Borcea, C.S. and Streinu,I.: "Frameworks with crystallographic symmetry", Philosophical Transactions of the Royal Society A (2014) **372**, 20120143.
5. Depmeier, W.: *The sodalite family - a simple but versatile framework structure*, Reviews in Mineralogy and Geochemistry **57** (2005), 203-240.
6. Dove, M.T.: *Theory of displacive phase transitions in minerals*, American Mineralogist **82** (1997), 213-244.
7. Fischer, R.X. and Baur, W.H.: *Symmetry relationships of sodalite (SOD)-type crystal structures*, Z. Kristallogr. **224** (2009), 185-197.
8. Hassan, I. and Grundy, H.D.: *The crystal structures of sodalite-group minerals*, Acta Crys. **B40** (1984), 6-13.
9. Kotani, M. and Sunada, T.: *Standard realizations of crystal lattices via harmonic maps*, Trans. Amer. Math.Soc. **353** (2000), 1-20.
10. Megaw, H.D.: Crystal Structures: A working Approach, W.B. Saunders Company, Philadelphia, (1973).
11. O'Keeffe, M.: *N-Dimensional Diamond, Sodalite and Rare Sphere Packings*, Acta Cryst. (1991), A47, 748-753.
12. Pauling, L.: *The structure of sodalite and helvite*, Z. Kristallogr. **74**(1930), 213-225.
13. Taylor, D.: *The thermal expansion behaviour of the framework silicates*, Mineralogical Mag. **38** (1972) , 593-604.
14. Thompson, William (Lord Kelvin): *On the Division of Space with Minimum Partitional Area*, Philosophical Magazine **24** (1887), 503, doi:10.1080/14786448708628135
15. Weaire, D.; Phelan, R.: *A counter-example to Kelvin's conjecture on minimal surfaces*, Phil. Mag. Lett. **69** (1994), 107110, doi:10.1080/09500839408241577

Computing the Straight Skeleton*

Extended Abstract

John C. Bowers

Department of Computer Science, James Madison University, Harrisonburg, VA, USA.
jbowers@cs.umass.edu

Abstract. The straight skeleton of a polygon is a certain tree structure on the polygon's interior made up of straight line segments. It serves as a straight-line analog of the medial axis of a polygon. It is an example of an angular bisector network because each of its line segments lies on an angle bisector of a pair of the polygon's edges. It has many applications, including applications in origami design, computer aided design (CAD), and graphics.
Though computing the medial axis of a polygon can be done in linear time, fast algorithms for computing the straight skeleton have not yet been found. In this paper we identify several elementary combinatorial and geometry properties of the straight skeleton that allow us to compute it efficiently from a secondary structure called the motorcycle graph of the polygon.

1 Introduction

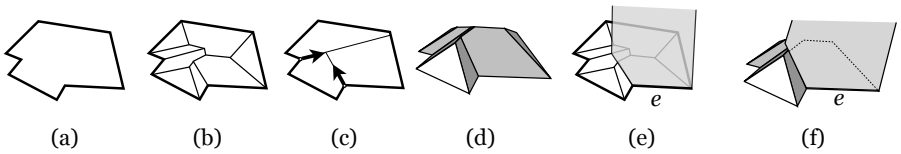


Fig. 1. (a) A polygon, (b) its straight skeleton, (c) its induced motorcycle graph (d) the straight skeleton roof, (e) a slab shown from above, (f) the slab shown in perspective.

An *angular bisector network* is a tree structure on the interior of a polygon comprised of line segments each of which lie on an angle bisector of the lines supporting two of the polygon's edges. (The edges need not be consecutive in the polygon.) The straight skeleton is one particular angular bisector network that has found many applications. It is used, for instance, in computer aided design to construct polygonal roofs on which water does not pool [1]; in graphics to generate mitered offset polygons; and more interestingly, in origami design.

* A version of this work with all details and proofs has been submitted to *Computational Geometry: Theory and Applications.*

In origami design, the straight skeleton is particularly interesting as a special case of Lang's universal molecule algorithm [8]. In [3] it was shown that most of the origami patterns created by the universal molecule algorithm are not foldable as panel-and-hinge structures; however, in the special case that the universal molecule is a straight skeleton, the pattern is always foldable [6].

The simplest way to define the straight skeleton is via a wavefront process in which the sides of the polygon are moved inwards in parallel, each at the same speed. The vertices in this motion move along the angle bisectors of their adjacent edges. If an edge collapses to zero-length it is contracted in the polygon, and if a vertex "hits" another side of the wavefront, the wavefront is split at the hit point into two polygons, and continues independently in each. The straight skeleton is defined as the trace of the vertices throughout this wavefront motion. Figure 1b depicts the straight skeleton of a polygon.

Computing the straight skeleton has proved challenging. This is particularly intriguing because of its similarity to the medial axis, which can be computed in linear time. Currently, the fastest algorithms for computing the straight skeleton require first computing a secondary structure called the *motorcycle graph* of the polygon. The motorcycle graph is defined by placing a "motorcycle" at each reflex vertex in the polygon. The motorcycle moves inwards along the vertex's angle bisector at a speed equal to speed of the vertex in the wavefront. As it moves it lays down a "track" behind it. When a motorcycle encounters either an edge of the polygon or another motorcycle's track it crashes, but its track persists. The motorcycle graph is the set of tracks left over after all motorcycles have crashed. A motorcycle graph is illustrated in fig. 1c. The fastest algorithm is that of [4], which requires that the motorcycle graph be provided as input. That algorithm takes $O(n \log r \log n)$ time given the motorcycle graph (where n is the number of vertices and r is the number of reflex vertices in the polygon). Currently, the fastest algorithm for computing the motorcycle graph is that of [9], which takes $O(r^{4/3+\epsilon})$ time (in non-degenerate cases) leading to a full straight skeleton computation in $O(n \log r \log n + r^{4/3+\epsilon})$ time.

A second way of defining the straight skeleton, which has proved useful for computation, is as the *straight skeleton roof*. This is depicted in fig. 1d. It is a lifting of the straight skeleton into $\mathbf{R}^3$, such that each vertex is lifted to the z-coordinate given by the time at which the vertex appeared in the wavefront. Each region of the straight skeleton becomes a face of the roof. It was shown in [9] that this roof is equivalent to the lower envelope of a set of partially infinite strips in $\mathbf{R}^3$ called *slabs* (see fig. 1(e, f)). Note that the slabs are defined with respect to the motorcycle graph.

Using the straight skeleton roof, a simple divide and conquer algorithm has been described that computes the straight skeleton of a *monotone polygon* in $O(n \log n)$ time [2]. (Recall that a *monotone polygon* is one in which there is a direction such that all lines parallel to that direction intersect the polygon in at most two points.) The algorithm makes use of several special properties of the straight skeleton of a monotone chain to compute two terrains. Computing the intersection between the two terrains allows the computation of the straight skele-

ton roof. Generalizing this to non-monotone polygons is not straight-forward, because the straight skeleton of a non-monotone chain is not a terrain, and dealing with the resulting objects is more challenging.

In this paper we identify several elementary combinatorial and geometric properties of the straight skeleton that allow us to generalize the straight skeleton roof to subchains of the polygon. We call our generalization a *partial roof* for the subchain. One key property of our generalization is that two partial roofs for adjacent subchains of the polygon can be merged in linear time by walking along their intersection in a manner similar to [2]. The second key property of our partial roof is that there is a unique object that satisfies our definition of a partial roof for the entire polygon, namely the straight skeleton roof itself. These two properties gives us a straightforward divide and conquer algorithm for computing the straight skeleton of a polygon from its slab set in $O(n \log n)$ time. First divide the polygon into two, then recursively compute a partial roof for each of the subchains, finally merge the two partial roofs to compute the partial roof for the entire polygon, which is the straight skeleton roof. In the remainder of this paper, we will present an overview of the main properties that make our approach work, and sketch the algorithm for merging partial roofs.

2 Straight skeleton properties

Known properties The following are several known properties of the straight skeleton of a polygon:

1. The straight skeleton is a tree on the interior of the polygon and the leaves of the tree are the polygon vertices.
2. It divides the interior of the polygon into a set of faces, each of which is incident along one entire edge of the polygon. We call the polygon edge incident to a face of the straight skeleton its ***base edge***.
3. Each face is monotone with respect to its base edge, meaning that any line orthogonal to the base edge intersects the face in at most two points.
4. Two faces of the straight skeleton are incident along at most one edge.

Critical edges Our goal is to generalize the straight skeleton roof to subchains of the polygon and to provide a method of merging partial roofs for adjacent subchains. Here we identify a relationship between straight skeleton edges and subchains of a polygon. Suppose C is a subchain of a polygon and e is a straight skeleton edge. We say that e is ***critical for*** C if both of the faces incident along e have their base edge on C. The reason for identifying these critical edges, is that they play a key role in our generalization of the straight skeleton roof to the subchain C.

Now suppose C is split into two subchains C_1 and C_2. Each critical edge e for C can be classified into one of three sets: those where the two base edges of the faces incident to e are in C_1, those where the two base edges of the faces incident to e are in C_2, and those where one of the base edges is in C_1 and one is in C_2. In the first case, the edge e is critical for C_1. In the second, e is critical for

C_2. Interestingly, in the third case e is neither critical for C_1 or C_2, but *becomes critical* when C_1 and C_2 are combined to create C. The key property of this third set is summarized in the following lemma:

Lemma 1. *Let E be the set of straight skeleton edges that are critical for C but not for C_1 or C_2. E forms a path in the straight skeleton starting at the vertex v common to C_1 and C_2.*

This lemma follows from the fact that the straight skeleton is a tree and each of its faces are incident along an edge of the polygon P. It is relatively simple, but is the main reason our merge operation works. The fact that the critical edges form a path allows us to describe a simple walk procedure that recovers this path. We call the set of edges that become critical when C_1 and C_2 are combined to form C the ***critical path*** for C_1 and C_2. Figure 2 depicts the straight skeleton of a polygon, two subchains C_1 and C_2, and the edges that are critical for C_1 (red), for C_2 (blue), and the critical path (gray).

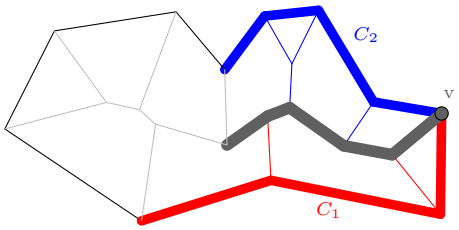


Fig. 2. A straight skeleton for a polygon and the edges that are critical for three subchains: C_1 (red), C_2 (blue), and C_1 and C_2 combined at v (thick gray).

Slab set for a polygon Recall from the introduction that an alternative definition of the straight skeleton is as a lower envelope of a set of partially infinite strips in $\mathbf{R}^3$ called *slabs*. Following Huber and Held [7] we define a single infinite strip in $\mathbf{R}^3$ called a "slab" for each edge e of the polygon. The slab lies in the plane Π_e through e that makes an angle of $\pi/4$ with the interior of P. If any of the vertices of e is reflex, then the vertex has an edge in the motorcycle graph, which we call its ***motorcycle arm***. An edge e may have zero, one, or two motorcycle arms. We will orient the polygon P counter-clockwise so that each edge has a well defined ***left*** and ***right*** vertex, and call the motorcycle arms its ***left*** and ***right*** motorcycle arms. For each edge e lift its motorcycle arms upwards onto Π_e.

We establish a local coordinate system for Π_e so that the positive x-direction points counter-clockwise along e in P and the y-direction points upward along the slope vector of Π_e. The ***slab*** s for e is defined as the region of Π_e above

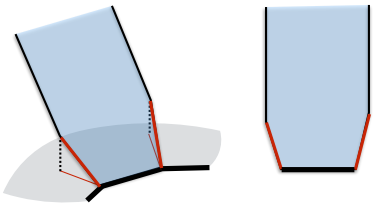


Fig. 3. *Left:* a slab defined by lifting the motorcycle arms (red) of its base edge up onto the plane through the base edge making an angle of $\pi/4$ with the interior of the polygon. *Right:* the *local view* of the slab.

the edge and lifted motorcycle arm chains for e. We call e the ***base edge*** of the slab, and the lifted motorcycle arms the ***motorcycle arms*** of the slab. See fig. 3. Recall that each motorcycle graph edge is a left arm of one polygon edge and the right arm of another. Thus a motorcycle edge in one slab always has a corresponding motorcycle edge in the other slab.

The ***slab set*** for P, denoted S_P is the set of slabs defined for each edge of P. For a subchain C of the polygon, the slab set $S_C \subseteq S_P$ for C is the set of slabs in S_P whose base edge is an edge of C.

3 Partial roofs

Local definition of the straight skeleton roof To define partial roofs, we first need the following alternative definition for the straight skeleton roof from [5]. Let s be the slab for an edge e of the polygon. Intersect s with all other slabs to obtain a set of line segments $\mathcal{L}_P(s)$ on s. The lower envelope of $\mathcal{L}_P(s)$ in s is the face of the straight skeleton supported by s. The *lower envelope* of a set of line segments, is the set of points that do not lie above any of the line segments.

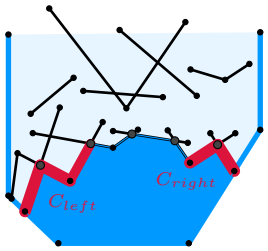


Fig. 4. The set of line segments $\mathcal{L}_P(s)$ on a slab s. The two red chains are C_{left} and C_{right} illustrate the critical chains for some subchain C of the polygon. The dark shaded region is the lower envelope of the line segments, which is the straight skeleton face for this slab.

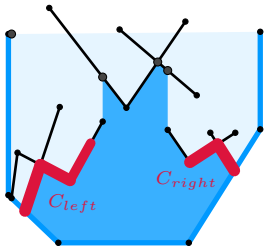


Fig. 5. The restriction of the line segments from fig. 4 to only those in $\mathcal{L}_C(s)$ for some subchain C. Notice that the two critical chains C_{left} and C_{right} are still supported by the segments and lie on the lower envelope of $\mathcal{L}_C(s)$.

Restricting the slab set to a subchain Our working definition of the straight skeleton roof uses the entire slab set for the polygon P. To generalize the roof to a subchain C of P, we restrict our attention to the slab set S_C for C. Let $\mathcal{L}_C(s)$ denote the set of line segments given by intersecting a slab $s \in S_C$ with all other slabs in S_C. Notice that since we are simply restricting which slabs we intersect, $\mathcal{L}_C(s) \subseteq \mathcal{L}_P(s)$. Furthermore, notice that if e is a critical edge for the straight skeleton, then both of its faces (and hence both of its supporting slabs) have their base edge in S_C. Thus e must be supported by one of the line segments in $\mathcal{L}_C(s)$. One further property that can be derived is that the set of critical edges on s form (at most) two polygonal chains on arrangement of line segments $\mathcal{L}_C(s)$, and these two chains are incident to the left and right motorcycle arms of s. For this reason, we call these the ***critical chains*** C_{left} and C_{right} of s.

Partial roof We defined the partial roof for a subchain C locally as we did with the straight skeleton above. For each slab s in S_C, we define a *partial face* for s is given by any two polygonal chains on the arrangement $\mathcal{L}_C(s)$ that contain the critical chains for s with respect to C. We call these the ***defining chains*** for the face. The lower envelope of these two chains gives us the a (possibly partially infinite) polygon on the surface of s. Suppose we now have a partial face for each slab in S_C. Every defining chain edge of a partial face for a slab s lies on the intersection of s with some other slab s' in S_C. If the partial face for s' also has an edge on the intersection of s and s' and the two edges are equal in $\mathbf{R}^3$, we say that the partial faces for s and s' are compatible and we glue the two faces together by identifying the two corresponding edges. A ***partial roof*** is given by the following construction. Start with a set of partial faces, one for each slab in S_C such that all pairs of partial faces are compatible. Glue every pair of partial faces along their corresponding edges to form a single surface. If that surface is topologically a disk, we call it a partial roof for C. The main property of partial roofs is that the partial roof for the entire polygon is the straight skeleton roof.

4 Merging partial roofs

A key property of a partial face is that the partial face for any given slab s contains the corresponding straight skeleton roof face. Thus if two straight skeleton roof faces are incident along an edge, any two corresponding partial faces intersect along the edge. We now want to merge two partial roofs R_1 and R_2 for C_1 and C_2. Recall that the critical edges for C_1 and C_2 form a path in the straight skeleton. Since each partial roof face contains its straight skeleton face, this path is on the intersection of R_1 and R_2. Walking along this intersection allows us to find the critical path. Once we have the critical path, we cut the two roofs R_1 and R_2 along this path, discard the part of each partial face above the path, and merge R_1 and R_2 by gluing them together along the path. We are glossing over the details here, but the punchline is that our method can be executed in linear time.

5 Conclusion

Given our merge operation, the following strategy computes a partial roof: divide the polygon into two, recursively compute a partial roof for each, and then merge the two into a partial roof for the entire polygon. Since the unique partial roof for an entire polygon is the straight skeleton roof, this gives us a method for computing the straight skeleton of a polygon. The algorithm has the same recurrence as merge sort, and thus requires $O(n \log n)$ time where n is the number of vertices in the polygon (given the slab set as input). Since computing the slab set requires computing the motorcycle graph, this leads to an algorithm for computing the straight skeleton in $O(n \log n + r^{4/3+\epsilon})$ time where r is the number of reflex vertices in the polygon.

References

1. O. Aichholzer, D. Alberts, F. Aurenhammer, and B. Gärtner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.
2. T. Biedl, M. Held, S. Huber, D. Kaaser, and P. Palfrader. Straight skeletons of monotone polygons. In *30th (EuroCG '14)*, Dead Sea, Israel, Mar. 2014.
3. J. C. Bowers and I. Streinu. Rigidity of origami universal molecules. In T. Ida and J. D. Fleuriot, editors, *Automated Deduction in Geometry*, volume 7993 of *Lecture Notes in Computer Science*, pages 120–142. Springer, 2012.
4. S.-W. Cheng, L. Mencel, and A. Vigneron. A faster algorithm for computing straight skeletons. In *Proc. 22nd Euro. Symp. on Algorithms (ESA 2014)*, Worclaw, Poland, Sept. 2014.
5. S. W. Cheng and A. Vigneron. Motorcycle graphs and straight skeletons. In *Proc. 13th Symp. on Discrete algorithms*, SODA '02, pages 156–165, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.
6. E. D. Demaine and M. L. Demaine. Computing extreme origami bases. Technical Report CS-97-22, Department of Computer Science, University of Waterloo, May 1997.

7. S. Huber and M. Held. Theoretical and practical results on straight skeletons of planar straight-line graphs. In *Proc. 27th Symp. on Computational geometry*, SoCG '11, pages 171–178, New York, NY, USA, 2011. ACM.
8. R. J. Lang. A computational algorithm for origami design. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pages 98–105, 1996.
9. A. Vigneron and L. Yan. A faster algorithm for computing motorcycle graphs. In *Proc. 29th Symp. on Computational geometry*, SoCG '13, pages 17–26, New York, NY, USA, 2013. ACM.

An Equivalence Proof Between Rank Theory and Incidence Projective Geometry

David Braun, Nicolas Magaud, and Pascal Schreck

ICube, UMR 7357 CNRS, University of Strasbourg
Pôle API, Bd Sébastien Brant, BP 10413, 67412 Illkirch, France
{david.braun, magaud, schreck}@unistra.fr

Abstract. Incidence geometry is a well-established theory which captures the very basic properties of all geometries in terms of belonging of points to lines, planes, etc. Moreover, considering incidence projective geometry leads to a very simple framework. This article considers two different approaches to formalize this theory in Coq. The first one consists in the usual geometric axiom system encountered in the literature. The second one relies on combinatorial aspects through the notion of rank which is based on the matroid structure of incidence geometry. In this paper, we prove in Coq the equivalence between these two approaches in both 2D and 3D. This result allows to study further automation of many proofs of projective geometry theorems. We already present some automation techniques in the proof of equivalence.

Keywords: automation, Coq, formalization, incidence, matroid, projective geometry, ranks

1 Introduction

In this article, we consider two axiom systems of incidence projective geometry. The equivalence proof between these two formalizations through the proof assistant Coq is our main original contribution.

The most elementary geometry that can be studied is the incidence geometry. It basically analyzes the incidence relation between two types of objects: points and lines. This geometry is ubiquitous, collinearity issues are found in various contexts. For example, it appears in graph theory, subways maps and even children games such as dobble. Moreover, this simple geometry can be defined by a fairly simple set of axioms both in the plane and in an at least three dimensional setting (noted $\geq$3D) [3, 6]. So it is conceivable to efficiently automate the reasoning.

We describe two geometrical axiomatizations. The first one is based on a classical geometrical characterization. The second one relies on the combinatorial notion of rank provided by the matroid structure of incidence geometry. The latter allows to represent homogeneously all incidences of this geometry (point-point, point-line, point-plan, line-line, line-plan, plan-plan) but also relations of

collinearity, coplanarity, etc. Thanks to the unique representation of all incidence relations as ranks, we can thus consider a more systematic automation.

Our main motivation is the mechanization of demonstrations to ease the task of mathematicians but also to more effortlessly study the degenerate cases. Formalization and proof of geometrical theorems are not an easy task. It needs to be very rigorous to take into account all non degeneracy conditions that can occur in this context. These conditions often lead to long and tedious technical proofs. That is why, many papers have emphasized the importance of investigating the problem of degenerate cases in formal geometry [7, 21, 28].

In this paper, we work in the specific context of projective geometry in which the notion of parallel does not exist. This simple framework is powerful enough to study the formalization and proof of complex problems as suggested in [23]. In addition, there is no loss of generality since it is possible to switch from projective geometry to affine geometry putting a chosen line at infinity. Moreover, it is possible to consider finite geometries containing only finite sets of points and lines that check the properties of projective framework. Examples of finite geometry such as Fano plane are often used to analyze the behavior of geometric theorems.

Some ideas behind this work have their origins in previous studies about combinatoric aspects of proofs in incidence geometry as described in [18, 24], and particularly a proof of Desargues theorem using only ranks [19,20]. To carry out the latter, the authors have proven only one direction of the equivalence theorem between the two theories: the one from a classical incidence projective geometry axiomatization to a rank-based axiom system in 3D and not the converse which was useless at the time. To our knowledge, mathematicians have not studied extensively the details of the equivalence proof between these two approaches.

Related Work Geometry is a good candidate to be studied through the proof assistants like Coq. Many formalizations of plane geometry, Hilbert and Tarski axiomatization has already be done [8,11,21,28]. In the narrower context of projective geometry, there is work in constructive projective geometry initiated by Von Plato [14,33]. Our formalization differs by considering a decidable incidence relation. In the field of automatic methods in projective geometry, we can mention the algebraic formalization of Grassmann-Cayley algebra [9,17,30]. Finally, to our knowledge there is no work treating incidence geometry in a projective framework using the equivalence between a geometrical characterization and a combinatorial approach.

Outline of the paper The paper is organized as follows. In Section 2, we present the axioms for incidence projective geometry and we give an preview of our Coq formalization. In Section 3, we expose our axiomatizations based on the concept of rank from matroid theory. Section 4 investigates in details the equivalence proof between these two formalizations. Finally, the conclusion and perspectives are discussed in Section 5.

Notation We use the naming convention $AXYN$ for our axioms. The letter stands A for axiom, X for the axiom number, Y may take two values (P = projective, R = rank) and N designs the dimension.

2 Axiom Systems of Projective Geometry

In a general setting, geometry such as Euclidean geometry is a complicated subject mixing objects like points, lines and planes with concepts of distance, angle, continuity, incidence etc. Among all these notions, we keep only two kinds of objects, points and lines, and the incidence relation between them to form the elementary incidence geometry.

2.1 Incidence Geometry

Incidence geometry is the study of a triple (Ω,Δ,Φ) called incidence structure. Ω refers to the set of points, Δ is the disjoint set composed of lines and Φ is the binary relation that unite them. Elements which are mutually incident constitute a subset $\Phi \subseteq \Omega \times \Delta$. Intuitively, a point and a line are within this subset if and only if the point is on the line. It is possible to describe this geometry informally with three axioms.

- There is always a line passing through two points.
- On any line, there are at least two points.
- There exist three points that are not aligned.

This characterization is sufficient to prove many theorems [3]. In addition, the enrichment of incidence geometry by new concepts keeps many fundamental results. Projective geometry relies on the basics of incidence geometry. It is a general setting in the hierarchy of geometries which assumes that two lines in a plane always meet [6]. Planes can be defined within our theory but they are not a primitive object. With the concepts of incidence geometry, we can describe properly projective geometry in 2-dimensional space and in at least 3-dimensional space.

2.2 Axioms System for Projective Plane Geometry

The axiom system for projective plane geometry consists of five axioms presented in Coxeter's book [6]. The first two axioms deal with construction of points and lines. We do not specify that the involved points (resp. lines) should be different in axiom *Line-Existence* (resp. *Point-Existence*). Indeed if these points (resp. lines) are equal, the line (resp. point) still exists. In fact, there exists an infinity of lines (resp. points). This formalization choice follows this general principle in formal geometry: it is decisive to define statements in the most general way possible. The next axiom (A3P2) concerns uniqueness of the two defined objects. Finally, axiom (A4P2) states that each line contains at least three points; and

axiom (A5P2) expresses that there always exists two distinct lines, which means dimension is at least 2. Together with axiom *Point-Existence* which show that the dimension is at most 2. Therefore the dimension is exactly 2. The formalization of this axiom system is straightforward, textbooks often use some variants of this system [6] but we choose the latter to ease mechanization of proofs.

(A1P2) Line-Existence : $\forall$ A B : Point, $\exists$ l : Line, A $\in$ l $\wedge$ B $\in$ l

(A2P2) Point-Existence : $\forall$ l m : Line, $\exists$ A : Point, A $\in$ l $\wedge$ A $\in$ m

(A3P2) Uniqueness : $\forall$ A B : Point, $\forall$ l m : Line, A $\in$ l $\wedge$ B $\in$ l $\wedge$ A $\in$ m $\wedge$ B $\in$ m $\Rightarrow$ A = B $\vee$ l = m

(A4P2) Three-Points : $\forall$ l : Line, $\exists$ A B C : Point,
A $\neq$ B $\wedge$ B $\neq$ C $\wedge$ A $\neq$ C $\wedge$ A $\in$ l $\wedge$ B $\in$ l $\wedge$ C $\in$ l

(A5P2) Lower-Dimension : $\exists$ l m: Line, $l \neq m$

2.3 Axioms System for Projective Space Geometry

In the same way, we define an axiom system for projective space geometry. The system still contains five axioms with three of them remaining unchanged (A1P3, A3P3, A4P3). *Pasch*'s axiom substitutes (A5P2) assuming that two coplanar lines always meet. Furthermore, we modify the axiom *Lower-Dimension* to capture a projective geometry in at least 3-dimensional space. For this, we assume that there exist two lines which do not meet. This time, the second axiom does not allow to limit the higher dimension, that is why we consider a geometry $\geq$3D.

(A1P3) Line-Existence : $\forall$ A B : Point, $\exists$ l : Line, A $\in$ l $\wedge$ B $\in$ l

(A2P3) Pasch : $\forall$ A B C D : Point, $\forall$ l_{AB} l_{CD} l_{AC} l_{BD} : Line,
A $\neq$ B $\wedge$ A $\neq$ C $\wedge$ A $\neq$ D $\wedge$ B $\neq$ C $\wedge$ B $\neq$ D $\wedge$ C $\neq$ D $\wedge$
A $\in$ l_{AB} $\wedge$ B $\in$ l_{AB} $\wedge$ C $\in$ l_{CD} $\wedge$ D $\in$ l_{CD} $\wedge$
A $\in$ l_{AC} $\wedge$ C $\in$ l_{AC} $\wedge$ B $\in$ l_{BD} $\wedge$ D $\in$ l_{BD} $\wedge$
($\exists$ I : Point, I $\in$ l_{AB} $\wedge$ I $\in$ l_{CD}) $\Rightarrow$
($\exists$ J : Point, J $\in$ l_{AC} $\wedge$ J $\in$ l_{BD})

(A3P3) Uniqueness : $\forall$ A B : Point, $\forall$ l m : Line,
A $\in$ l $\wedge$ B $\in$ l $\wedge$ A $\in$ m $\wedge$ B $\in$ m $\Rightarrow$ A = B $\vee$ l = m

(A4P3) Three-Points : $\forall$ l : Line, $\exists$ A B C : Point,
A $\neq$ B $\wedge$ B $\neq$ C $\wedge$ A $\neq$ C $\wedge$ A $\in$ l $\wedge$ B $\in$ l $\wedge$ C $\in$ l

(A5P3) Lower-Dimension : $\exists$ l m : Line, $\forall$ p : Point, p $\notin$ l $\vee$ p $\notin$ m

2.4 Formalization in Coq

Implementing these two systems of axioms is rather immediate in the Coq proof assistant; we present below our formalization of the projective space. The notation $\in$ denotes the predicate `Incid`. To enhance modularity, we take advantage of the *type classes* and functors of Coq. The most difficult notation to handle for the reader probably is the curly-brackets notation for constructive existential quantification over the sort Type. For example, the formula `forall l:Line, {P:Point | ∼ Incid P l}` states that `∀ l:Line, ∃ P:Point, ¬ Incid P l`.

To observe equivalences between different variants of our axiomatizations, we split our axioms in different classes considering the dimension. Moreover, this decomposition allows us to bring out the minimal axiom system required to demonstrate a set of properties. First, we build a type class with common axioms between the dimensions namely *Line-Existence*, *Uniqueness* and *Three-Points*. Then, we construct two classes, the first one captures the whole projective plane, and the second one describes the projective space. For the sake of clarity, we present here a simplified version of our implementation.

```
Class ProjectiveSpaceOrHigher `(PPS : PreProjectiveSpace) := {

(* Line-Existence *)
A1P3 : forall (A B : Point) , {l : Line | Incid A l /\ Incid B l};

(* Pasch *)
A2P3: forall A B C D : Point, forall lAB lCD lAC lBD : Line,
  ~ A [==] B /\ ~ A [==] C /\ ~ A [==] D /\
  ~ B [==] C /\ ~ B [==] D /\ ~ C [==] D ->
  Incid A lAB /\ Incid B lAB ->
  Incid C lCD /\ Incid D lCD ->
  Incid A lAC /\ Incid C lAC ->
  Incid B lBD /\ Incid D lBD ->
  (exists I : Point, (Incid I lAB /\ Incid I lCD)) ->
  exists J : Point, (Incid J lAC /\ Incid J lBD);

(* Uniqueness *)
A3P3: forall (A B : Point)(l1 l2 : Line), Incid A l1 -> Incid B l1 ->
  Incid A l2 -> Incid B l2 -> A [==] B \/ l1 = l2;

(* Three-Points *)
A4P3: forall l : Line, {A : Point & {B : Point & {C : Point |
  (~ A [==] B /\ ~ A [==] C /\ ~ B [==] C /\
  Incid A l /\ Incid B l /\ Incid C l)}}};

(* Lower-Dimension *)
A5P3: {l1 : Line & {l2 : Line | forall p : Point,
  ~(Incid p l1 /\ Incid p l2)}}

}.
```

The main difference between our formalization and the axiom system shown above comes from decidability issues and equality relations. The equality on points, denoted [==], is a parameter of our theory. This equality allows to make transparent the way in which the point is built. As the underlying logic of the Coq system is intuitionist, we have to declare explicitly which predicates are decidable. Assuming only the decidability of the predicate `Incid`, we prove the decidability of all other predicates used in this theory (points, lines) [18]. We choose to not use a parametric equality for lines since it can be represented as a set of points in the other axiomatization. It is quite possible to use one, but it has not been necessary.

3 A Rank-based Axiom System

In Section 2, we present a standard axiomatization for projective geometry as a reference. We propose here an alternative axiom system based on the notion of rank. This new approach takes homogeneously the flats into account and we hope that it will simplify the proof schemes in higher dimensions.

3.1 Matroid Properties

The concept of rank comes from matroid theory [29]. Matroids were introduced by Whitney in 1935 to capture abstractly the essence of dependence. Whitney's definition embraces a surprising diversity of combinatorial structures. Matroid allows to capture and generalize the main set properties of linear dependence in vector spaces. When combined with a finite set of points, it catches incidence (collinearity, coplanarity, ...) between these points. However matroids apply in a much larger object class. Other natural examples are obtained from graph theory, fields, greedy algorithms. There are several cryptomorphic ways to define a matroid. In our context, we use the definition based on ranks. Using ranks allows to deal only with points which makes proofs easier because we do not handle directly lines or planes. An integer function *rk* on E a finite set is the rank function of a matroid if and only if the following conditions are satisfied:

(* nonnegative and subcardinal *)
(A1R2-R3) : $0 \leq \mathrm{rk}(\mathrm{X}) \leq \left|\mathrm{X}\right|, \forall\ \mathrm{X} \subseteq \mathrm{E}$

(* nondecreasing *)
(A2R2-R3) : $\mathrm{rk}(\mathrm{X}) \leq \mathrm{rk}(\mathrm{Y}), \forall\ \mathrm{X} \subseteq \mathrm{Y}$

(* submodular *)
(A3R2-R3) : $\mathrm{rk}(\mathrm{X} \cup \mathrm{Y}) + \mathrm{rk}(\mathrm{X} \cap \mathrm{Y}) \leq \mathrm{rk}(\mathrm{X}) + \mathrm{rk}(\mathrm{Y}), \forall\ \mathrm{X}, \mathrm{Y} \subseteq \mathrm{E}$

3.2 Rank to Describe Incidence Projective Geometry

In the framework of projective geometry, we define a rank function on finite sets of points which checks the three conditions above. We specify the notion of

flat which is a set of points closed by the collinearity/coplanarity relation. The rank of a flat is the cardinal of a smallest set generating A (see Fig. 1 for some examples).

rk{A,B} = 1	A = B
rk{A,B} = 2	A ≠ B
rk{A,B,C} = 2	A,B,C are collinear with at least two of them distinct
rk{A,B,C} ≤ 2	A,B,C are collinear
rk{A,B,C} = 3	A,B,C are not collinear
rk{A,B,C,D} = 3	A,B,C,D are coplanar, not all collinear
rk{A,B,C,D} = 4	A,B,C,D are not coplanar

Fig. 1. Some rank statements and their geometric interpretations

Using this definition, it can be shown that every projective space has a matroid structure, but the converse is not true. To capture all the projective geometry, we need to introduce some additional axioms to matroid's one.

3.3 2D Rank-based Axioms System

We present a rank-based axiom system to describe projective plane. The first two axioms establish the non degeneracy of the rank function. The others are more or less direct translations of the axioms of projective geometry.

(A4R2) Rk-Singleton : ∀ P : Point, rk{P} ≥ 1

(A5R2) Rk-Couple : ∀ P Q: Point, P ≠ Q ⇒ rk{P, Q} ≥ 2

(A6R2) Rk-Inter : ∀ A B C D, ∃ J, rk{A, B, J} = rk{C, D, J} = 2

(A7R2) Rk-Three-Points : ∀ A B, ∃ C, rk{A, B, C} = rk{B, C} = rk{A, C} = 2

(A8R2) Rk-Lower-Dimension : ∃ A B C, rk{A, B, C} ≥ 3

3.4 ≥3D Rank-based Axioms System

In the same way, we define a rank-based axiom system to describe projective space. Again, we modify only the axioms of Pasch and Lower-Dimension.

(A4R3) Rk-Singleton : ∀ P : Point, rk{P} ≥ 1

(A5R3) Rk-Couple : ∀ P Q: Point, P ≠ Q ⇒ rk{P, Q} ≥ 2

(A6R3) Rk-Pasch : $\forall$ A B C D, rk{A, B, C, D} $\leq 3 \Rightarrow \exists$ J, rk{A, B, J} = rk{C, D, J} = 2

(A7R3) Rk-Three-Points : $\forall$ A B, $\exists$ C, rk{A, B, C} = rk{B, C} = rk{A, C} = 2

(A8R3) Rk-Lower-Dimension : $\exists$ A B C D, rk{A, B, C, D} ≥ 4

The Implementation in Coq follows the same process, we use *type classes* to increase modularity of the code depending of the dimension.

Now that we have specified the axiomatization, we will focus on proof of the equivalence between these two axiom systems. In previous works [19, 20], only the implication of the ranks to $\geq$3D projective geometry has been proven. It is sufficient to justify the case study on the formalization of Desargues theorem with the ranks [19, 20]. To allow bilateral translation and to increase the possibilities of automation, we have completed the proof of this equivalence.

4 Equivalence Proof

We prove the following statement:

Theorem 1. *Axiomatization on incidence projective geometry and rank-based axioms system are equivalent respectively in 2D and $\geq$ 3D.*

To achieve this proof, we split the demonstration of the equivalence depending on the dimension and the direction.

4.1 From Ranks to Projective Geometry

We begin with the implication from ranks to projective geometry both 2D and $\geq$ 3D. The conduct of the proofs remains the same no matter the dimension.

Preliminaries

First, we need to characterize the notion of line and incidence that does not exist in the axiom system based on ranks. We build, using an inductive definition, a line from two distinct points. A point P is incident to a line, if the rank of the triple formed by the two points constituting the line and the point P remains equal to 2.

```
Definition Point := Point.

Inductive LineInd : Type :=
|Cline : forall (A B : Point)(H : ~ A[==]B), LineInd.

Definition Line := LineInd.
```

```
Definition Incid (P : point)(l : Line) := rk ((fstP l)(sndP l) P) = 2.
```

From there we can express the axioms of projective geometry and prove them using rank axioms.

Submodularity

We detail proof techniques used to demonstrate the five axioms in projective geometry both 2D and ≥3D. First, we usually prove equalities on ranks (rk(a) = rk(b)) in two steps : first that rk(a) ≤ rk(b), then rk(a) ≥ rk(b). Second, we work systematically with the axiom of submodularity (A3R2-R3). The issue is that to determine the intersection of two finite set of points, we need to distinguish cases about the equalities of points. The resulting proofs becomes more complex with these distinctions. Therefore, we define a particular lemma derived from the axiom A3R2-R3 that does not consider the theoretical intersection but a lower approximation of this intersection (noted ⊓).

Definition 1. *(Literal Intersection).*
Let L_1 and L_2 be two sets of points. By definition $L_1 \sqcap L_2$ is the intersection of the two sets of points considered syntactically.

From this literal intersection, we can derive a more appropriate version of the axiom ignoring case distinctions:

Lemma 1. *(A3R2-R3-lit),*
$\forall\ X\ Y,\ rk(X \cup Y) + rk(X \sqcap Y) \leq rk(X) + rk(Y).$

In Coq, it is not possible to define the literal intersection. To capture the meaning of the lemma, we define a practical version:

Lemma 2. *(A3R2-R3-alt),*
$\forall\ X\ Y\ I,\ I \subseteq X \cap Y \Rightarrow rk(X \cup Y) + rk(I) \leq rk(X) + rk(Y).$

This functional version of the submodularity will be extensively employed in each proof. To illustrate the mechanism of proof of this subsection, we detail the demonstration of the Uniqueness property.

Proof of the uniqueness property

Lemma 3. *(A3P2)(A3P3) Uniqueness,*
$\forall\ A\ B : Point, \forall\ l\ m : Line,\ A \in l \wedge B \in l \wedge A \in m \wedge B \in m \Rightarrow A = B \vee l = m$

Proof (test). We begin this proof by performing a case distinction on the equality between A and B. The first case A = B is immediately checked with the goal. If A ≠ B, by unfolding definitions of `Line` and `Incid`, it follows that :

We have $A <> B$
Let P ∈ l, Q ∈ l and $P <> Q$
Let R ∈ m, S ∈ m and $R <> S$
Incid A l ⇒ rk{P, Q, A} = 2
Incid B l ⇒ rk{P, Q, B} = 2
Incid A m ⇒ rk{R, S, A} = 2
Incid B m ⇒ rk{R, S, B} = 2
} *Assumptions*

---

l = m ⇒ rk(P Q R S}) = 2 } *Goal*

To continue, we determine the rank of two new sets using A3R2-R3-alt: rk{P, Q, A, B} and rk{R, S, A, B}.

rk({P, Q, A} ∪ {P, Q, B}) + rk({P, Q, A} ∩ {P, Q, B})
≤ rk{P, Q, A} + rk{P, Q, B}
⇒ rk{P, Q, A, B} + rk{P, Q} ≤ rk{P, Q, A} + rk{P, Q, B}
⇒ rk{P, Q, A, B} ≤ 2

Analogously, we calculate rk{R, S, A, B} ≤ 2. Then we establish that rk{P, Q, R, S, A, B} ≤ 2.

rk({P, Q, A, B} ∪ {R, S, A, B}) + rk({P, Q, A, B} ∩ {R, S, A, B})
≤ rk{P, Q, A, B} + rk{R, S, A, B}
⇒ rk{P, Q, R, S, A, B} + rk{A, B} ≤ rk{P, Q, A, B} + rk{R, S, A, B}
⇒ rk{P, Q, R, S, A, B} ≤ 2

Using A2R2-R3, we prove that:

{P, Q, R, S} ⊂ {P, Q, R, S, A, B} ⇒
rk{P, Q, R, S} ≤ rk{P, Q, R, S, A, B}

So rk{P, Q, R, S} ≤ 2 and contains at least two distinct points, we conclude that rk{P, Q, R, S}=2.

Coq implementation

At the implementation level, to make the deduction on abstract sets, we use the tactic `fsetdecide` provided by the library on Containers [16]. We also use setoids to make substitutions when we have to indicate to the system that the sets are identical. Finally to address inequalities, we use `omega` [5][1] tactic. We proved with similar techniques others lemmas of projective geometry.

[1] Omega: a solver of quantifier-free problems in Presburger Arithmetic

4.2 From Projective Geometry to Ranks

The opposite direction is more difficult to handle, we need to specify the concept of rank from projective geometry. This implication is much more difficult to prove.

Preliminaries

It is not possible to directly describe rank, we have to define intermediate definitions representing the different returned values of the rank function. To characterize the rank, we are inspired by the concept of flat from the matroid theory. The set of points is either empty or it represents a point, a line, a plane or a space. To represent abstract sets, we can manipulate AVLs or lists. Given that we are working on sets of small sizes, we choose to work with lists for the implementation.

```
Definition rkl s := match s with
| nil => 0
| x :: nil => 1
| s => if contains_four_non_coplanar_points s then 4 else
          if contains_three_non_collinear_points s then 3 else
             if contains_two_distinct_points s then 2 else 1
end.
```

The three predicates `contains_four`, `contains_three` and `contains_two` put bounds on the dimension of the set. For instance the coplanarity, either there is a quadruple of non coplanar points and the set represents a space, or it continue recursion analyzing collinearity. These predicates form what is called an intermediate layer. They help to make transition between the two axiom systems and are often accompanied by dozens of lemmas. It is desirable to remain vigilant during the development of such definitions to take into account all the degenerate cases (mainly coincident points).

```
Fixpoint contains_four_non_coplanar_points l :=
match l with
| nil => false
| a::r => if coplanar_with_all a (all_triples r)
              then contains_four_non_coplanar_points r
              else true
end.
```

Proof techniques

Similarly, we present the main technical aspects to perform demonstrations of the eight axioms. We exploit mostly the induction mechanism. Manipulation of the three conditionals in the rank's characterization and the management of

degenerate cases yield many goals which significantly increase the size of the proofs. Resulting proofs are often tedious, however it is possible to automate many steps. For this, we use Ltac the tactical language provided by Coq. We identify specific patterns within hypothesis and goals that we simplify by a sequence of tactics and lemmas. This tactic allows to unfold as much as possible the goal while introducing a maximum of assumptions. If the goal has become trivial with a simple equality/inequality, the work is finished. Otherwise one has to make explicit a contradiction in assumptions. This logic work is the core of the proof and must be done by hand. To raise contradictions in context, it makes sense to write intermediate lemmas.

```
Ltac my_rank :=
  repeat match goal with
  |[H : _ |- _] => progress intros
  |[H : _ |- _] => progress intro
  |[H : _ |- _ <-> _] => split
  |[H : _ |- _ /\ _] => split
  |[H : _ /\ _ |- _] => destruct H
  |[H : _ |- _] => solve[intuition]
  |[H : _ |- _] => progress contradiction
  |[H : ?X :: _ = nil |- _] => inversion H
  |[H : false = true |- _] => inversion H
  |[H : true = false |- _] => inversion H
  |[H : ?X[==]?X -> False |- _] => apply False_ind;apply H;reflexivity
  |[H : _ |- (if if ?X then _ else _
                then _ else _) = _ \/ _] => case_eq X
  |[H : _ |- (if ?X then _ else _) = _ \/ _] => case_eq X
  ...
  end.
```

Proof of the nondecreasing property

We give an overview with the proof of the nondecreasing property of the matroid theory.

Lemma 4. *(A2R2-R3),*
$\forall\ X \subseteq\ Y,\ rk(X) \leq rk(Y).$

In our context, the rank function can take five possible values from 0 to 4, which leads us twenty five different cases to treat when we make a structural induction on X and Y. First case $0 \leq 0$, second case $0 \leq 1$, etc. With the tactic, we eliminate fifteen trivial goals immediately wherein X is included in Y. To prove the remaining ones, we build a contradiction and we simplify with a tactic that automatically handles the calculation of inclusion. For example, we clarify that a plane can not be included in a line. It is therefore impossible to reach the case $3 \leq 2$.

Coq implementation

Lemma A3R2-R3 is the hardest lemma to prove with the management of the intersection and the union. It contains many nontrivial cases that we decompose into several lemmas as below.

```
Lemma matroid3_rk2_rk2_interrk2_to_unionrk2 :
forall l m,
rkl l = 2 ->
rkl m = 2 ->
rkl (inter l m) = 2 ->
rkl (union l m) = 2.
```

Another recurring issue is caused by our parametric equality on points. We must write a morphism for each definition that we cross when we perform a rewrite. We prove that the equality is compatible and consistent with the definition in which it is used. In this way, it is possible to substitute a set X similar to a set Y directly in the rank definition. The writing of such morphism in Coq is as below:

```
Instance rank_morph : Proper (@equivlistA Point eq => (@Logic.eq nat)) rkl.
Proof.
...
Qed.
```

This leads to the following lemma where the predicate `equivlist` indicates that the two lists are equal:

Lemma 5. *rank_morph,*
$\forall\ x\ y$ *: list Point, equivlist* $x\ y \rightarrow$ *rkl* $x =$ *rkl* y

4.3 Statistics

Finally, we give some data about our library, we also detail the impact of automation that we have put in place throughout the process. Overall, our development consists of more than 30 000 lines and 850 lemmas. We dedicate 14 000 lines to demonstrate the equivalence organized as shown in the Fig. 2 below. Using tactics and splitting the demonstration into lemmas has divided by three the size of the formalization.

5 Conclusion and Future Work

This paper explain how projective geometry can be formalized in Coq using two different axioms systems in both 2D and 3D. We detail in particular the original axiomatization based on the notion of rank from the matroid theory. Then we

	From rank to PG		From PG to rank	
	2D	≥3D	2D	≥3D
lines of Coq specs	250	350	650	1 050
lines of Coq proofs	300	800	2 600	11 000

Fig. 2. Organization of equivalence proof in Coq

prove rigorously equivalence between these two formalizations. We highlight the tools and techniques used to achieve it by presenting a typical proof. Finally we give some thoughts on proof automation in presence of ranks.

In the future, we plan to carry on our investigations in two main directions. On the one hand, we expect to write a reliable algorithm for performing a bilateral translation between the two approaches. In this way we can alternate between geometry for visualization and combinatorics for automation and computation.

On the other hand, we shall study how to effectively automate demonstrations about ranks at different levels. First, we will investigate the possibilities of full automation in the case of finite geometry like Fano plane. Secondly, we are interested in a partial automation of many steps in the Desargues theorem proof presented in [19,20]. We want to make proof as readable as possible by removing technical details, thus being as close as possible to mathematical proof.

Availability The full Coq development is available at the following url: http://galapagos.gforge.inria.fr/

References

1. Bertot, Yves and Castéran, Pierre: Interactive Theorem Proving and Program Development, Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science, Springer Science & Business Media (2004)
2. Boutry, Pierre and Narboux, Julien and Schreck, Pascal and Braun, Gabriel: Using small scale automation to improve both accessibility and readability of formal proofs in geometry. Automated Deduction in Geometry 2014 pp. 1–19 (2014)
3. Buekenhout, Francis: Handbook of Incidence Geometry: buildings and foundations. Elsevier (1995)
4. Castéran, Pierre and Sozeau, Matthieu: A Gentle Introduction to Type Classes and Relations in Coq (2016)
5. Coq development team: The Coq proof Assistant Reference Manual. https://coq.inria.fr/distrib/current/files/Reference-Manual.pdf (2016)
6. Coxeter, Harold Scott Macdonald: Projective Geometry. Springer Science & Business Media (2003)
7. Dehlinger, Christophe and Dufourd, Jean-François and Schreck, Pascal: Higher-Order Intuitionistic Formalization and Proofs in Hilbert's Elementary Geometry. Automated Deduction in Geometry pp. 306–324 (2000)

8. Duprat, Jean: Une axiomatique de la géométrie plane en Coq. Journées Francophones des Langages Applicatifs pp. 123–136 (2008)
9. Fuchs, Laurent and Thery, Laurent: A formalization of grassmann-cayley algebra in Coq and its application to theorem proving in projective geometry. Automated Deduction in Geometry 6877, 51–67 (2010)
10. Génevaux, Jean-David and Narboux, Julien and Schreck, Pascal: Formalization of Wu's simple method in Coq. Certified Programs and Proofs 7086, 71–86 (2011)
11. Guilhot, Frédérique: Formalisation en Coq et visualisation d'un cours de géometrie pour le lycée. Journées Francophones des Langages Applicatifs 7, 15 (2004)
12. Janičić, Predrag and Narboux, Julien and Quaresma, Pedro: The Area Method : a Recapitulation. Journal of Automated Reasoning 48(4), 489–532 (2012)
13. Jermann, Christophe and Trombettoni, Gilles and Neveu, Bertrand and Mathis, Pascal: Decomposition of Geometric Constraint Systems: a survey. International Journal of Computanional Geometry & Application 16(05n06), 379–414 (2006)
14. Kahn, Gilles: Constructive geometry according to Jan von Plato (1995)
15. Kusak, Eugeniusz: Desargues theorem in projective 3-space. J. of Formalized Mathematics 2 (1990)
16. Lescuyer, Stéphane: First-Class Containers in Coq. Stud. Inform. Univ. 9(1), 87–127 (2011)
17. Li, Hongbo and Wu, Yihong: Automated short proof generation for projective geometric theorems with Cayley and bracket algebras: I. Incidence geometry. Journal of Symbolic Computation 36(5), 717–762 (2003)
18. Magaud, Nicolas and Narboux, Julien and Schreck, Pascal: Formalizing Projective Plane Geometry in Coq. Automated Deduction in Geometry 6301, 141–162 (2008)
19. Magaud, Nicolas and Narboux, Julien and Schreck, Pascal: Formalizing Desargues theorem in Coq using ranks. ACM pp. 1110–1115 (2009)
20. Magaud, Nicolas and Narboux, Julien and Schreck, Pascal: A Case Study in Formalizing Projective Geometry in Coq: Desargues Theorem. Computational Geometry : Theory and Applications 45(8), 406–424 (2012)
21. Meikle, Laura I and Fleuriot, Jacques D: Formalizing Hilbert's Grundlagen in Isabelle/Isar. Theorem proving in higher logics 2758, 319–334 (2003)
22. Michelucci, D and Schreck, P: Detecting induced incidences in the projective plane. isiCAD Workshop. Citeseer (2004)
23. Michelucci, Dominique and Foufou, Sebti and Lamarque, Loic and Schreck, Pascal: Geometric constraints solving:some tracks. ACM pp. 185–196 (2006)
24. Michelucci, Dominique and Schreck, Pascal: Incidence constraints : a combinatorial approach. International J. of Computational Geometry & Application 16(05n06), 443–460 (2006)
25. Moulton, Forest Ray: A Simple Non-Desarguesian Plane Geometry. Transactions of the American Mathematical Society 3(2), 192–195 (1902)
26. Narboux, Julien: A Decision Procedure for Geometry in Coq. Theorem Proving in Higher Order Logics 3223, 225–240 (2004)
27. Narboux, Julien: Formalisation et automatisation du raisonnement géométrique en Coq. Ph.D. thesis, Université Paris Sud-Paris XI (2006)
28. Narboux, Julien: Mechanical Theorem Proving in Tarski's Geometry. Automated Deduction in Geometry 4869, 139–156 (2006)
29. Oxley, James G: Matroid Theory, vol. 3. Oxford University Press, USA (2006)
30. Richter-Gebert, Jürgen: Mechanical theorem proving in projective geometry. Annals of Mathematics and Artificial Intelligence 13, 139–172 (1995)
31. Sozeau, Matthieu and Oury, Nicolas: First-class type classes. Theorem Proving in Higher Order Logics 5170, 278–293 (2008)

32. Tarski, Alfred: What is Elementary Geometry ? (1983)
33. von Plato, Jan: The axioms of constructive geometry. Annals of pure and applied logic 76(2), 169–200 (1995)

From Hilbert to Tarski

Gabriel Braun, Pierre Boutry, and Julien Narboux

ICube, UMR 7357 CNRS, University of Strasbourg
Pôle API, Bd Sébastien Brant, BP 10413, 67412 Illkirch, France
{gabriel.braun, boutry, narboux}@unistra.fr

Abstract. In this paper, we describe the formal proof using the Coq proof assistant that Tarski's axioms for plane neutral geometry (excluding continuity axioms) can be derived from the corresponding Hilbert's axioms. Previously, we mechanized the proof that Tarski's version of the parallel postulate is equivalent to the Playfair's postulate used by Hilbert [9] and that Hilbert's axioms for plane neutral geometry (excluding continuity) can be derived from the corresponding Tarski's axioms [12]. Hence, this work allows us to complete the formal proof of the equivalence between the two axiom systems for neutral and plane Euclidean geometry. Formalizing Hilbert's axioms is not completely straightforward, in this paper we describe the corrections we had to make to our previous formalization. We mechanized the proof of Hilbert's theorems that are required in our proof of Tarski's axioms. But this connection from Hilbert's axioms, allows to recover the many results we obtained previously in the context of Tarski's geometry: this includes the theorems of Pappus[13], Desargues, Pythagoras and the arithmetization of geometry [8].

Keywords: formal proof, geometry, foundations, Tarski's axioms, Hilbert's axioms, Coq

1 Introduction

There are several ways to define the foundations of geometry. In the synthetic approach, the axiomatic system is based on some geometric objects and axioms about them. The best known modern axiomatic systems based on this approach are Hilbert's [17] and Tarski's ones [32]. In the analytic approach, a field $\mathbb{F}$ is assumed (usually $\mathbb{R}$) and the space is defined as $\mathbb{F}^n$. In the mixed analytic/synthetic approaches, one assumes both the existence of a field and also some geometric axioms. For example, the axiomatic systems for geometry used for education in North America are based on Birkhoff's axiomatic system [7] in which the field serves to measure distances and angles. This is called the metric approach. A modern development of geometry based on this approach can be found in the books of Millman [21] or Moise [22]. The textbook of Hartshorne [16] provides a development of geometry based on Hilbert's axioms. A rigorous development of geometry based on Tarski's axioms appears in the first part of the book by Tarski, Szmielew and Schwabhäuser [27].

The GeoCoq library gathers results about the foundations of geometry formalized in Coq [23]. It includes a formalization of the first part of [27]. We formalized in Coq the

link between the synthetic geometry defined by Tarski's axioms and analytic geometry [8]. We also formalized the link from Tarski's axioms to Hilbert's axioms [12], Beeson has later written a note [5] to demonstrate that the main results to obtain Hilbert"s axioms are contained in [27]. In this paper, we complete the picture, by proving formally that Tarski's axioms can be derived from Hilbert's axiom.

Dehlinger, Dufourd and Schreck have studied the formalization of Hilbert's foundations of geometry in the intuitionist setting of Coq [14]. They have highlighted the fact that Hilbert's proofs require the excluded middle axiom applied to the point equality and the role of non degenerate conditions. They focus on the first two groups of axioms and prove some betweenness properties. Meikle and Fleuriot have done a similar study within the Isabelle/HOL proof assistant [20]. They went up to twelfth[1] theorem of Hilbert's book. Scott has continued the formalization of Meikle using Isabelle/HOL and revised it [28]. He has corrected some "subtle errors in the formalization of Group III by Meikle". Scott was interested in trying to obtain readable proofs. Later, he developed a system within the HOL-Light proof assistant to automatically fill some gaps in the incidence proofs [30]. In our work, even if we use some tactics to obtain some proofs more easily [11,10], we do not focus on obtaining readable proofs. Scott then focused on proving a theorem, which Hilbert's states without proof: that a special case of the Jordan Curve Theorem can be proved using the first two groups of Hilbert's axioms only [29]. Moreover Richter has formalized a substantial number of results based on Hilbert's axioms and a metric axiom system using HOL-Light [25]. Finally Beeson and Wos have used the OTTER automatic theorem prover to mechanize some proofs in Tarski's geometry [3,4]. They manage to automatically prove many results from the first twelve chapter of [27], with the help of some human generated hints OTTER can prove all the lemmas necessary to derive Hilbert's axioms.

The earlier errors in Meikle's formalization found by Scott can frighten the reader. How can we be sure that there are no errors in the formalization presented in this paper? They are two kinds of potential errors:

1. the axioms are inconsistent
2. the axioms do not capture the desired theory - some standard geometry theorems may not be provable from the axioms

In our previous work [12], we have shown that Tarski's axioms for plan Euclidean geometry can serve as a model for the corresponding Hilbert's axioms. In an unpublished result, Boutry has built a formal proof that the Cartesian plane over a Pythagorean ordered field is a model of Tarski's axioms (A1-A10, i.e. excluding the continuity axiom). This can convince the reader that Tarski's axioms as they are formalized are consistent and hence our formalization of Hilbert's axiom system as well. Makarios has also shown using the Isabelle/HOL proof assistant that there are both euclidean and a non-euclidean models of Tarski's axioms for plane neutral geometry [19]. However our axiom system could be weaker than necessary. As a matter of fact, the axiom system we proposed in 2012 was not sufficient to prove Tarski's axioms and we had to correct it. In this paper, we present a formal proof that the formalization of Hilbert's axioms is not only correct

[1] We use the numbering of theorems as of the tenth edition.

but also sufficient, in the sense that we can obtain the culminating result of both [17] and [27]: the arithmetization of geometry.

Outline

The paper is organized as follows. In Section 2, we describe Tarski's axiom system (the version used in [27]), then (Sec. 3) our formalization of Hilbert's axioms for plane Euclidean geometry. In doing so, we present the errors we had to correct in our previous axiomatic system to obtain the equivalence with Tarski's axioms. Finally, in Section 4, we provide the proof that Tarski's axioms can be derived from Hilbert's axioms.

2 Tarski's Axioms

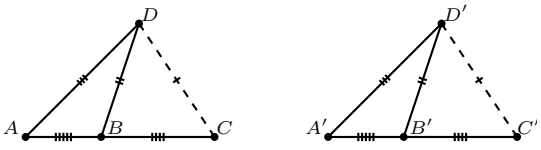
We use the axioms that serves as a basis for [27]. For an explanation of the axioms and their history see [32]. Table 1 lists the axioms for neutral geometry.

Let us recall that Tarski's axiom system is based on a single primitive type depicting points and two predicates, namely between noted by $-\!\!-$ and congruence noted by $\equiv$. $A{-}B{-}C$ means that A, B and C are collinear and B is between A and C (and B may be equal to A or C). $AB \equiv CD$ means that the line-segments $\overline{AB}$ and $\overline{CD}$ have the same length (the orientation does not need to be the same).

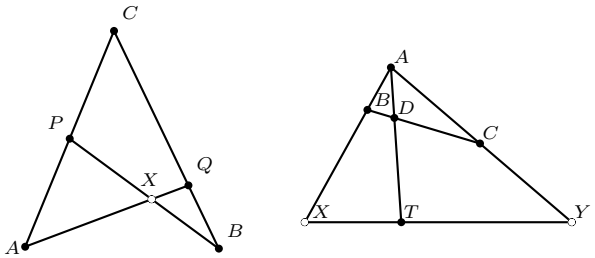
Figure 2 provides the Coq's formalization of these axioms. We worked in an intuitionistic setting but we included the axiom `point_equality_decidability`.

A1	Symmetry	$AB \equiv BA$
A2	Pseudo-Transitivity	$AB \equiv CD \wedge AB \equiv EF \Rightarrow CD \equiv EF$
A3	Cong Identity	$AB \equiv CC \Rightarrow A = B$
A4	Segment construction	$\exists E, A{-}B{-}E \wedge BE \equiv CD$
A5	Five-segment	$AB \equiv A'B' \wedge BC \equiv B'C' \wedge$ $AD \equiv A'D' \wedge BD \equiv B'D' \wedge$ $A{-}B{-}C \wedge A'{-}B'{-}C' \wedge A \neq B \Rightarrow CD \equiv C'D'$
A6	Between Identity	$A{-}B{-}A \Rightarrow A = B$
A7	Inner Pasch	$A{-}P{-}C \wedge B{-}Q{-}C \Rightarrow \exists X, P{-}X{-}B \wedge Q{-}X{-}A$
A8	Lower Dimension	$\exists ABC, \neg A{-}B{-}C \wedge \neg B{-}C{-}A \wedge \neg C{-}A{-}B$
A9	Upper Dimension	$AP \equiv AQ \wedge BP \equiv BQ \wedge CP \equiv CQ \wedge P \neq Q$ $\Rightarrow A{-}B{-}C \vee B{-}C{-}A \vee C{-}A{-}B.$
A10	Parallel postulate	$\exists XY\,(A{-}D{-}T \wedge B{-}D{-}C \wedge A \neq D \Rightarrow$ $A{-}B{-}X \wedge A{-}C{-}Y \wedge X{-}T{-}Y)$

Table 1: Tarski's axiom system for neutral geometry.



(a) Five-segment axiom



(b) Inner Pasch's axiom (c) Parallel postulate

Fig. 1: Illustration for three axioms

The symmetry axiom (A1 on Table 1) for equidistance together with the transitivity axiom (A2) for equidistance imply that the equi-distance relation is an equivalence relation. The identity axiom for equidistance (A3) ensures that only degenerate line segments can be congruent to a degenerate line segment. The axiom of segment construction (A4) allows to extend a line segment by a given length. The five-segments axiom (A5) is similar to the Side-Angle-Side principle, but expressed without mentioning angles, using only the betweenness and congruence relations (Fig. 1a). The lengths of $\overline{AB}$, $\overline{AD}$ and $\overline{BD}$ fix the angle $\widehat{CBD}$. The identity axiom for betweenness expresses that the only possibility to have B between A and A is to have A and B equal. Tarski's relation of betweenness is non-strict, unlike Hilbert's. The inner form of the Pasch's axiom (A7, Fig. 1b) is a variant of the axiom that Pasch introduced in [24] to repair the defects of Euclid. Pasch's axiom intuitively says that if a line meets one side of a triangle and does not pass through the endpoints of that side, then it has to meet one of the other sides of the triangle. Inner Pasch's axiom is a form of the axiom that holds even in 3-space, i.e. does not assume a dimension axiom. The lower 2-dimensional axiom (A8) asserts that the existence of three non-collinear points. The upper 2-dimensional axiom (A9) implies that all the points are coplanar. The version of the parallel postulate (A10) is a statement which can be expressed easily in the language of Tarski's geometry (Fig. 1c). It is equivalent to the uniqueness of parallels or Euclid's 5th postulate.

```
Class Tarski_neutral_dimensionless :=
{
 Tpoint : Type;
 Bet : Tpoint -> Tpoint -> Tpoint -> Prop;
 Cong : Tpoint -> Tpoint -> Tpoint -> Tpoint -> Prop;
 cong_pseudo_reflexivity : forall A B, Cong A B B A;
 cong_inner_transitivity : forall A B C D E F,
   Cong A B C D -> Cong A B E F -> Cong C D E F;
 cong_identity : forall A B C, Cong A B C C -> A = B;
 segment_construction : forall A B C D,
   exists E, Bet A B E /\ Cong B E C D;
 five_segment : forall A A' B B' C C' D D',
   Cong A B A' B' ->
   Cong B C B' C' ->
   Cong A D A' D' ->
   Cong B D B' D' ->
   Bet A B C -> Bet A' B' C' -> A <> B -> Cong C D C' D';
 between_identity : forall A B, Bet A B A -> A = B;
 inner_pasch : forall A B C P Q,
   Bet A P C -> Bet B Q C ->
   exists X, Bet P X B /\ Bet Q X A;
 PA : Tpoint;
 PB : Tpoint;
 PC : Tpoint;
 lower_dim : ~ (Bet PA PB PC \/ Bet PB PC PA \/ Bet PC PA PB)
}.

Class Tarski_neutral_dimensionless_with_dec_point_equality
 `(Tn : Tarski_neutral_dimensionless) :=
{
 point_equality_decidability : forall A B : Tpoint, A=B \/ ~A=B
}.

Class Tarski_2D
 `(TnEQD : Tarski_neutral_dimensionless_with_dec_point_equality) :=
{
 upper_dim : forall A B C P Q,
   P <> Q -> Cong A P A Q -> Cong B P B Q -> Cong C P C Q ->
   (Bet A B C \/ Bet B C A \/ Bet C A B)
}.

Class Tarski_2D_euclidean `(T2D : Tarski_2D) :=
{
 euclid : forall A B C D T,
   Bet A D T -> Bet B D C -> A<>D ->
   exists X, exists Y,
   Bet A B X /\ Bet A C Y /\ Bet X T Y
}.
```

Fig. 2: The formalization of Tarski's axioms

3 Hilbert's Axioms

Our formalization of Hilbert's axiom system is derived on the French translation of the tenth edition annotated by Rossier [18].

Hilbert's axiom system is based on two abstract types: points and lines (as we limit ourselves to 2-dimensional geometry we did not introduce 'planes' and the related axioms).

In our initial version of Hilbert's axioms, we made several mistakes. None of the axioms were incorrect (as they are formally proved from Tarski's axioms), but some should be strengthened and some others are useless because they can be derived. For each group of axioms, we detail the changes we made to our previous formalization, we do not detail all the axioms that have already been described in [12]. But, we give the full list of axioms as a reference.

3.1 Group I

Group I of axioms contains the incidence axioms.

First, we had to change the lower dimensional axiom. Hilbert states that there exists three non collinear points and three points are said to be collinear if there exists a line going through these three points. This assumption is not enough, because in a world without lines, assuming that there are three non collinear points does not imply that they are distinct. Indeed, there is a model of the first two groups of Hilbert's axioms with only one point and no lines (interpreting congruence by the empty relation). We can construct a line only if we have two distinct points. Scott's formalization does not need this modification because in Isabelle/HOL all types are inhabited, hence his formalization includes implicitly the fact that there is at least one line. Meikle's and Richter's formalizations enforce that the three points are distinct.

Hence, for the lower dimension axiom, we state that there is a line and point not on this line:

```
l0 : Line;
P0 : Point;
plan : ~ Incid P0 l0;
```

As there is an axiom stating there are always at least two points on a line, the former axiom stating that there are three non collinear points can be derived.

Second, we had to introduce the property that line equality is an equivalence relation and that incidence is a morphism for line equality:

```
EqL_Equiv : Equivalence EqL;
Incid_morphism :
  forall P l m, Incid P l -> EqL l m -> Incid P m;
```

Finally, as we are working in an intuitionist setting we had to introduce some decidability properties which allow to perform case distinctions. It would be interesting to formalize a constructive version of Hilbert's axioms, following Beeson's work [1], we leave this for future work.

```
Incid_dec : forall P l, Incid P l \/ ~Incid P l;
eq_dec_pointsH : forall A B : Point, A=B \/ ~ A=B;
```

The complete list of axioms for group I is given in Fig. 3.

```
Point : Type;
Line  : Type;
EqL   : Line -> Line -> Prop;
EqL_Equiv : Equivalence EqL;
Incid : Point -> Line -> Prop;
Incid_morphism :
  forall P l m, Incid P l -> EqL l m -> Incid P m;
Incid_dec : forall P l, Incid P l \/ ~ Incid P l;
eq_dec_pointsH : forall A B : Point, A=B \/ ~ A=B;
line_existence :
  forall A B, A <> B -> exists l, Incid A l /\ Incid B l;
line_uniqueness :
  forall A B l m,
    A <> B ->
    Incid A l -> Incid B l -> Incid A m -> Incid B m ->
        EqL l m;
two_points_on_line :
  forall l,
    { A : Point & { B | Incid B l /\ Incid A l /\ A <> B}};
ColH :=
  fun A B C => exists l, Incid A l /\ Incid B l /\ Incid C l;
l0 : Line;
P0 : Point;
plan : ~ Incid P0 l0;
```

Fig. 3: Formalization of Group I

3.2 Group II

Group II of axioms contains the betweenness axioms. We denote by $A\text{-}\!\!*B\text{-}\!\!*C$ Hilbert's betweenness predicate, which is strict. It expresses the fact that B is on the line AC between A and C and different from A and C. We could not derive the fact that if $A\text{-}\!\!*B\text{-}\!\!*C$ then $A \neq C$ from our former axioms so we added this property as an axiom. The fact that $A \neq B$ and $B \neq C$ (which is assumed by Greenberg, Hartshorne and Richter) is not necessary as it can be derived from the other axioms. The fact that A should be different from C is not explicit in Hilbert's book.

```
between_diff : forall A B C, BetH A B C -> A <> C;
```

The property `between_one` states that given three collinear points distinct points at least one of them is between the other two:

```
between_one :
  forall A B C,
    A <> B -> A <> C -> B <> C -> ColH A B C ->
    BetH A B C \/ BetH B C A \/ BetH B A C.
```

In our earlier formalization as well as earlier editions of Hilbert's book, this property was taken as an axiom. Following the proof by Wald published by Hilbert in later editions, we derived it from the other axioms. Richter assumes this property. Moreover, in the axiom `between_only_one`, we removed one of the conjuncts as it can be derived. We now have:

```
between_only_one : forall A B C, BetH A B C -> ~ BetH B C A;
```

instead of:

```
between_only_one :
forall A B C, BetH A B C -> ~ BetH B C A /\ ~ BetH B A C;
```

The other axioms could be kept unmodified. The axioms for the second group are given in Fig. 4.

```
BetH   : Point -> Point -> Point -> Prop;
between_col :  forall A B C, BetH A B C -> ColH A B C;
between_diff : forall A B C, BetH A B C -> A <> C;
between_comm : forall A B C, BetH A B C -> BetH C B A;
between_out :  forall A B, A <> B -> exists C, BetH A B C;
between_only_one : forall A B C, BetH A B C -> ~ BetH B C A;
cut :=
  fun l A B => ~ Incid A l /\ ~ Incid B l /\
               exists I, Incid I l /\ BetH A I B;
pasch :
  forall A B C l,
    ~ ColH A B C -> ~ Incid C l -> cut l A B ->
    cut l A C \/ cut l B C;
```

Fig. 4: Formalization of Group II

3.3 Group III

Group III of axioms contain those about congruence of segments and angles.

Congruence of segments Hilbert defines congruence as a relation about segments, where segments are defined as unordered pairs of points. In our formalization, we chose to avoid defining the concept of segment. We denote by $\equiv_H$ the Hilbert segment congruence relation. Hence, we have an axiom which says that segments can be permuted on the right. Other permutations can be derived.

```
cong_permr : forall A B C D, CongH A B C D -> CongH A B D C;
```

The uniqueness of segment construction can be derived if one assume the reflexivity of congruence of angles, therefore we dropped this axiom. Richter assumes uniqueness of segment construction. The other axioms were not changed, the full list is given on Fig. 5. Our formalization of the segment addition axiom follows Hilbert's prose. It is based on the definition of the concept of disjoint segments. Note that in the segment addition axiom, the concept of disjoint segments could be replaced by a betweenness assumption stating that B is between A and C and B' is between A' and C', we proved it as lemma:

```
Lemma addition_betH : forall A B C A' B' C',
  BetH A B C -> BetH A' B' C' ->
  CongH A B A' B' -> CongH B C B' C' ->
  CongH A C A' C'.
```

Congruence of angles In early editions of the *Foundations of Geometry*, Hilbert had taken pseudo-transitivity of congruence of angles as an axiom. Later Rosenthal has shown that this axiom can be derived from the others [26]. We used the later version of Hilbert's axioms. Note that, as we need transitivity of congruence in our proofs, we had to formalize Rosenthal's proofs. These proofs are rather technical, not very interesting in a pedagogical context, that is why Hartshorne chose to take the simpler, not minimal, axiom system of the earlier editions [16]. Richter also assumes the transitivity of congruence of angles. In our previous formalization, we defined the concept of angles ABC as three points A, B and C, with a proof that $A \neq B$ and $B \neq C$. We did not want to open the Pandora box of the dependent types [6] (types that depend on a proof), so we dropped the proofs from the angle type, and even made implicit the angle type. To be faithful to Hilbert, some non degeneracy conditions are added to ensure that angles are neither flat nor null. This makes the proof of the Tarski's five-segment axiom (A5) more involved.

We used a predicate of arity six for the congruence of angles:

```
CongaH :
  Point -> Point -> Point -> Point -> Point -> Point -> Prop;
```

```
CongH : Point -> Point -> Point -> Point -> Prop;
cong_permr : forall A B C D, CongH A B C D -> CongH A B D C;
cong_pseudo_transitivity :
  forall A B C D E F,
    CongH A B C D -> CongH A B E F -> CongH C D E F;
cong_existence :
  forall A B l M,
    A <> B -> Incid M l ->
    exists A', exists B', Incid A' l /\ Incid B' l /\
                          BetH A' M B' /\
                          CongH M A' A B /\ CongH M B' A B;
disjoint := fun A B C D => ~ exists P, BetH A P B /\ BetH C P D;
addition :
  forall A B C A' B' C',
    ColH A B C -> ColH A' B' C' ->
    disjoint A B B C -> disjoint A' B' B' C' ->
    CongH A B A' B' -> CongH B C B' C' ->
    CongH A C A' C';
```

Fig. 5: Formalization of Group III, part 1 : segment congruence axioms

As for the congruence of segments we need a permutation property about angle congruence:

```
congaH_permlr :
  forall A B C D E F, CongaH A B C D E F -> CongaH C B A F E D;
```

Our approach does not use rays, so we need to state that two angles represented by the same rays are congruent. This is the purpose of axiom `congaH_outH_congaH`. The predicate `outH P A B` states that B belongs to the ray PA:

```
outH :=
  fun P A B => BetH P A B \/ BetH P B A \/ (P <> A /\ A = B);
```

```
congaH_outH_congaH :
  forall A B C D E F A' C' D' F',
    CongaH A B C D E F ->
    outH B A A' -> outH B C C' -> outH E D D' -> outH E F F' ->
    CongaH A' B C' D' E F';
```

Recall that Hilbert's axiom III 4 states that:

Given an angle α, a ray h emanating from a point O and given a point P, not on the line generated by h, there is a unique ray h' emanating from O, such that the angle α' defined by (h, O, h') is congruent with α and such that every point inside α' and P are on the same side with respect to the line generated by h.

We simplified the formalization of Hilbert's axiom III 4, instead of considering every point inside the angle, our proof shows that it is sufficient to state that the point that defines the angle is on the same side as P. Hence, we can save the burden of defining what it means for a point to be inside an angle. Our version is also simpler than Scoot's one, which follows Hilbert's definition.

We say that two points are on the same side of a line, if there is a point P such that they are both on opposite sides wrt. P. The fact that two points are on opposite sides of a line is defined by the `cut` predicate of Group II.

```
hcong_4_existence :
  forall A B C O X P,
    ~ ColH P O X -> ~ ColH A B C ->
    exists Y, CongaH A B C X O Y /\ same_side' P Y O X;
hcong_4_uniqueness :
  forall A B C O P X Y Y',
    ~ ColH P O X  -> ~ ColH A B C ->
    CongaH A B C X O Y -> CongaH A B C X O Y' ->
    same_side' P Y O X -> same_side' P Y' O X ->
    outH O Y Y'
```

The full list of axioms is given in Fig. 6.

3.4 Group IV

In 2012, we had an axiom saying that given a line and a point there exists a parallel line through this point. Only the uniqueness of the parallel should be assumed as the existence can be derived from other axioms. The uniqueness axiom is depicted on Fig. 7.

4 Proving Tarski's Axioms

To prove Tarski's axioms we use a trick: we worked on both sides. We derived some lemmas in the context of Hilbert's axioms and we also proved Tarski's axioms in the context of a variant of these axioms. We introduced this variant of Tarski's axiom system inspired by the one adopted by Beeson in [2]. We added the following two properties as axioms: symmetry of betweenness and inner transitivity and we number them as in [32]. We modified Pasch's axiom to have a version which excludes the degenerate case when the triangle ABC is flat. This non degeneracy condition is crucial because obtaining the general case from Hilbert's version is not trivial.

Formally, we consider the following axioms:

$$
\begin{aligned}
A_{14} &:= \forall ABC,\ A{-}B{-}C \Rightarrow C{-}B{-}A \\
A_{15} &:= \forall ABCD,\ A{-}B{-}D \wedge B{-}C{-}D \Rightarrow A{-}B{-}C \\
A'_7 &:= \forall ABCPQ,\ A{-}P{-}C \wedge B{-}Q{-}C \wedge \\
&\quad A \neq P \wedge P \neq C \wedge B \neq Q \wedge Q \neq C \wedge \neg Col(ABC) \Rightarrow \\
&\quad \exists X, P{-}X{-}B \wedge Q{-}X{-}A
\end{aligned}
$$

```
CongaH :
  Point -> Point -> Point -> Point -> Point -> Point -> Prop;
conga_refl : forall A B C, ~ ColH A B C -> CongaH A B C A B C;
conga_comm : forall A B C, ~ ColH A B C -> CongaH A B C C B A;
congaH_permlr :
  forall A B C D E F, CongaH A B C D E F -> CongaH C B A F E D;
cong_5 :
  forall A B C A' B' C',
    ~ ColH A B C -> ~ ColH A' B' C' ->
    CongH A B A' B' -> CongH A C A' C' ->
    CongaH B A C B' A' C' ->
    CongaH A B C A' B' C';
same_side := fun A B l => exists P, cut l A P /\ cut l B P;
same_side' :=
  fun A B X Y =>
    X <> Y /\
    forall l, Incid X l -> Incid Y l -> same_side A B l;
outH :=
  fun P A B => BetH P A B \/ BetH P B A \/ (P <> A /\ A = B);
congaH_outH_congaH :
  forall A B C D E F A' C' D' F',
    CongaH A B C D E F ->
    outH B A A' -> outH B C C' -> outH E D D' -> outH E F F' ->
    CongaH A' B C' D' E F';
hcong_4_existence :
  forall A B C O X P,
    ~ ColH P O X -> ~ ColH A B C ->
    exists Y, CongaH A B C X O Y /\ same_side' P Y O X;
hcong_4_uniqueness :
  forall A B C O P X Y Y',
    ~ ColH P O X  -> ~ ColH A B C ->
    CongaH A B C X O Y -> CongaH A B C X O Y' ->
    same_side' P Y O X -> same_side' P Y' O X ->
    outH O Y Y'
```

Fig. 6: Formalization of Group III, part 2: angle congruence axioms

```
Para := fun l m => ~ exists X, Incid X l /\ Incid X m;
euclid_uniqueness :
  forall l P m1 m2,
    ~ Incid P l ->
    Para l m1 -> Incid P m1-> Para l m2 -> Incid P m2 ->
    EqL m1 m2
```

Fig. 7: Formalization of Group IV

We denote by $\mathcal{V}$ the following set of axioms:

$$\{A_1, A_2, A_3, A_4, A_5, A'_7, A_8, A_9, A_{14}, A_{15}\}.$$

Figure 8 provides an overview of the structure of the proof.

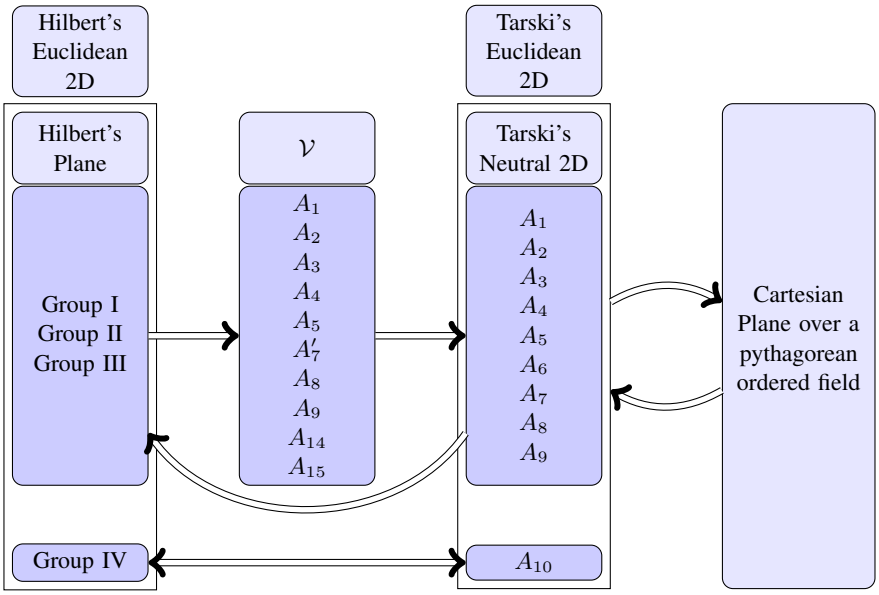


Fig. 8: Overview of the proofs

4.1 Hilbert Plane is equivalent to Tarski A1-A9

The proof is divided into two parts.

First, we proved that the variant of Tarski $\mathcal{V}$ implies Tarski's axioms. The fact that the axiom A_6 can be derived from $\mathcal{V}$ is Theorem 2.17 of Gupta's Ph.D. Thesis [15].

Second, we proved that $\mathcal{V}$ follows from Hilbert's axioms. Hilbert's betweenness relation is strict, whereas Tarski's one is not. Obviously, we defined Tarski's betweenness relation (`Bet`) from Hilbert's one (`BetH`) as:

```
Definition Bet A B C := BetH A B C \/ A = B \/ B = C.
```

Hilbert's congruence relation is defined only for non degenerate segments, whereas Tarski's one include the case of the null segment:

```
Definition Cong A B C D :=
      (CongH A B C D /\ A <> B /\ C <> D) \/ (A = B /\ C = D).
```

Axioms A_1, A_2, A_3, A_4, A_8 and A_{14} are already axioms in Hilbert or easy consequences of the axioms. A_{15} is a theorem in Hilbert which can be proved easily. Tarski's version of Pasch's axiom is stronger than Hilbert's one, because it provides information about the relative position of the points. We could recover the non degenerate case of Tarski's version of Pasch (A_7') using some betweenness properties and repeated applications of Hilbert's version of Pasch. The five-segment axiom requires a longer proof. The non degenerate case is a trivial consequence of the Side-Side-Side theorems and the fact that if two angles are congruent their supplements are congruent as well. Those theorems are proved by Hilbert as theorems 18 and 14. To prove these two theorems we had to formalize the proof of Hilbert's theorems 12, 15, 16 and 17 as well. Hilbert's proofs can be formalized without serious problem; we only had to introduce some lemmas about the relative position of two points and a line. For example, we had to prove that the same-side relation is transitive: if A and B are of the same side of l, and B and C are on the same side of l then A and C are also on the same side of l. As already noticed by Meikle and Scott, these lemmas, which are as difficult to prove as Hilbert's other theorems are completely implicit in Hilbert's prose. The non-obvious part of the proof has been the degenerate case of the five-segment axiom and the upper 2-dimensional axiom (A_9).

To prove the degenerate case of the five-segment axiom (when the point D belongs to the line AB), we had to prove that then D' also belongs to line $A'B'$. We also had to prove many degenerate cases which reduce to segment addition and subtraction. Segment subtraction can be deduced from uniqueness of segment construction and from addition. We give here only the proof of the key lemma (Lemma 2 below), assuming the following lemma:

Lemma 1.

$$A\!-\!B\!-\!C \wedge A'\!-\!B'\!-\!C' \wedge AC \equiv_H A'C' \wedge AB \equiv_H A'B' \Rightarrow A'\!-\!B'\!-\!C'$$

Lemma 2.

$$A\!-\!B\!-\!C \wedge AB \equiv_H A'B' \wedge BC \equiv_H B'C' \wedge AC \equiv_H A'C' \Rightarrow \mathrm{Col}\ A'\ B'\ C'$$

Proof. We will prove this lemma by contradiction so let us assume that B' does not belong to line $A'C'$. Let B'' be a point on $A'C'$ such that $A'B'' \equiv_H AB$. Let C'' a point such that $B'C'' \equiv_H BC$ and $A'\!-\!B'\!-\!C''$ (Fig. 9). So the triangle $B'C''C'$ is isosceles in B'. Then Hilbert's theorem 12 lets us prove that $B'C'C'' \cong B'C''C'$. By Lemma 1, we have that $A'\!-\!B''\!-\!C'$. We can derive $B''C' \equiv_H BC$ by subtraction and then $A'C' \equiv_H A'C''$ by addition. Therefore triangle $C'A'C''$ is isosceles in A', hence Hilbert's theorem 12 implies that $A'C''C' \cong A'C'C''$. By transitivity of angle congruence (Hilbert's theorem 19), we know that $B'C'C'' \cong A'C'C''$. Finally we obtain a contradiction as the uniqueness of angle construction and the fact that B' and C'' are on the same side of $C'C''$ let us prove that $C'B'$ is the same ray as $C'A'$.

The last axiom we need to prove is the upper 2-dimensional axiom. The proof is not completely straightforward because we do not assume decidability of intersection of lines: we can not distinguish cases to know if two lines intersect or not.

Let us first prove two useful lemmas.

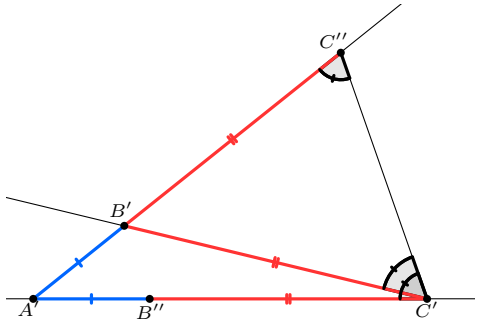


Fig. 9: Proof of Lemma 2

Lemma 3. *If two points A and B are not collinear with two points X and Y, then either they are one the same side of the line XY or they are on opposite sides of this line.*

Proof. First, one can construct a point C, such that points A and C are on the opposite side of the line XY. Therefore, there exists a point I collinear with X and Y. If A, B and I are collinear, then either $A\text{-}B\text{-}I$, $B\text{-}A\text{-}I$ or $A = B$ and then A and B are on the same side of line XY or $A\text{-}I\text{-}B$ and then A and B are on opposite sides of line XY, as, neither A nor B can be equal to I since they would then be collinear with X and Y. Finally, if A, B and C are not collinear, then Pasch's axiom lets us conclude the proof.

In order to prove Lemma 5 we first prove a particular case which will be used repeatedly throughout this proof.

Lemma 4. *If three distinct points A, B and C are equidistant from two different points P and Q, then, assuming that A is collinear with P and Q, these points are collinear.*

Proof. We know that neither B or C are collinear with P and Q because if they were then they would be equal to A, thus obtaining a contradiction. Therefore, using the previous lemma, either B and C are on opposite sides or on the same side of line PQ.

- If they are on opposite sides of line PQ, then we name I the point of intersection between this line and the segment $\overline{BC}$. If we can prove that A is equal to point I we will be done. To do this we just have to prove that I is equidistant from P and Q. Using Hilbert's theorem 18, we know that the angles $\widehat{PAB}$ and $\widehat{QAB}$ are equal. Then Hilbert's theorem 12 lets us prove that I is equidistant from P and Q.
- If they are on the same side of line PQ, the previous lemma states that either P and Q are on opposite sides or on the same side of line BC.
  - If they are on the same side, the uniqueness axioms let us prove that they are equal, therefore obtaining a contradiction.

- If they are on opposite side, then we name I the point of intersection between this line and the line PQ. Without loss of generality, let us consider that B is between C and I (if they are equal then A, B and C are trivially collinear). Using Hilbert's theorems 14 and 18, we know that the angles $\widehat{PBI}$ and $\widehat{QBI}$ are equal. Then Hilbert's theorem 12 let us prove that I is equidistant from P and Q. Therefore A is equal to I and we are done.

Lemma 5. *If three distinct points A, B and C are equidistant from two different points P and Q, then these points are collinear.*

Proof. We just have to consider the case where neither A, B or C are collinear with P and Q since otherwise the previous lemma lets us conclude. We know that either at least two of these points are on opposite sides of the line PQ or all the points are one the same side of this line.

- If they are on opposite sides of line PQ, then we name I the point of intersection between this line and the segment formed by the points which are on opposite sides of this line. As in the previous lemma we can prove that I is equidistant from P and Q. Then apply the previous lemma twice we know that A, B and I as well as A, C and I are collinear and the transitivity of collinearity allows us conclude.
- If they are on the same side of line PQ, either P and Q are on opposite sides or on the same side of line AB.
  - If they are on the same side, the uniqueness axioms let us prove that they are equal, therefore obtaining a contradiction.
  - If they are on opposite side, then we name I the point of intersection between this line and the line PQ. As in the previous lemma we can prove that I is equidistant from P and Q. Then apply the previous lemma twice we know that A, B and I as well as A, C and I are collinear and the transitivity of collinearity allows us conclude.

4.2 Euclidean Hilbert Plane is equivalent to Tarski A1-A10

The fact that Playfair's postulate can be derived from Tarski's version of the postulate appears in Chapter 12 of [27], that we have formalized previously. The reverse implication and many other equivalence results have been described in [9]. For this implication, we have to assume the decidability of intersection of line: given two lines either they intersect or they do not.

5 Conclusion

We have obtained a formal proof that Hilbert's geometric axioms implies Tarski's ones. It is interesting to note, that to prove this result we had to correct the axioms we had given in [12], both removing unnecessary axioms and adding some properties. The connection between Hilbert's axioms and Tarski's ones allows us to obtain all the results we had proven in the context of Tarski's axiom system, in particular the culminating result which is the arithmetization of geometry.

In our experience, manipulating Hilbert's axioms is more tedious than Tarski's ones. As the axioms involve both points and lines (and planes in 3D), a single result can be stated in many different ways, and it requires formally a lot of administrative work to convert assumptions about lines into assumptions about points and vice-versa. Even if this administrative work can be partially automated, using for example the techniques proposed by Scott and Fleuriot [31], we have the impression that the mechanization of geometry based on a single primitive type as in Tarski's axiom system is more convenient. Moreover, Hilbert chose to exclude some degenerate cases from its axioms: angles are never null nor full, segments are never degenerate, this makes the proofs more tedious than in Tarski's setting. Still, Hilbert's axiom system has the advantage that axioms are grouped and each group can serve to develop a significant part of geometry, whereas in a development of geometry based on Tarski's axioms, one has to use both the betweenness and congruence axioms very early. Our formalization of Hilbert's axiom system avoids the concept of sets of points, segments, rays and angles. Our current formalization can hence be considered less faithful to the original than the one we presented in 2012, but this new formalization produces axioms which are both clearer and simpler to use.

Perspectives

This work completes the formalization of a non trivial part of Hilbert's book. But, the formalization could be completed. Among the chapters which have not been formalized, we can cite the one about meta-mathematical results, and the one about arithmetization of hyperbolic geometry. It would also be useful to extend the formalization to 3D.

It would be also interesting to generate readable proofs from our Coq formalization to obtain a mechanized book about foundations of geometry.

We leave to future work the proof that our version of Hilbert's axiom is equivalent to a version in which angles are defined as a pair of rays.

Availability

The full Coq development is available here (release 2.1.0):
`http://geocoq.github.io/GeoCoq/`

References

1. Michael Beeson. Constructive geometry. In *Proceedings of the Tenth Asian Logic Colloquium*. World Scientific, 2010.
2. Michael Beeson. A constructive version of Tarski's geometry. *Annals of Pure and Applied Logic*, 166(11):1199–1273, 2015.
3. Michael Beeson and Larry Wos. OTTER proofs of theorems in Tarskian geometry. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, Vienna, Austria, July 19-22, 2014, Proceedings*, volume 8562 of *Lecture Notes in Computer Science*, pages 495–510. Springer, 2014.

4. Michael Beeson and Larry Wos. Finding Proofs in Tarskian Geometry. *Journal of Automated Reasoning*, submitted, 2015.
5. Michel Beeson. Proving Hilbert's axioms in Tarski geometry. http://www.michaelbeeson.com/research/papers/TarskiProvesHilbert.pdf, 2014.
6. Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development, Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.
7. George D Birkhoff. A set of postulates for plane geometry, based on scale and protractor. *Annals of Mathematics*, 33:329–345, 1932.
8. Pierre Boutry, Gabriel Braun, and Julien Narboux. From Tarski to Descartes: Formalization of the Arithmetization of Euclidean Geometry. In *The 7th International Symposium on Symbolic Computation in Software (SCSS 2016)*, EasyChair Proceedings in Computing, page 15, Tokyo, Japan, March 2016.
9. Pierre Boutry, Julien Narboux, and Pascal Schreck. Parallel postulates and decidability of intersection of lines: a mechanized study within Tarski's system of geometry. submitted, July 2015.
10. Pierre Boutry, Julien Narboux, and Pascal Schreck. A reflexive tactic for automated generation of proofs of incidence to an affine variety. October 2015.
11. Pierre Boutry, Julien Narboux, Pascal Schreck, and Gabriel Braun. Using small scale automation to improve both accessibility and readability of formal proofs in geometry. In Francisco Botana and Pedro Quaresma, editors, *Preliminary Proceedings of the 10th International Workshop on Automated Deduction in Geometry (ADG 2014), 9-11 July 2014, University of Coimbra, Portugal*, pages 1–19, Coimbra, Portugal, July 2014.
12. Gabriel Braun and Julien Narboux. From Tarski to Hilbert. In Tetsuo Ida and Jacques Fleuriot, editors, *Post-proceedings of Automated Deduction in Geometry 2012*, volume 7993 of *LNCS*, pages 89–109, Edinburgh, United Kingdom, September 2012. Springer.
13. Gabriel Braun and Julien Narboux. A synthetic proof of Pappus' theorem in Tarski's geometry. *Journal of Automated Reasoning*, 2015. accepted, revision pending.
14. Christophe Dehlinger, Jean-François Dufourd, and Pascal Schreck. Higher-Order Intuitionistic Formalization and Proofs in Hilbert's Elementary Geometry. In *Automated Deduction in Geometry*, volume 2061 of *Lecture Notes in Computer Science*, pages 306–324. Springer, 2001.
15. Haragauri Narayan Gupta. *Contributions to the axiomatic foundations of geometry*. PhD thesis, University of California, Berkley, 1965.
16. Robin Hartshorne. *Geometry : Euclid and beyond*. Undergraduate texts in mathematics. Springer, 2000.
17. David Hilbert. *Foundations of Geometry (Grundlagen der Geometrie)*. Open Court, La Salle, Illinois, 1960. Second English edition, translated from the tenth German edition by Leo Unger. Original publication date, 1899.
18. David Hilbert. *Les fondements de la géométrie*. Dunod, Paris, jacques gabay edition, 1971. Edition critique avec introduction et compléments préparée par Paul Rossier.
19. Timothy James McKenzie Makarios. *A Mechanical Verification of the Independence of Tarski's Euclidean Axiom*. PhD thesis, Victoria University of Wellington, 2012. Master Thesis.
20. Laura Meikle and Jacques Fleuriot. Formalizing Hilbert's Grundlagen in Isabelle/Isar. In *Theorem Proving in Higher Order Logics*, volume 2758 of *Lecture Notes in Computer Science*, pages 319–334. Springer, 2003.
21. Richard S Millman and George D Parker. *Geometry, A Metric Approach with Models*. Springer Science & Business Media, 1991.
22. E.E. Moise. *Elementary Geometry from an Advanced Standpoint*. Addison-Wesley, 1990.

23. Julien Narboux. Mechanical Theorem Proving in Tarski's Geometry. In Francisco Botana Eugenio Roanes Lozano, editor, *Post-proceedings of Automated Deduction in Geometry 2006*, volume 4869 of *Lecture Notes in Computer Science*, pages 139–156, Pontevedra, Spain, 2008. Springer.
24. Moritz Pasch. *Vorlesungen über neuere Geometrie*. Springer, 1976.
25. William Richter. Formalizing Rigorous Hilbert Axiomatic Geometry Proofs in the Proof Assistant Hol Light.
26. Artur Rosenthal. Vereinfachungen des Hilbertschen Systems der Kongruenzaxiome. *Mathematische Annalen*, 71(2):257–274, June 1911.
27. Wolfram Schwabhäuser, Wanda Szmielew, and Alfred Tarski. *Metamathematische Methoden in der Geometrie*. Springer-Verlag, Berlin, 1983.
28. Phil Scott. *Mechanising Hilbert's Foundations of Geometry in Isabelle*. Master thesis, Edinburgh, 2008.
29. Phil Scott and Jacques Fleuriot. Compass-free Navigation of Mazes. In James H. Davenport and Fadoua Ghourabi, editors, *SCSS 2016. 7th International Symposium on Symbolic Computation in Software Science*, volume 39 of *EPiC Series in Computing*, pages 143–155. EasyChair, 2016.
30. Phil Scott and Jacques D. Fleuriot. An Investigation of Hilbert's Implicit Reasoning through Proof Discovery in Idle-Time. In *Automated Deduction in Geometry - 8th International Workshop, ADG 2010, Munich, Germany, July 22-24, 2010, Revised Selected Papers*, pages 182–200, 2010.
31. Phil Scott and Jacques D. Fleuriot. A Combinator Language for Theorem Discovery. In *Intelligent Computer Mathematics - 11th International Conference, AISC 2012, 19th Symposium, Calculemus 2012, 5th International Workshop, DML 2012, 11th International Conference, MKM 2012, Systems and Projects, Held as Part of CICM 2012, Bremen, Germany, July 8-13, 2012. Proceedings*, pages 371–385, 2012.
32. Alfred Tarski and Steven Givant. Tarski's System of Geometry. *The bulletin of Symbolic Logic*, 5(2):175–214, June 1999.

Mass points, Bézier Curves and Conics: a Survey

Lionel Garnier[1] and Jean-Paul Bécar[2]

[1] LE2I, UMR CNRS 6306, University of Burgundy, 21000 Dijon, France
[2] LAMAV-CGAO, CNRS 2956, Le Mont-Houy, 59313 Valenciennes cedex 9

Abstract. It is well known that rational quadratic Bézier curves define conics. The use of massic points permits one to define a semi-conic in the Euclidean plane. Moreover, from a given quadratic Bézier curve, we determine the properties of the underlying conic using only one theorem which leads to a very simple program.

Keywords: Massic points; rational quadratic Bézier curves; invarients of conics

1 Introduction

In the computer aided domain, the use of rational Bezier curves play an important role for geometric modelling. Among these curves, the second order curves define a conic arc (see [1–4]). Such a curve is considered as the set of barycentres of weighted points. However, some problems occur for the representation of non bounded parabolic or hyperbolic arcs. One answer is the use of weighted points and vectors in the same space (see [5]). In that space, vectors get a null weight. Thus Bezier curves can be generalised. In the second order rational curves, by the use of some homographic parameter changes, all features of conics such as focus and directrix can be determined (see [4]).

Our method offers a new approach of others method developed by G. Albrecht (see [6]), R. Goldman and W. Wang (see [7]), E. Lee (see [8]), J. Sánchez-Reyes (see [9]). Using one of these methods, the construction of a Bézier curve is not possible in a non-Euclidean space. Some other examples of these models, in the space of spheres, can be found in [10–12].

Moreover, some researchers use homogeneous coordinates to model a semi-circle (see [13–15]), but A. Piegl noticed (see [14]) that: "A point in projective space is what mathematicians call an equivalence class. This means that $\overrightarrow{P_j}(x_j, y_j, z_j, 0)$ and $\overrightarrow{P_j}^*(\alpha x_j, \alpha y_j, \alpha z_j, 0)$ are two representations of the same point in projective space. However, substituting $\overrightarrow{P_j}$ and $\overrightarrow{P_j}^*$ into Eq (7.34) (see [14]) clearly results in two different curves." We can choose a vector using perpendicular conditions and pseudo-metric conditions to determine a Bézier curve which models a given conic seen as a circle (for an adequate non-degenerate indefinite quadratic form) (see [11]). Some applications of the representations, on the space of spheres, of Dupin cyclides using rational quadratic Bézier curves with mass points can be found in (see [16–19]). In this space, the use of the homogeneous coordinates are not possible.

This paper is organised as follows: Section 2 presents the mass points and rational quadratic Bézier curves where their control points are mass points. In section 3, we determine the elements of a conic defined by a rational quadratic Bézier curve in the Euclidean plane. The last section draws some conclusions.

In this paper, $(O;\overrightarrow{\imath_0};\overrightarrow{\jmath_0})$ designates the initial direct reference frame in the usual Euclidean affine plane $\mathcal{P}$. Let $\overrightarrow{\mathcal{P}}$ be the set of the vector plane.

2 About mass points

Let I be a finite subset of $\mathbb{N}$. Consider a family $(A_i;\omega_i)_{i\in I}$ of weighted points in the affine plane satisfy

$$\sum_{i\in I}\omega_i \neq 0$$

The centre of mass (or barycentre) G of the weighted points $(A_i;\omega_i)_{i\in I}$ is the unique point in $\mathcal{P}$ defined by

$$\sum_{i\in I}\omega_i\overrightarrow{GA_i} = \overrightarrow{0} \tag{1}$$

A mass point is a couple (M,m) such that: if m is equal to 0, $\overrightarrow{M}$ is a vector belonging to the vector plane $\overrightarrow{\mathcal{P}}$ otherwise M is a point belonging to the affine plane $\mathcal{P}$. So, a mass point is a weighted point or a vector. J.C. Fiorot (see [5, 20, 21]) defined the set of mass points as

$$\widetilde{\mathcal{P}} = (\mathcal{P}\times\mathbb{R}^*)\cup\left(\overrightarrow{\mathcal{P}}\times\left\{\overrightarrow{0}\right\}\right)$$

What is the barycentre of weighted points when the sum of the weights vanishes? It is well known that this sum is a vector in the vector plane $\overrightarrow{\mathcal{P}}$ and for example:

$$(A;B)\in\mathcal{P}\Rightarrow \mathtt{bar}\left\{(A;-1)\,;(B;1)\right\}\ \text{is:}\quad \overrightarrow{AB}\in\overrightarrow{\mathcal{P}}$$

where $\mathtt{bar}\left\{(M;\omega)\,;(N;\mu)\right\}$ designates the barycentre of the weighted points $(M;\omega)$ and $(N;\mu)$. Moreover, if $(A_i,\omega_i)_{i\in I}$ is a family of weighted points with $\sum_{i\in I}\omega_i = 0$, then we cannot define the barycentre of this family, but we obtain the vector

$$\overrightarrow{u} = \sum_{i\in I}\omega_i\overrightarrow{MA_i}$$

for any point M in $\mathcal{P}$ (the vector $\overrightarrow{u}$ does not depend on the point M).

To construct a vector space, we have to define some operations in $\widetilde{\mathcal{P}}$ using analytical geometric methods and then the addition, noted $\oplus$, is defined as follow:

- $\omega\neq 0\Longrightarrow (M;\omega)\oplus(N;-\omega) = \left(\omega\overrightarrow{NM};0\right)$;

- if we have $\omega\mu\left(\omega+\mu\right)\neq 0$, then
  $(M;\omega)\oplus(N;\mu)=\left(\texttt{bar}\left\{(M;\omega);(N;\mu)\right\};\omega+\mu\right)$
- $(\overrightarrow{u};0)\oplus(\overrightarrow{v};0)=(\overrightarrow{u}+\overrightarrow{v};0)$;
- $\omega\neq 0\Longrightarrow(M;\omega)\oplus(\overrightarrow{u};0)=\left(\mathcal{T}_{\frac{1}{\omega}\overrightarrow{u}}(M);\omega\right)$ where $\mathcal{T}_{\overrightarrow{w}}$ is the translation of $\mathcal{P}$ of vector $\overrightarrow{w}$.

In the same way, on the space $\widetilde{\mathcal{P}}$, we define the multiplication by a scalar, noted $\odot$, as follow:

- $\alpha\neq 0\Longrightarrow\alpha\odot(M;\omega)=(M;\alpha\omega)$
- $\omega\neq 0\Longrightarrow 0\odot(M;\omega)=\left(\overrightarrow{0};0\right)$
- $\alpha\odot(\overrightarrow{u};0)=(\alpha\overrightarrow{u};0)$

Let G_2 be the midpoint of the segment $[MN]$. We have

$$\begin{aligned}(M;1)\oplus(N;1)&=(M;2)\oplus(M;-1)\oplus(N;1)\\&=(M;2)\oplus\left(\overrightarrow{MN};0\right)\\&=\left(\mathcal{T}_{\frac{1}{2}\overrightarrow{MN}}(M);2\right)=(G_2;2)\end{aligned}$$

and we generalise the associativity of the barycentre to the set of vectors.

Notice that we do not use homogeneous coordinates, we do not use quotient space and we can define any metric or pseudo-metric on $\overrightarrow{\mathcal{P}}$ (see [16, 19]). The notion of barycentre is lost using homogeneous coordinates. For more details on this space, the reader can refer to books of Fiorot and Jeannin (see [5, 20]).

2.1 Rational quadratic Bézier curves in $\widetilde{\mathcal{P}}$

Let us recall the definition of the quadratic Bernstein polynomials

$$B_0(t)=(1-t)^2, B_1(t)=2t(1-t), B_2(t)=t^2, t\in[0,1] \tag{2}$$

Now, we can define a rational quadratic Bézier curve having three control mass points $(P_0;\omega_0)$, $(P_1;\omega_1)$ and $(P_2;\omega_2)$, Definition 1.

Definition 1 *: Rational quadratic Bézier curve (BR curve) in $\widetilde{\mathcal{P}}$*

Let ω_0, ω_1 and ω_2 be three real numbers. Let $(P_0;\omega_0)$, $(P_1;\omega_1)$ and $(P_2;\omega_2)$ be three mass points in $\widetilde{\mathcal{P}}$, these points are not colinear.

Define two sets $I=\{i|\omega_i\neq 0\}$ and $J=\{i|\omega_i=0\}$

Define the function ω_f from $[0;1]$ to $\mathbb{R}$ as follows

$$\omega_f(t)=\sum_{i\in I}\omega_i\times B_i(t) \tag{3}$$

A mass point $(M;\omega)$ or $(\overrightarrow{u};0)$ belongs to the quadratic Bézier curve defined by the three control mass points $(P_0;\omega_0)$, $(P_1;\omega_1)$ and $(P_2;\omega_2)$, if there is a real t_0 in $[0;1]$ such that:

- *if* $\omega_f(t_0) \neq 0$ *then we have*

$$\overrightarrow{OM} = \frac{1}{\omega_f(t_0)} \left(\sum_{i \in I} \omega_i B_i(t_0) \overrightarrow{OP_i} \right) + \frac{1}{\omega_f(t_0)} \left(\sum_{i \in J} B_i(t_0) \overrightarrow{P_i} \right) \tag{4}$$

- *if* $\omega_f(t_0) = 0$ *then we have*

$$\overrightarrow{u} = \sum_{i \in I} \omega_i B_i(t_0) \overrightarrow{OP_i} + \sum_{i \in J} B_i(t_0) \overrightarrow{P_i} \tag{5}$$

To simplify the rest of this paper, we introduce:

Notation 1 *:*

The notation $BR\{(P_0;\omega_0);(P_1;\omega_1);(P_2;\omega_2)\}$ *designates a BR curve with the following mass points of control* $(P_0;\omega_0)$, $(P_1;\omega_1)$ *and* $(P_2;\omega_2)$.

The notation $\overline{BR\{(P_0;\omega_0);(P_1;\omega_1);(P_2;\omega_2)\}}$ *designates the proper conic containing the curve* $BR\{(P_0;\omega_0);(P_1;\omega_1);(P_2;\omega_2)\}$.

2.2 Some properties

If $J = \emptyset$, we do not modify the Bézier curve if we multiply all the weights by a non-zero real. What happens when the control points are weighted points and/or vectors? We can state:

Lemma 1 *Let* $(P_0;\omega_0)$, $(P_1;\omega_1)$ *and* $(P_2;\omega_2)$ *be three mass points* $\widetilde{\mathcal{P}}$.

Let λ *be a non-zero real. If* $\sum_{i \in I} \omega_i B_i(t_0)$ *is not equal to* 0, *we have*

$$BR\left\{(P_i;\omega_i)_{i\in I};\left(\overrightarrow{P_j};0\right)_{j\in J}\right\} = BR\left\{(P_i;\lambda\omega_i)_{i\in I};\left(\lambda\overrightarrow{P_j};0\right)_{j\in J}\right\} \tag{6}$$

Proof: left to the reader. ■

If $J = \emptyset$, the reduced discriminant of the denominator[3], Formula (3), is

$$\Delta' = \omega_1^2 - \omega_2\omega_0 \tag{7}$$

and we can state the following fundamental result:

- ⋆ if $\omega_1^2 - \omega_2\omega_0 = 0$ then the denominator has one and only one root, the curve is a parabolic arc;
- ⋆ if $\omega_1^2 - \omega_2\omega_0 > 0$ then the denominator has two distinct roots, the curve is a hyperbolic arc;
- ⋆ if $\omega_1^2 - \omega_2\omega_0 < 0$ then the denominator does not vanish, the curve is an elliptical arc.

[3] Formula (3) is the denominator in the definition of the BR curve.

We can note that we obtain the same result if $J \neq \emptyset$, see the proposition 5.1.6 of (see [5]).

We can note w.l.o.g. (see Lemma 1), one of the weights can be equal to 1. If ω_0 is not equal to 0, we choose $\omega_0 = 1$, else, we choose $\omega_1 = 1$, and we can characterise the type of the conic from the mass points of the BR curve, see Table 1.

Conic	Three weighted points	Points and vectors
Parabola	$(P_0;1)$, $(P_1;\omega)$ $\left(P_2;\omega^2\right)$	$(P_0;1)$, $\left(\overrightarrow{P_1};0\right)$ $\left(\overrightarrow{P_2};0\right)$
Ellipse	$(P_0;1)$, $(P_1;\omega_1)$, $(P_2;\omega_2)$, $\omega_2 > \omega_1^2$	$(P_0;1)$, $\left(\overrightarrow{P_1};0\right)$ $(P_2;1)$
Hyperbola	$(P_0;1)$, $(P_1;\omega_1)$ $(P_2;\omega_2)$, $\omega_2 < \omega_1^2$	$(P_0;1)$, $\left(\overrightarrow{P_1};0\right)$ $(P_2;-1)$
		$\left(\overrightarrow{P_0};0\right)$, $(P_1;1)$ and $\left(\overrightarrow{P_2};0\right)$

Table 1. Types of conics defined by Bézier curves with control mass points, see Figure 1.

3 New conic parameters determination

In this section, we compute the new control mass points from the previous control mass points and a homographic function h : we compute mass points to obtain a representation of the semi-conic using a Bézier curve, see Figure 1.

3.1 Homographic function

Theorem 1

Let $(P_0;\omega_0)$, $(P_1;\omega_1)$ and $(P_2;\omega_2)$ be three mass points.

Let a, b, c and d be four reals and h be the homographic function defined by

$$\begin{aligned} h : \overline{\mathbb{R}} &\longrightarrow \overline{\mathbb{R}} \\ u &\longmapsto \frac{a\,(1-u)+bu}{c\,(1-u)+du} \quad \textit{with} \quad \begin{vmatrix} a & b \\ c & d \end{vmatrix} \neq 0 \end{aligned} \tag{8}$$

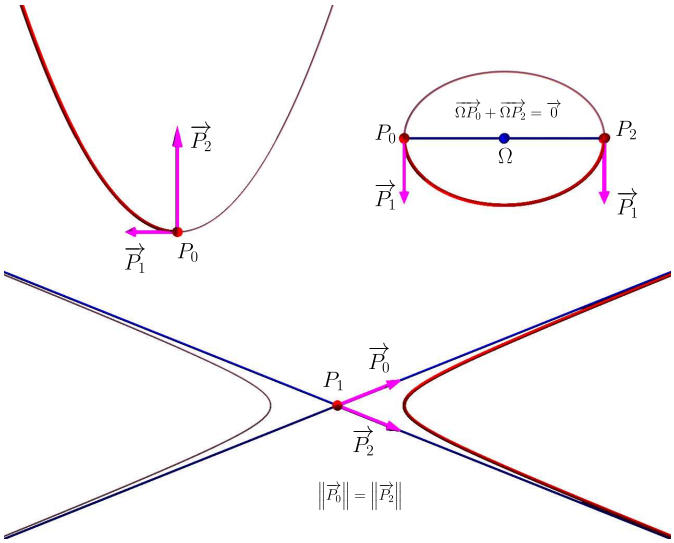


Fig. 1. Standard representation of a semi-conic using a BR curve with three control mass points.

Let $(Q_0;\varpi_0)$, $(Q_1;\varpi_1)$ *and* $(Q_2;\varpi_2)$ *be the mass points defined as follow*

$$\begin{cases} (Q_0;\varpi_0) = (c-a)^2 \odot (P_0;\omega_0) \oplus 2a\,(c-a) \odot (P_1;\omega_1) \\ \qquad\qquad \oplus\, a^2 \odot (P_2;\omega_2) \\ (Q_1;\varpi_1) = (c-a)\,(d-b) \odot (P_0;\omega_0) \\ \qquad\qquad \oplus\, (bc-2ab+ad) \odot (P_1;\omega_1) \\ \qquad\qquad \oplus\, ab \odot (P_2;\omega_2) \\ (Q_2;\varpi_2) = (d-b)^2 \odot (P_0;\omega_0) \oplus 2b\,(d-b) \odot (P_1;\omega_1) \\ \qquad\qquad \oplus\, b^2 \odot (P_2;\omega_2) \end{cases} \tag{9}$$

Let γ_P *be the map of* $BR\{(P_0;\omega_0);(P_1;\omega_1);(P_2;\omega_2)\}$. *Let* γ_Q *be the map of* $BR\{(Q_0;\varpi_0);(Q_1;\varpi_1);(Q_2;\varpi_2)\}$. *Then* $\gamma_Q = \gamma_P \circ h$ *and*

$$\overline{BR\{(P_0;\omega_0);(P_1;\omega_1);(P_2;\omega_2)\}} = \overline{BR\{(Q_0;\varpi_0);(Q_1;\varpi_1);(Q_2;\varpi_2)\}} \tag{10}$$

Proof: see [4]. ■

In the rest of this paper, we will just write the result using Formula (9) but the reader keeps in mind that $\gamma_Q = \gamma_P \circ h$ and that Formula (10) is always true.

We note, using Theorem 10 with $a = 0$, $c = \dfrac{1}{\sqrt{\omega_0}}$ and $d = b = \dfrac{1}{\sqrt{\omega_2}}$, that only the weights of P_0 and P_2 change. Moreover, if ω_0 and ω_2 are two positive real numbers, if $\omega_1 \neq 0$, we have

$$BR\left\{(P_0;\omega_0);(P_1;\omega_1);(P_2;\omega_2)\right\} = BR\left\{(P_0;1);\left(P_1;\frac{\omega_1}{\sqrt{\omega_0\omega_2}}\right);(P_2;1)\right\} \tag{11}$$

else, (if $\omega_1 = 0$), we have

$$BR\left\{(P_0;\omega_0);\left(\overrightarrow{P_1};0\right);(P_2;\omega_2)\right\} = BR\left\{(P_0;1);\left(\frac{1}{\sqrt{\omega_0\omega_2}}\overrightarrow{P_1};0\right);(P_2;1)\right\} \tag{12}$$

Thus we generalise to vectors the formulae given by Farin (see [22]). In the rest of this paper, if the first control point is a weighted point, we take $\omega_0 = 1$.

3.2 The parabola case

From Formula (7), the $BR\left\{(P_0;1);(P_1;\omega_1);(P_2;\omega_2)\right\}$ is a parabola iff we have

$$\omega_2 = \omega_1^2$$

Theorem 2

Let $(P_0;1)$, $(P_1;\omega)$ and $\left(P_2;\omega^2\right)$ be three control weighted points of a BR curve. We distinguish two cases.

1. *If $\omega = 1$. Let h be defined by $h(u) = \dfrac{u}{1-u}$.*
   Then, the point $(P_0;1)$ is unchanged and we obtain

$$\begin{cases} \left(\overrightarrow{U_1};0\right) = \quad \left(\overrightarrow{P_0P_1};0\right) \\ \left(\overrightarrow{U_2};0\right) = \left(\overrightarrow{P_1P_0} + \overrightarrow{P_1P_2};0\right) \end{cases} \tag{13}$$

2. *If $\omega \notin \{0;1\}$. Let h be defined by $h(u) = \dfrac{u}{1-\omega}$.*
   Then, the point $(P_0;1)$ is unchanged and we obtain

$$\begin{cases} \left(\overrightarrow{U_1};0\right) = \left(\dfrac{\omega}{1-\omega}\overrightarrow{P_0P_1};0\right) \\ \left(\overrightarrow{U_2};0\right) = \left(\left(\dfrac{\omega}{1-\omega}\right)^2\left(\overrightarrow{P_1P_0} + \overrightarrow{P_1P_2}\right);0\right) \end{cases} \tag{14}$$

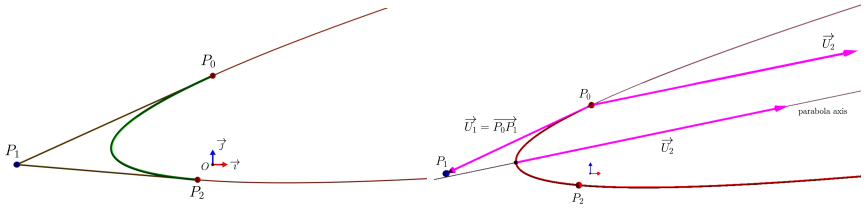
Proof: This proof is obtained using Theorem 10 with $a = 0$, $b = 1$, $c = 1$ and $d = 0$ for the first case and $a = 0$, $b = \frac{1}{1-\omega}$, $c = 1$ and $d = 1$ for the second case. ■

In Theorem 2, if t_0 belongs to $[0,1[$, the mass point of the BR curve is a weighted point $(M, B_0(t_0))$ defined by

$$\overrightarrow{OM} = \frac{1}{B_0(t_0)}\left(B_0(t_0)\overrightarrow{OP_0} + B_1(t_0)\overrightarrow{U_1} + B_2(t_0)\overrightarrow{U_2}\right) \tag{15}$$

whereas, for $t_0 = 1$, the mass point of the BR curve is the vector $\left(\overrightarrow{U_2};0\right)$.

Figures 2 show the effect of Theorem 2 with $(P_0;1) = ((0.;6);1)$, $(P_1;\omega) = ((-13.;0);1)$ and $\left(P_2;\omega^2\right) = ((-1;-1);1)$.



(a) (b)

Fig. 2. Illustration of Theorem 2 with $\omega = 1$. (a): the initial BR curve with three control weighted points. (b): the new BR curve with three control mass points (one weighted point and two vectors).

The second step of this work is done using the following theorem.

Theorem 3 *Let* $(P_0;1)$, $\left(\overrightarrow{U_1};0\right)$ *and* $\left(\overrightarrow{U_2};0\right)$ *be three control mass points of a BR curve such that*

$$\overrightarrow{U_1} \bullet \overrightarrow{U_2} \neq 0$$

Let h *be defined by*

$$h(u) = \frac{ru}{(1-u)+ru} \text{ with } r = \frac{1}{\left\|\overrightarrow{U_2}\right\|^2}\overrightarrow{U_1} \bullet \overrightarrow{U_2} \tag{16}$$

Then, the point $(P_0;1)$ *is unchanged and we obtain*

$$\begin{cases} \left(\overrightarrow{V_1};0\right) = \left(r\overrightarrow{U_1};0\right) \\ \left(\overrightarrow{V_2};0\right) = \left(r^2\overrightarrow{U_2};0\right) \end{cases} \tag{17}$$

Proof: This proof is obtained using Theorem 10 with $a = 0$, $d = b = r$ and $c = 1$. ■

Figure 3 shows an application of Theorem 3 with $(P_0;1) = ((0.;6);1)$, $(P_1;1) = ((-13.;0);1)$ and $(P_2;1) = ((-1;-1);1)$.

Now we can state:

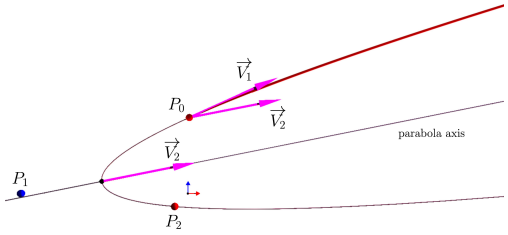


Fig. 3. Illustration of Theorem 3 with $\omega = 1$.

Theorem 4 *Let* $(P_0;1)$, $\left(\overrightarrow{V_1};0\right)$ *and* $\left(\overrightarrow{V_2};0\right)$ *be three control mass points of a BR curve such that*

$$\overrightarrow{V_1} \bullet \overrightarrow{V_2} \neq 0$$

Let h *be defined by* $h(u) = \dfrac{-(1-u)+u}{u}$.
Then, the point $(P_0;1)$ *is unchanged and we obtain*

$$\begin{cases} (Q_0;1) &= \left(\mathcal{T}_{\overrightarrow{V_2}-2\overrightarrow{V_1}}(P_0);1\right) \\ \left(\overrightarrow{W_1};0\right) &= \left(\overrightarrow{V_1}-\overrightarrow{V_2};0\right) \end{cases} \tag{18}$$

Proof: This proof is obtained using Theorem 1 with $a = -1$, $b = d = 1$ and $c = 0$. ■

Figure 4 gives an illustration of Theorem 4 with $(P_0;1) = ((0.;6);1)$, $(P_1;1) = ((-13.;0);1)$ and $(P_2;1) = ((-1;-1);1)$.

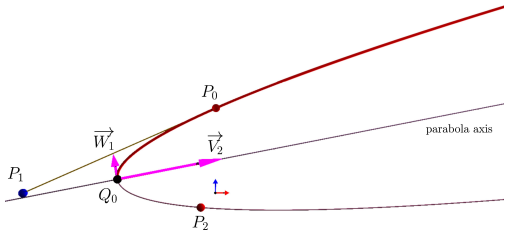


Fig. 4. Illustration of Theorem 4 with $\omega = 1$.

We can state a fundamental remark. The vectors defined in Formula (18) verify

$$\overrightarrow{W_1} \bullet \overrightarrow{V_2} = 0 \tag{19}$$

Now we can determine the parabola invariants.

Theorem 5 *:*

Let $\omega \in \mathbb{R}^*$. Let $(P_0;1)$, $(P_1;\omega)$ and $\left(P_2;\omega^2\right)$ be three control mass points of a BR curve. Let $BR\left\{(Q_0;1);\left(\overrightarrow{W_1};0\right);\left(\overrightarrow{V_2};0\right)\right\}$, be the BR curve obtained using Theorems 2, 3 and 4.

Then, $\overline{BR\{(P_0;1);(P_1;\omega);(P_2;\omega^2)\}}$ is a parabola $\mathtt{P}$, its vertex is Q_0, its equation is

$$Y = \frac{\left\|\overrightarrow{V_2}\right\|^2}{4\left\|\overrightarrow{W_1}\right\|}X^2$$

in the orthonormal reference frame $\left(Q_0; \dfrac{1}{\left\|\overrightarrow{W_1}\right\|}\overrightarrow{W_1}; \dfrac{1}{\left\|\overrightarrow{V_2}\right\|}\overrightarrow{V_2}\right)$.

Proof: see [4]. ■

The vector $\overrightarrow{W_1}$ is tangent to the parabola at Q_0 whereas the focal axis is defined by Q_0 and $\overrightarrow{V_2}$.

Figure 5 shows two arcs of the same parabola: $BR\{(P_0;1),(P_1;1),(P_2;1)\}$ on one hand and $BR\{(Q_0;1),\left(\overrightarrow{W_1};0\right),\left(\overrightarrow{V_2};0\right)\}$ on the other hand. This curve offers the computation of the elements of the parabola using Theorem 5.

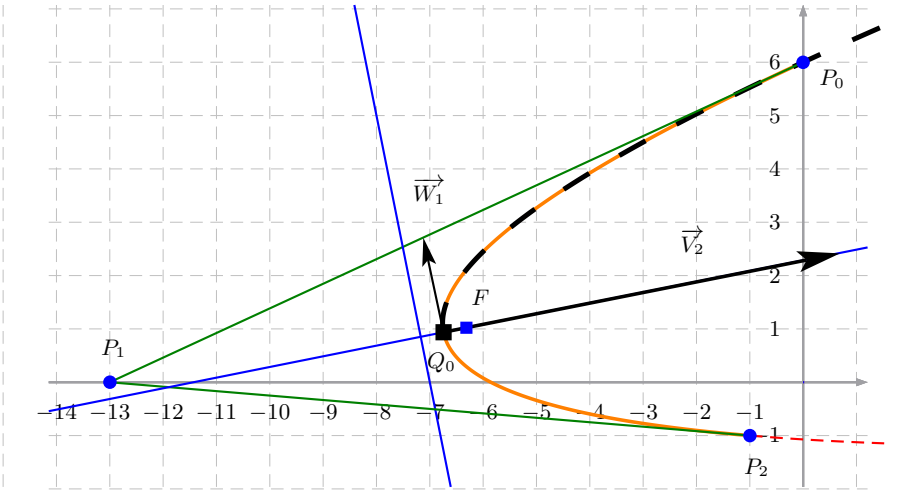


Fig. 5. Two arcs of the same parabola defined with two BR curves with control mass points $(P_0;1)$, $(P_1;1)$ and $(P_2;1)$ (in orange) on one hand $(Q_0;1)$, $\left(\overrightarrow{W_1};0\right)$ and $\left(\overrightarrow{V_2};0\right)$ (in black) on the other hand. The first mass point is the weighted point $(Q_0;1)$ whereas the last mass point is the vector $\left(\overrightarrow{V_2};0\right)$.

3.3 The ellipse case

In this section, the weights satisfy the condition $\omega_1^2 - \omega_2\omega_0 < 0$ and w.l.o.g., we can suppose $\omega_0 = 1$ and $\omega_2 > 0$. Moreover, using Theorem 10 with $a = 0$, $b = d = -1$ and $c = 1$, we can suppose $\omega_1 \geq 0$. In fact, using the multiplication of a mass point by a scalar, we transform the mass point $(P_1;\omega_1)$ into the mass point $-1 \odot (P_1;\omega_1)$. So a weighted point $(P_1;\omega_1)$ (resp. vector $\left(\overrightarrow{P_1};0\right)$) becomes $(P_1;-\omega_1)$ (resp. vector $\left(-\overrightarrow{P_1};0\right)$). Now we can begin the change of the control mass points and we can state:

Theorem 6 *Let $(P_0;1)$, $(P_1;\omega_1)$, $\omega_2 > \omega_1^2 \neq 0$, and $(P_2;\omega_2)$ be three control weighted points of a BR curve. Let h be defined by*

$$h(u) = \frac{ru}{(1-u)+(1+r)u} \text{ with } r = -\frac{1}{\omega_1} \tag{20}$$

Then, the point $(P_0;1)$ is unchanged and we obtain

$$\begin{cases} \left(\overrightarrow{U_1};0\right) = \left(\overrightarrow{P_1P_0};0\right) \\ (P_3;\varpi_3) = \left(\texttt{bar}\left\{(P_0;1);(P_1;-2)\left(P_2;\frac{\omega_2}{\omega_1^2}\right)\right\};-1+\frac{\omega_2}{\omega_1^2}\right) \end{cases} \tag{21}$$

Proof: This proof is obtained using Theorem 1 with $a = 0$, $b = r$, $c = 1$ and $d = 1 + r$. ■

We note that the last weight is positive and the midpoint O_0 of $[P_0P_3]$ is the centre of the ellipse.

Figure 6 shows the effect of Theorem 6 with $(P_0;1) = ((0.;6);1)$, $(P_1;1) = \left((-13.;0);\frac{1}{2}\right)$ and $(P_2;1) = ((-1;-1);1)$ (the curve is in green). The intermediate weighted point is transformed into a vector, the last weighted point is modified into another weighted point. The obtained curve is in red.

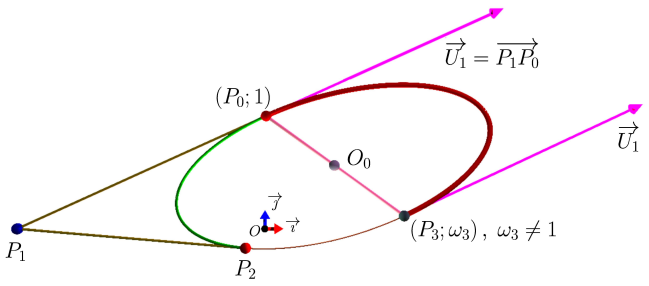


Fig. 6. Illustration of Theorem 6.

The next theorem offers the weights of the weighted points to be equal to 1.

Theorem 7 *:*

Consider $BR\left\{(P_0;1);\left(\overrightarrow{U_1};0\right);(P_3;\varpi_3)\right\}$ *obtained using Theorem 6.*

Let h *be defined by* $h(u)=\dfrac{u}{\sqrt{\omega_3}(1-u)+u}$.

Then, the point $(P_0;1)$ *is unchanged, the weight of* P_3 *becomes* 1 *and we obtain*

$$\overrightarrow{V_1}=\frac{1}{\sqrt{\omega_3}}\overrightarrow{U_1} \tag{22}$$

Proof We use Theorem 10 with $a=0$, $d=b=1$ and $c=\sqrt{\omega_3}$, and then we use Formula (6) with $\lambda=\frac{1}{\omega_3}$.■

Figure 7 shows the effect of Theorem 7 with $(P_0;1)=((0.;6);1)$, $(P_1;1)=\left((-13.;0);\frac{1}{2}\right)$ and $(P_3;1)=((-1;-1);1)$.

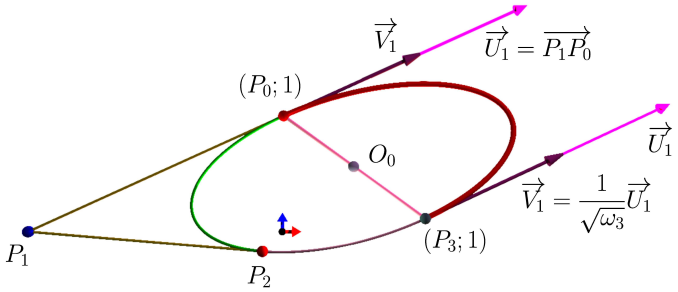


Fig. 7. Illustration of Theorem 7.

Let k_1 and k_2 be the reals defined by

$$\begin{cases} k_1 = \overrightarrow{P_0P_3}\bullet\overrightarrow{V_1} \\ k_2 = \overrightarrow{V_1}^2-\dfrac{1}{4}P_0P_3^2 \end{cases} \tag{23}$$

If $k_1=0$ and $k_2=0$ then the $BR\left\{(P_0;1);\left(\overrightarrow{V_1};0\right);(P_3;1)\right\}$ is a circular arc, see [4]. In the rest of this section, we do not consider this case. The following theorem permits us to model a semi-ellipse.

Theorem 8 *Let* $(P_0;1)$, $\left(\overrightarrow{V_1};0\right)$ *and* $(P_3;1)$ *be the control mass points of a Bézier curve, obtained using Theorem 7.*

If $k_2=0$, *we take* $\theta=\frac{\pi}{8}$, *else we take* $\theta=\dfrac{1}{4}\arctan\left(\dfrac{k_1}{k_2}\right)$.

We define two points P_4 and P_5 as follow

$$\begin{cases} P_4 = \mathtt{bar}\left\{\left(P_0;\cos^2(\theta)\right);\left(P_3;\sin^2(\theta)\right)\right\} \\ P_5 = \mathtt{bar}\left\{\left(P_3;\cos^2(\theta)\right);\left(P_0;\sin^2(\theta)\right)\right\} \end{cases}$$

Let h be defined by

$$h(u) = \frac{\sin(\theta)(1-u)+\cos(\theta)u}{(\cos(\theta)+\sin(\theta))(1-u)+(\cos(\theta)-\sin(\theta))u} \tag{24}$$

Then, we obtain

$$\begin{cases} (Q_0;1) &= \left(\mathcal{T}_{\sin(2\theta)\overrightarrow{V_1}}(P_4);1\right) \\ \left(\overrightarrow{W_1};0\right) &= \left(\dfrac{\sin(2\theta)}{2}\overrightarrow{P_0P_3}+\cos(2\theta)\overrightarrow{V_1};0\right) \\ (Q_2;1) &= \left(\mathcal{T}_{-\sin(2\theta)\overrightarrow{V_1}}(P_5);1\right) \end{cases} \tag{25}$$

Proof: The reader can use Theorem 1 with $a=\sin(\theta)$, $b=\cos(\theta)$, $c=a+b$ and $d=b-a$. ■

We note that the mass point defined by Formula (25) verify

$$\overrightarrow{Q_0Q_2}\bullet\overrightarrow{W_1}=0$$

and now we can determine the invariants of the ellipse and we can state:

Theorem 9 *Invariants of the ellipse* $\overline{BR\left\{(Q_0;1);\left(\overrightarrow{W_1};0\right);(Q_2;1)\right\}}$

Let us consider the $BR\left\{(Q_0;1);\left(\overrightarrow{W_1};0\right);(Q_2;1)\right\}$ *obtained, using Theorems 6, 7 and 8, from* $BR\{(P_0;1);(P_1;\omega_1);(P_2;\omega_2)\}$ *with* $\overrightarrow{Q_0Q_2}\bullet\overrightarrow{W_1}=0$.

We distinguish two cases.

- $\overrightarrow{W_1}^2-\dfrac{1}{4}Q_0Q_2^2<0$, *we take*

$$\overrightarrow{\imath}=\frac{1}{a}\frac{1}{2}\overrightarrow{Q_0Q_2} \text{ with } a=\left\|\frac{1}{2}\overrightarrow{Q_0Q_2}\right\|$$

$$\overrightarrow{\jmath}=\frac{1}{b}\overrightarrow{W_1} \text{ with } b=\left\|\overrightarrow{W_1}\right\|$$

- $\overrightarrow{W_1}^2-\dfrac{1}{4}Q_0Q_2^2>0$, *we take*

$$\overrightarrow{\imath}=\frac{1}{a}\overrightarrow{W_1} \text{ with } a=\left\|\overrightarrow{W_1}\right\|$$

$$\overrightarrow{\jmath}=\frac{1}{b}\frac{1}{2}\overrightarrow{Q_0Q_2} \text{ with } b=\left\|\frac{1}{2}\overrightarrow{Q_0Q_2}\right\|$$

Let O_0 be the midpoint of the segment $[Q_0Q_2]$. In the orthonormal reference frame $(O_0;\overrightarrow{\imath};\overrightarrow{\jmath})$, the equation of the ellipse $\overline{BR\left\{(Q_0;1);\left(\overrightarrow{W_1};0\right);(Q_2;1)\right\}}$ *is*

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$$

Proof: see [4]. ■

Figure 8 shows two arcs of the same ellipse defined by two BR curves of control mass points $(P_0;1)$, $\left(P_1;\frac{1}{2}\right)$ and $(P_2;1)$ on one hand $(Q_0;1)$, $\left(\overrightarrow{W_1};0\right)$ and $(Q_2;1)$ on the other hand. This last curve permits us to compute the invariants of the ellipse using Theorem 9. The points Q_0 and Q_2 are two vertices on one of the two axis (the focal axis or the non-focal axis).

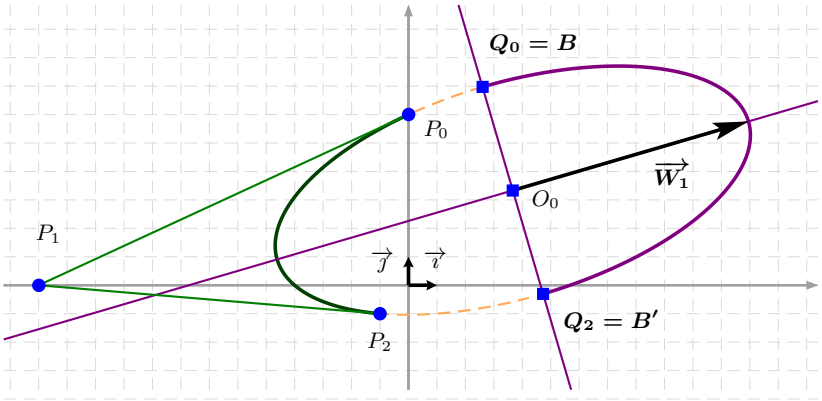


Fig. 8. Two arcs of an ellipse. From a BR curve of control weighted points $(P_0;1)$, $\left(P_1;\frac{1}{2}\right)$ and $(P_2;1)$, we obtain a semi-ellipse defined by the BR curve of control mass points $(Q_0;1)$, $\left(\overrightarrow{W_1};0\right)$ and $(Q_2;1)$.

3.4 The hyperbola case

In this section, we take $\omega_0 = 1$ and then we have the following condition

$$\omega_1^2 - \omega_2 > 0$$

During the first step, we change the weighted points $(P_0;1)$ and $(P_2;\omega_2)$ into vectors having the directions of the asymptotes of the hyperbola. The denominator of $BR\{(P_0;1);(P_1;\omega_1);(P_2;\omega_2)\}$ is

$$B_0(t) + \omega_1 B_1(t) + \omega_2 B_2(t) = (1 - 2\omega_1 + \omega_2)\,t^2 + 2(\omega_1 - 1)\,t + 1 \qquad (26)$$

and we have two cases.

- If $1 - 2\omega_1 + \omega_2 \neq 0$ then the degree of the denominator equals 2 and we have two roots t_1 and t_2 which define two vectors;
- If $1 - 2\omega_1 + \omega_2 = 0$ then the degree of the denominator equals 1 then we have one root t_1 which defines a vector. The second vector is obtained as the barycentre of $(P_0;1)$, $(P_1;-2\omega_1)$ and $(P_2;\omega_2)$ (the sum of the weights is equal to 0).

Theorem 10 *Let $(P_0;1)$, $(P_1;\omega_1)$ and $(P_2;\omega_2)$ be three control weighted points of a BR curve. We have two cases.*

- *If $1-2\omega_1+\omega_2\neq 0$, the equation given in Formula (26) has two roots t_1 and t_2 defined by*

$$t_1=\frac{1-\omega_1+\sqrt{\omega_1^2-\omega_2}}{1-2\omega_1+\omega_2},\quad t_2=\frac{1-\omega_1-\sqrt{\omega_1^2-\omega_2}}{1-2\omega_1+\omega_2}$$

Let h be defined by $h(u)=t_1(1-u)+t_2u$. Then

$$\begin{cases} \overrightarrow{U_0} & = B_0(t_1)\overrightarrow{\Omega P_0}+\omega_1 B_1(t_1)\overrightarrow{\Omega P_1}+\omega_2 B_2(t_1)\overrightarrow{\Omega P_2} \\ (Q_1;\varpi_1) & = \left(\mathtt{bar}\{(P_0;\alpha_0);(P_1;\alpha_1);(P_2;\alpha_2)\};\dfrac{2\left(\omega_2-\omega_1^2\right)}{(1-2\omega_1+\omega_2)}\right) \\ \overrightarrow{U_2} & = B_0(t_2)\overrightarrow{\Omega P_0}+\omega_1 B_1(t_2)\overrightarrow{\Omega P_1}+\omega_2 B_2(t_2)\overrightarrow{\Omega P_2} \end{cases} \quad (27)$$

with

$$(\alpha_0,\alpha_1,\alpha_2)=\left((1-t_1)(1-t_2),\ \omega_1(t_2-2t_1t_2+t_1),\ t_1t_2\omega_2\right)$$

- *If $1-2\omega_1+\omega_2=0$, the root of the equation given in Formula (26) is*

$$t_0=\frac{1}{2(1-\omega_1)}$$

Let h be defined by $h(u)=\dfrac{t_0(1-u)+u}{(1-u)}$. Then

$$\begin{cases} \overrightarrow{U_0} & = B_0(t_0)\overrightarrow{\Omega P_0}+\omega_1 B_1(t_0)\overrightarrow{\Omega P_1}+\omega_2 B_2(t_0)\overrightarrow{\Omega P_2} \\ (Q_1;\varpi_1) & = (\mathtt{bar}\{(P_0;\alpha_0);(P_1;\alpha_1);(P_2;\alpha_2)\};\omega_1-1) \\ \overrightarrow{U_2} & = \overrightarrow{\Omega P_0}-2\omega_1\overrightarrow{\Omega P_1}+\omega_2\overrightarrow{\Omega P_2} \end{cases} \quad (28)$$

with

$$(\alpha_0,\alpha_1,\alpha_2)=t_0-1,\ \omega_1(1-2t_0),\ t_0\omega_2$$

Proof: This proof is obtained using Theorem 10 with $a=t_1$, $b=t_2$, $c=1$ and $d=1$ for the first case and $a=t_0$, $b=c=1$ and $d=0$ for the second case. ■

Figure 9 shows the effect of the first case of Theorem 10 with $(P_0;1)=\left(\left(\sqrt{2};1\right);1\right)$, $(P_1;\omega_1)$ and $(P_2;1)=\left(\left(2;-\sqrt{3}\right);1\right)$ with

$$(P_1;\omega_1)=\left(\left(\frac{\sqrt{2}}{2}-\frac{\sqrt{3}}{3};1+\left(1+\sqrt{3}\right)\left(1-\sqrt{2}\right)\right);\frac{\sqrt{2\sqrt{3}+4\sqrt{2}+2}}{2}\right)$$

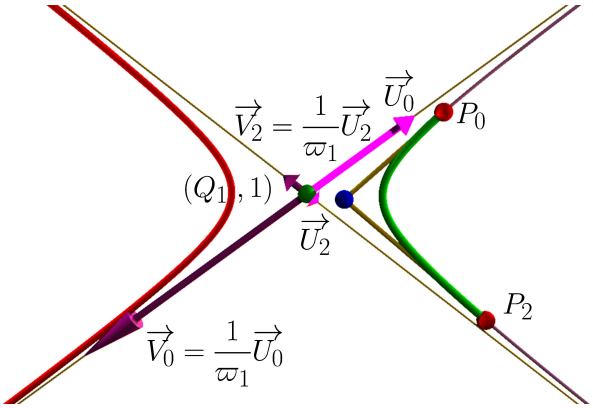


Fig. 9. Illustration of the first case of Theorem 10.

Let us define $\overrightarrow{V_0} = \frac{1}{\varpi_1}\overrightarrow{U_0}$ and $\overrightarrow{V_2} = \frac{1}{\varpi_1}\overrightarrow{U_2}$. The new control mass points of the Bézier curve are $\left(\overrightarrow{V_0}; 0\right)$, $(Q_1; 1)$ and $\left(\overrightarrow{V_2}; 0\right)$. The centre of the hyperbola is Q_1, its weight is equal to 1. In the next theorem, we obtain two vectors having the same norm.

Theorem 11 *Let* $\left(\overrightarrow{V_0}; 0\right)$, $(Q_1; 1)$ *and* $\left(\overrightarrow{V_2}; 0\right)$ *be the control mass points of a BR curve obtained using previous theorems. Let* h *be defined by*

$$h(u) = \frac{ru}{\frac{1}{r}(1-u) + ru} \text{ with } r = \sqrt[4]{\frac{\left\|\overrightarrow{V_0}\right\|}{\left\|\overrightarrow{V_2}\right\|}} \tag{29}$$

Then

$$\begin{cases} (Q_0; \varpi_0) = \left(\overrightarrow{W_0}; 0\right) = \left(r^4\ \overrightarrow{V_0}; 0\right) \\ (Q_2; \varpi_0) = \left(\overrightarrow{W_2}; 0\right) = \left(r^4\ \overrightarrow{V_2}; 0\right) \end{cases} \tag{30}$$

with the fundamental relation

$$\left\|\overrightarrow{W_0}\right\| = \left\|\overrightarrow{W_2}\right\| \tag{31}$$

Proof : From Theorem 10 with $a = 0$, $d = b = r$ and $c = \frac{1}{r}$. ■
Figure 10 shows the effect of Theorem 11 from the example given in Figure 9.

The last step consists in computing the hyperbola invariants.

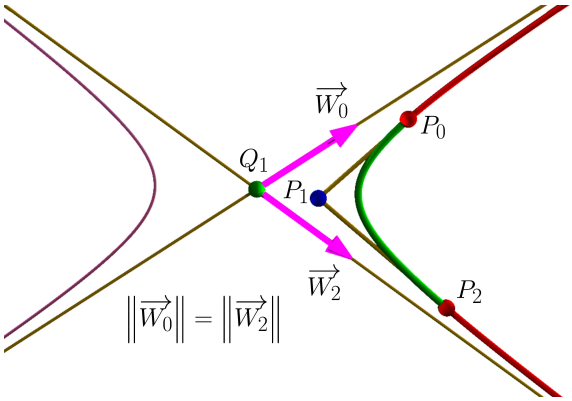


Fig. 10. Illustration of Theorem 11 from Figure 9.

Theorem 12 *Invariants of a hyperbola*
The curve $BR\left\{\left(\overrightarrow{W_0};0\right);(Q_1;1);\left(\overrightarrow{W_2};0\right)\right\}$ *is obtained using Theorems 10 and 11.*
Define

$$\overrightarrow{\imath} = \frac{1}{a}\frac{1}{2}\left(\overrightarrow{W_0}+\overrightarrow{W_2}\right) \quad \textit{with} \quad a = \frac{\left\|\overrightarrow{W_0}+\overrightarrow{W_2}\right\|}{2} \tag{32}$$

and

$$\overrightarrow{\jmath} = \frac{1}{b}\frac{1}{2}\left(\overrightarrow{W_0}-\overrightarrow{W_2}\right) \quad \textit{with} \quad b = \frac{\left\|\overrightarrow{W_0}-\overrightarrow{W_2}\right\|}{2} \tag{33}$$

Then $\overline{BR\left\{\left(\overrightarrow{W_0};0\right);(Q_1;1);\left(\overrightarrow{W_2};0\right)\right\}}$ *is the hyperbola of centre* Q_1*, of equation*

$$x \times y = \frac{\left\|\overrightarrow{W_0}\right\|^2}{4}$$

in the orthonormal reference frame $\left(Q_1;\frac{1}{\left\|\overrightarrow{W_0}\right\|}\overrightarrow{W_0};\frac{1}{\left\|\overrightarrow{W_2}\right\|}\overrightarrow{W_2}\right)$*, of equation*

$$\frac{x^2}{a^2}-\frac{y^2}{b^2}=1$$

in the orthonormal reference frame $(Q_1;\overrightarrow{\imath};\overrightarrow{\jmath})$*.*

Proof: see [4]. ■
We remark that the vertices A and A' of the hyperbola satisfy

$$\overrightarrow{Q_1A} = \frac{1}{2}\left(\overrightarrow{W_0}+\overrightarrow{W_2}\right) = -\overrightarrow{Q_1A'} \tag{34}$$

Figure 11 shows two arcs of the same hyperbola. The first arc is modelled by a BR curve, its control weighted points are $(P_0;1)$, $\left(P_1;\frac{\sqrt{2\sqrt{3}+4\sqrt{2}+2}}{2}\right)$ and $(P_2;1)$, Figure 9. The second arc is a branch of the hyperbola, the control mass points of the BR curve are $\left(\overrightarrow{W_0};0\right)$, $(Q_1;1)$ and $\left(\overrightarrow{W_2};0\right)$. From this curve, using Theorem 12, we can determine the elements of this hyperbola.

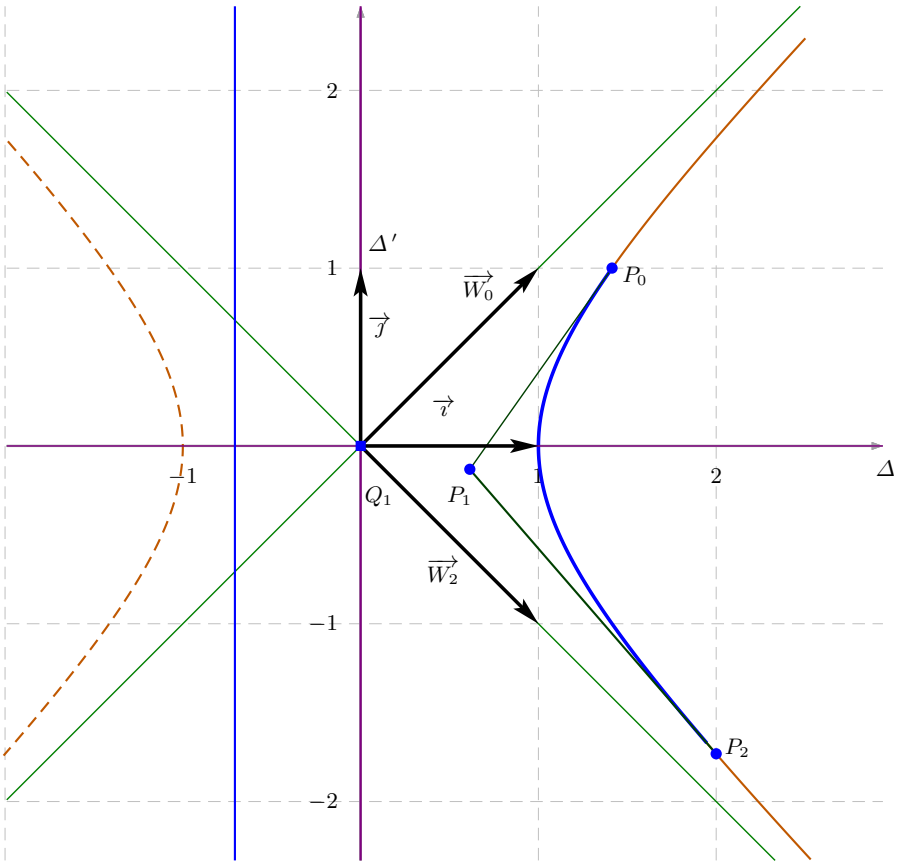


Fig. 11. A connected arc of hyperbola mode led by a BR curve of control mass points $(P_0;1)$, $\left(P_1;\frac{\sqrt{2\sqrt{3}+4\sqrt{2}+2}}{2}\right)$ and $(P_2;1)$, a BR curve of control mass points $\left(\overrightarrow{W_0};0\right)$, $(Q_1;1)$ and $\left(\overrightarrow{W_2};0\right)$ which models a branch of the Hyperbola.

4 Conclusion

We have presented a survey of mass points: in the same space, we aggregate the affine points and the vectors. Moreover, we can generalise the notion of barycentre and its associativity to the vectors. We have modelled non-degenerate conics using rational quadratic Bézier curves. In the Euclidean plane, starting with these kinds of curves, we have computed the elements of these conics (parabola, ellipse and hyperbola).

To complete this discussion, the reader may consult (see [23]) which gives a modern application of conics in the context of curve approximations and (see [16, 24]) where the authors compute the characteristic circles of a Dupin cyclide modelled in the space of spheres.

Acknowledgements

The authors warmly thank Prof R. Goldman from Rice University in Houston for his review and his helpful comments.

References

1. Bézier, P.: Courbe et surface. 2ème edn. Volume 4. Hermès, Paris (1986)
2. Casteljau, P.D.: Mathématiques et CAO. Volume 2 : formes à pôles. Hermes (1985)
3. Garnier, L.: Mathématiques pour la modélisation géométrique, la représentation 3D et la synthèse d'images. Ellipses (2007) ISBN : 978-2-7298-3412-8.
4. Bécar, J.P.: Forme (BR) des coniques et de leurs faisceaux. PhD thesis, Université de Valenciennes et de Hainaut-Cambrésis, LIMAV (1997)
5. Fiorot, J.C., Jeannin, P.: Courbes et surfaces rationnelles. Volume RMA 12. Masson (1989)
6. Albrecht, G.: Mathematical methods for curves and surfaces. Vanderbilt University, Nashville, TN, USA (2001) 15–24
7. Goldman, R.N., Wang, W.: Using invariants to extract geometric characteristics of conic sections from rational quadratic parameterizations. Int. J. Comput. Geometry Appl. **14** (2004) 161–187
8. Lee, E.: The rational Bézier representation for conics. In (ed.), G.F., ed.: In Geometric Modeling, Algorithms and New Trends. SIAM, Philadelphia (1985) 3–19
9. Sánchez-Reyes, J.: Characteristics of conic segments in Bézier form. In: International Conference on Innovative Methods in Product Design, Venise, Italy (2011) 231–234
10. Garnier, L., Druoton, L., Langevin, R.: Subdivisions itératives d'arcs d'ellipses et d'hyperboles et application à la visualisation de cyclides de Dupin. Revue Electronique Francophone d'Informatique Graphique **6** (2012) 1–36
11. AFIG, ed.: Inversions de coniques à centres vues comme des cercles, Université de Limoges (2013)
12. Garnier, L., Druoton, L., Bécar, J.P.: Points massiques, espace des sphères et "hyperbole". In: G.T.M.G. 2015, Poitiers (2015) http://gtmg2015.conference.univ-poitiers.fr/programme_details.

13. Versprille, K.J.: Computer-aided Design Applications of the Rational B-spline Approximation Form. PhD thesis, Syracuse, NY, USA (1975) AAI7607690.
14. Piegl, L., Tiller, W.: The NURBS book. Monographs in visual communication. Springer (1995)
15. Farin, G.: From conics to nurbs: A tutorial and survey. IEEE Comput. Graph. Appl. **12** (1992) 78–86
16. Druoton, L., Garnier, L., Langevin, R.: Iterative construction of Dupin cyclide characteristic circles using non-stationary Iterated Function Systems (IFS). Computer-Aided Design **45** (2013) 568–573 Solid and Physical Modeling 2012, Dijon.
17. Garnier, L., Bécar, J.P., Druoton, L.: Subdivisions de courbes de Bézier quadratiques, Lyon, France, Université de Lyon, LIRIS (2015) http://liris.cnrs.fr/afig2015/?page_id=939#sthash.IbJxJhLu.dpuf.
18. Garnier, L., Bécar, J.P., Morin, G., Fuchs, L.: Une application de l'espace des sphres : détermination des sphères de Dandelin, Lyon, France, Université de Lyon, LIRIS (2015) http://liris.cnrs.fr/afig2015/?page_id=939#sthash.IbJxJhLu.dpuf.
19. Garnier, L., Bécar, J.P., Druoton, L.: Surfaces canal et courbes de bézier rationnelles quadratiques, Dijon, France, Université de Bourgogne, Le2i (2016)
20. Fiorot, J.C., Jeannin, P.: Courbes splines rationnelles, applications à la CAO. Volume RMA 24. Masson (1992)
21. Goldman, R.: On the algebraic and geometric foundations of computer graphics. ACM Trans. Graph. **21** (2002) 52–86
22. Farin, G.: NURBS from Projective Geometry to Pratical Use. 2 edn. A K Peters, Ltd (1999) ISBN 1-56881-084-9.
23. Albrecht, G., Bécar, J.P., Farin, G.E., Hansford, D.: On the approximation order of tangent estimators. Computer Aided Geometric Design **25** (2008) 80–95
24. Garnier, L., Druoton, L.: Constructions, dans l'espace des sphères, de carreaux de cyclides de Dupin à bords circulaires. Revue Electronique Francophone d'Informatique Graphique **7** (2013) 17–40

A New Formalization of Origami in Geometric Algebra

Tetsuo Ida[1], Jacques Fleuriot[2], and Fadoua Ghourabi[3]

[1] University of Tsukuba, Tsukuba, Japan.
ida@cs.tsukuba.ac.jp
[2] University of Edinburgh, Edinburgh, U.K.
jdf@inf.ed.ac.uk
[3] Ochanomizu University, Tokyo, Japan.
ghourabi.fadoua@ocha.ac.jp

Abstract. We present a new formalization of origami modeling and theorem proving using a geometric algebra. We formalize in Isabelle/HOL a geometric algebra $\mathcal{G}_3$) to treat origamis in both 2D and 3D physical space. We define $\mathcal{G}_3$ as a type class of Isabelle/HOL. The objects in $\mathcal{G}_3$ are multivectors. We prove that the co-datatype of a multivector is an element instance of the type class $\mathcal{G}_3$. We prove by Isabelle/HOL a large number of identities and equations that hold in $\mathcal{G}_3$. With $\mathcal{G}_3$ we then reformulate Huzita's elementary origami folds in equations of multivectors. They are translated to the equations of the polynomial ring. By solving the equations, we can construct origamis computationally, and furthermore prove geometric properties of the constructed origami. We show an example of trisecting an arbitrary angle by origami using $\mathcal{G}_3$.

1 Introduction

We show an application of a geometric algebra to computational origami. There are several definitions of geometric algebra. We take the one expounded by Hestenes [1]. We call for geometric algebra, as we are investigating the extension of the computational origami system Eos [2] to deal with modeling and verifying 3D origamis more systematically. There are two main reasons for this.

Firstly, a uniform treatment of 3D and 2D models within the same system is important to us since many origamis can be made in a single flat 2D plane, whereas folding papers apparently involve the notion of superpositions of flat planes. This mere fact shows the need for the concept of 3D. In other words, modeling origami entirely in 3D destroys the simplicity of origami geometric problems, whereas to model origamis entirely in 2D is too restrictive, as we are mostly interested in the 3D shapes of origami as the result of construction.

Secondly, the algebraic and geometric structures used for computational purposes are often boiled down to the collection of simple symbolic forms by the translations via the Cartesian coordinate system. Whether the resulting ones are polynomials or matrices, for example, the geometric meanings that originally pertain to geometric objects are often lost in the translation. Here, we

need adequate abstraction layers between the geometric language of our daily use and the language for the computational purposes.

We organize the rest of the paper as follows. In Section 2, we present the geometric algebra (to be abbreviated to GA, hereafter), but only as much as is needed for the computational origami construction and verification, with the use of new notations that facilitate the reading of identities involved. As we are treating formulas of the GA in two computer languages, some deviation from the usual mathematical notations, and complication in expressing the formulas are inevitable. To simplify our explanation, we remove type information from the GA formulas of this paper. From the outset of Section 2, we introduce the notion of a multivector, i.e. the element of GA, together with its operations. Then, we formalize the multivector and the operations on it using Isabelle/HOL. In Section 3, we briefly introduce Huzita's elementary fold operations since we use them in origami geometry. In Section 4, we describe Huzita's fold in GA. In Section 5, we show a simple example to illustrate our method of geometric theorem proving using GA. We give an example of trisecting an arbitrary angle by origami. Although it is a typical 2D origami problem, indeed had been one of the famous impossibility problems in classical Euclidean geometry, it can serve as a non-trivial illustration of GA. In Section 6, we briefly discuss research works in the past two decades whose methodology and goal we share, i.e. geometric theorem proving. In Section 7, we summarize our results and point out the directions for further research.

2 Geometric algebra

2.1 Objects and operations

We are going to define a geometric algebra $\mathcal{G}$, following Hestenes' definition[1] with the application to origami in mind. In particular, we would like to describe $\mathcal{G}_3$ as a formal system to reason about the geometric objects and their operations on them in 2D and 3D physical spaces.

Elements of $\mathcal{G}_3$ are called *multivectors*. A multivector consists of the following components: a real number (we also call it a scalar, to be consistent with the usage in the linear algebra), a vector, a bivector, and a trivector. They are sometimes written as 0-vector, 1-vector, 2-vector, and 3-vector, respectively. The vectors and bivectors are three dimensional. A vector is a triplet of scalars, generally written as (x_1, x_2, x_3). We fix the basis of the 3D vector space to be the set of the orthonormal vectors σ_1, σ_2, and σ_3. By convention, they form a right-handed set. In traditional 3D geometry, they correspond to the unit vectors along x-axis, y-axis, and z-axis, respectively. The basis of the 3D bivector (linear) space is the set of the orthonormal bivectors $\mathfrak{i}_1$, $\mathfrak{i}_2$, and $\mathfrak{i}_3$. They are related to σ_1, σ_2, and σ_3 as $\mathfrak{i}_1 = \sigma_2 * \sigma_3$, $\mathfrak{i}_2 = \sigma_3 * \sigma_1$, and $\mathfrak{i}_3 = \sigma_1 * \sigma_2$. The multiplication operator * is one of the operators of the geometric algebra and will be explained in Sub-section 2.2 in conjunction with type `mvec` of multivectors. It is worth mentioning, however, that unlike usual arithmetics, * is non-commutative. The 3D trivector space is a one-dimensional space. The basis of the 3D trivector

(linear) space is the set $\{\ \iota\ \}$. The trivector ι is a unit trivector and is equal to $\sigma_1 * \sigma_2 * \sigma_3$. In summary, we will write a multivector $\mathbf{x}$ as a quadruple $(\lambda, \mathbf{v}, \mathbf{b}, \mathbf{t})$, where λ, $\mathbf{v}$, $\mathbf{b}$, and $\mathbf{t}$ are a scalar, a vector, a bivector and a trivector, respectively. An i-vector of a multivector is said to be at grade i of the multivector. The i-vector of $\mathbf{x}$ is denoted by $\langle \mathbf{x} \rangle_i$, or $\mathbf{x}_i$ if the context guarantees the unambiguity. We call a multivector $\mathbf{x}$ *homogeneous* if $\langle \mathbf{x} \rangle_i = \mathbf{x}$ for a single i such that $0 \leq i \leq 3$. Moreover, we call i-vector $\mathbf{x}_i$ of such a homogeneous multivector $\mathbf{x}$ *i-blade*.

We can show that, by appropriately defining * and + on multivectors, the set of multivectors together with addition operator + (with neutral element 0) and multiplication operator * (with neutral element 1) form an algebra (unit) ring, where the unit is the multiplicative neutral element 1. Furthermore, real numbers and i-vectors ($i = 1,2,3$) form a real linear spaces. We can see that the quadratic form of the multivectors can be defined, and it provides the squared magnitude of the multivectors. All these properties qualify our algebra $\mathcal{G}_3$ as a commonly agreed geometric algebra.

For geometric reasoning, we define $\cdot$ and $\wedge$, the operators for constructing inner and outer products, respectively, as follows. For any r-blade $\mathbf{a}_r$ and s-blade $\mathbf{b}_s$, we define each inner and outer products as follows:

$$\mathbf{a}_r \cdot \mathbf{b}_s = \langle \mathbf{a}_r * \mathbf{b}_s \rangle_{|r-s|}$$

$$\mathbf{a}_r \wedge \mathbf{b}_s = \begin{cases} \langle \mathbf{a}_r * \mathbf{b}_s \rangle_{r+s} & r + s \leq 3 \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Inner and outer products of $\mathcal{G}_3$ agree with the notions of inner and outer products in the usual vector algebra and via the former, algebraic expressions in $\mathcal{G}_3$ often lead to easier geometric interpretations. For instance, orthogonality and parallelism of 1-vectors $\mathbf{a}$ and $\mathbf{b}$ are expressed as $\mathbf{a} \cdot \mathbf{b} = 0$ and $\mathbf{a} \wedge \mathbf{b} = \mathbf{0}$, respectively.

2.2 Formalization in Isabelle/HOL

We first define concrete data structures. We define a type `mvec` of a multivector as a codatatype in Isabelle. We could have used a datatype but by defining this notion coalgebraically, we can formalize the various multivector operations more elegantly by considering their grades separately.

```
codatatype mvec =
 Mvec (Scalar: real) (Vec: "real^3") (Bivec: "real^3") (Trivec: real)
```

We can then start to take advantage of Isabelle type classes [3] to show that our concrete type `mvec` is an instance of various algebraic structures. We do so by providing in each case suitable definitions for the class operations and proofs of their properties as specified by the class. For instance, to start with, we easily show that our multivectors form an abelian group under addition.

```
instantiation mvec :: ab_group_add
begin
primcorec zero_mvec where
  "Scalar 0 = 0"
| "Vec 0 = 0"
| "Bivec 0 = 0"
| "Trivec 0 = 0"

primcorec plus_mvec where
  "Scalar (x + y) = Scalar x + Scalar y"
| "Vec (x + y) = Vec x + Vec y"
| "Bivec (x + y) = Bivec x + Bivec y"
| "Trivec (x + y) = Trivec x + Trivec y"

primcorec uminus_mvec where
  "Scalar (-x) = - Scalar x"
| "Vec (-x) = - Vec x"
| "Bivec (-x) = - Bivec x"
| "Trivec (-x) = - Trivec x"

primcorec minus_mvec where
  "Scalar (x - y) = Scalar x - Scalar y"
| "Vec (x - y) = Vec x - Vec y"
| "Bivec (x - y) = Bivec x - Bivec y"
| "Trivec (x - y) = Trivec x - Trivec y"
instance
by intro_classes (simp_all add: mvec_eq_iff)
end
```

The use of a codatatype `mvec` enables us to use corecursion to define addition, for instance, and nicely separate the results for its scalar, vector, bivector and trivector parts.

We also define the geometric product and then show that multivectors form a monoid. We only give below a brief extract of the formalization due to space considerations. The proof scripts of the formalization of $\mathcal{G}_3$ are available at `http://www.i-eos.org:8080/ieos/OpenXchive`.

```
instantiation mvec :: monoid_mult
begin

primcorec times_mvec where
  "Scalar (x * y) = Scalar x  * Scalar y  + (Vec x  · Vec y) -
(Bivec x  · Bivec y) - Trivec x * Trivec y"

| "Vec (x * y)    = ..."
| "Bivec (x * y)    = ..."
| "Trivec (x * y)    = ..."
end
```

The full definition of the geometric product for `mvec` is derived by using the definitions of $\mathfrak{i}_1$, $\mathfrak{i}_2$, and $\mathfrak{i}_3$, and the property that σ_1, σ_2, and σ_3 are pairwise anti-commute.

With multiplication defined, we show `mvec` is a multiplicative monoid.

To obtain an algebra, we define a scalar product (denoted by $*_R$ in Isabelle) and show that it has the expected property by proving that `mvec` is an instance of Isabelle's algebra type class:

```
instantiation mvec :: real_algebra
begin

primcorec scaleR_mvec where
  "Scalar (r *R x) = r * Scalar x"
| "Vec (r *R x) =  r *R (Vec x)"
| "Bivec (r *R x) =  r *R (Bivec x)"
| "Trivec (r *R x) = r * Trivec x"

instance
proof
  fix a b :: real and x y :: mvec
  show "a *R (x + y) = a *R x + a *R y"
    by (simp add: mvec_eq_iff distrib_left scaleR_add_right)
  show...
qed
end
```

We consider $\mathcal{G}_3$ now. Let us first recapitulate what we have done so far. We assume that type scalar is synonymous to type real, and define datatypes `vec`, `bivec`,`trivec` and `mvec`, which represent types 1-vector, bivector and multivector. In Isabelle/HOL syntax these are written as

```
type_synonym scalar  =  real
type_synonym vec = scalar^3
type_synonym bivec = scalar^3
codatatype
 mvec = Mvec (Scalar: scalar) (Vec: vec) (Bivec: bivec) (Trivec: trivec)
```

The symbols `Mvec`, `Scalar`, `Vec`, `Bivec`, and `Trivec` play a similar role of the constructor symbols. For example, `Mvec 0 (Vec 1 0 0) (Bivec 0 1 0) 1)` is an Isabelle object of type `mvec`. This object is translated to the term `Mvec[0, Vec[1, 0, 0], Bivec[0, 1, 0], Trivec[1]]` of *Mathematica*.

Next we define $\mathcal{G}_3$ as a type class.

```
class g3= ring+ real_algebra + one+
fixes quad_form:: "'a ⇒ bool"
```

After defining functions that operate on the terms of the defined (co)datatypes and lemmas that will be necessary for proving `mvec` is a type in $\mathcal{G}_3$, we complete the instantiation proof by issuing

instantiation `mvec::g3`

The identities relating objects of types `mvec`, `scalar`, `vec`, `bivec`, and `trivec` are then defined for computational purposes. These can be straightforwardly proved in most cases using the multivariate package of Isabelle/HOL. We formalized a large number of identities involving the addition, multiplication, the inner and outer products of multivectors in this way. These are used as rewrite rules during geometric operations and during verification. The identities can be found in standard textbooks on geometric algebra, e.g. [4, 1] and [5], but in our approach, they are first proved in Isabelle/HOL and then are manually translated to those expressions of *Mathematica* for their use.

An alternative representation of multivectors in $\mathcal{G}_3$ (and $\mathcal{G}_2$) would be as one level tuples since the bases can be n-tuples (8-tuples in $\mathcal{G}_3$). However, the representation of a multivector as a combination of blades is more convenient from the verification and programming points of view since it carries typing information.

3 HO: Set of Huzita's elementary folds

We next give a set HO, Huzita's elementary fold set [6][4]. HO plays a fundamental role in origami geometry, as the (five) postulates of the 2D Euclidean geometry play. In the 2D Euclidean geometry, compasses and straightedges are used, but HO relies only on folding of a piece of paper without breaking it. HO is written in the natural language, as are the five Euclidean postulates. Even though geometric notions are implicit in Huzita's statements, he added the comment about the corresponding geometric implication in each statement. Namely, (O1), described shortly, corresponds to line-drawing by applying a straightedge, (O4) to the perpendicular footing, (O5) to drawing a tangent to a parabola and (O6) to "non-existing in the geometry so far known thus making origami geometry superior".

In Eos, for reasoning about and computation of geometric objects of origami, they are first transcribed to expressions of the language of the fragment of the first-order logic and then to the elements of a polynomial ring over real. HO consists of the following six statements (O1) $\sim$ (O6). For the implementation of Eos and its relation to HO, the readers are referred to [7].

In this paper, we extend HO to enable us to construct a class of 3D origamis. We confine our attention to the cases that all the points and lines that occur as the parameters to the Huzita's elementary folds lie on the same plane $\mathcal{P}$, and that the set $\mathcal{F}$ of the origami faces that we fold should lie on that plane. We call $\mathcal{F}$ and $\mathcal{P}$ *base face set* and *base plane* of the elementary fold, respectively.

[4] In some publications, Huzita-Justin set is explained as the complete elementary fold set, and add one more elementary operation (O7). However, in this paper, we restrict ourselves to Huzita's set since we would like to focus on algebraic interpretations rather than axiomatic treatment.

Let $\mathcal{O}$ denote an origami structure on the base plane $\mathcal{P}$, whose details are left unspecified in this paper. We fold an origami $\mathcal{O}$ along a fold line determined by the parameters of each elementary operation. The parameters are lines determined by a pair of distinct points on the faces, and points on the faces.

(O1) Given two distinct points P and Q, both on $\mathcal{F}$, fold $\mathcal{O}$ along the line on $\mathcal{P}$ that passes through P and Q.

(O2) Given two distinct points P and Q, both on $\mathcal{F}$, fold $\mathcal{O}$ along the line on $\mathcal{P}$ to superpose P and Q.

(O3) Given two distinct lines m and n, both on $\mathcal{F}$, fold $\mathcal{O}$ along a line on $\mathcal{P}$ to superpose m and n.

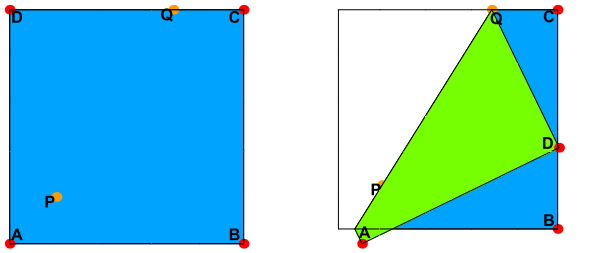
(O4) Given a line m and a point P, both on $\mathcal{F}$, fold $\mathcal{O}$ along the line on $\mathcal{P}$ passing through P to superpose m onto itself.

(O5) Given a line m, a point P not on m and a point Q, where m, $P$and $Q$are on $\mathcal{F}$, fold $\mathcal{O}$ along a line on $\mathcal{P}$ passing through Q to superpose P and m.

(O6) Given lines m and n, a point P not on m and a point Q not on n, where m and n are distinct or P and Q are distinct, and furthermore m, n, P and Q are on $\mathcal{F}$, fold $\mathcal{O}$ along a line on $\mathcal{P}$ to superpose P and m, and Q and n.

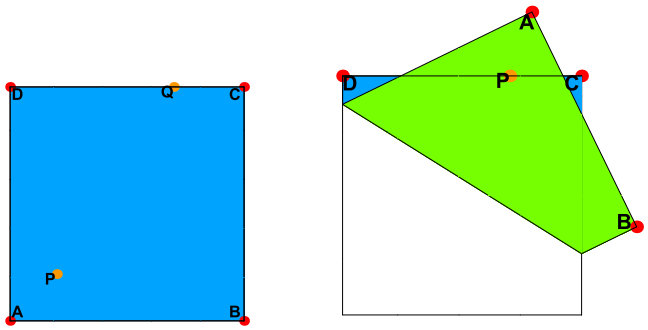
Subsequently, we often denote a line by a sequence of two points, like PQ, which is the line passing through points P and Q.

Clearly, those statements, if interpreted as those of the decision problems, the answers are all positive. We have algorithms, using algebraic methods, to find fold lines, if any, along which we fold origami $\mathcal{O}$ in each case. We will show how we translate (O1) $\sim$ (O6) into expressions of $\mathcal{G}_3$ involving multivector $\mathbf{x}$ which represents a fold line.



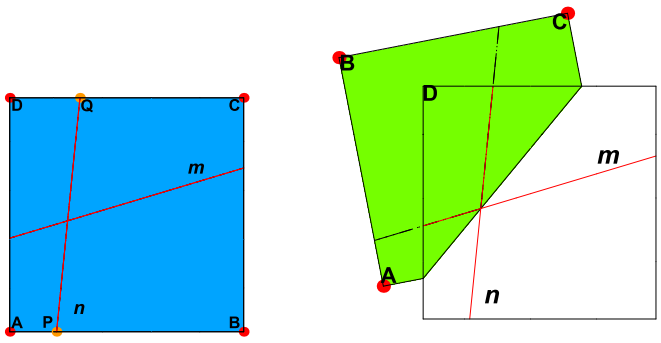
(a) Points P and Q (b) Folded along fold line PQ

Fig. 1: Fold (O1)



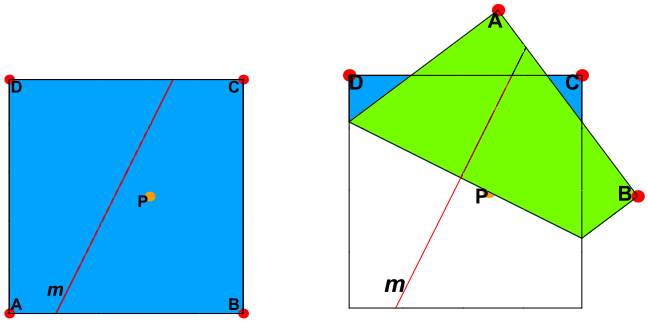
(a) Points P and Q (b) Folded along the fold line

Fig. 2: Fold (O2)



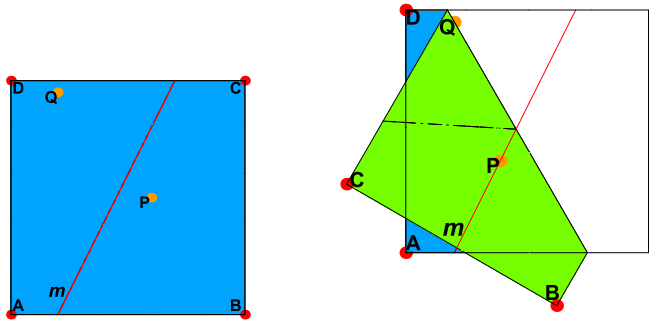
(a) Lines m and n (b) Folded along one of the two fold lines

Fig. 3: Fold (O3)



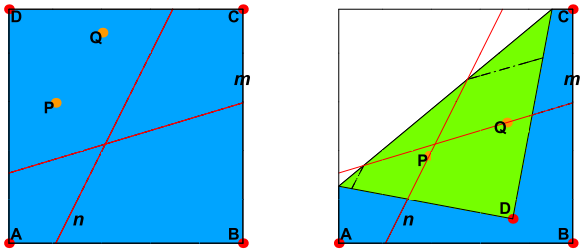
(a) Line m and point P (b) Folded along the fold line

Fig. 4: Fold (O4)



(a) Line m, points P and Q (b) Folded along one of the two fold lines

Fig. 5: Fold (O5)



(a) Lines m, n, and points P, Q (b) Folded along one of the three fold lines

Fig. 6: Fold (O6)

4 HO in the geometric algebra

4.1 O1 in GA

In $\mathcal{G}_3$, points, lines, planes, etc. are treated uniformly as multivectors. Therefore, points `P`, `Q` etc. are multivectors $\mathbf{p}$ and $\mathbf{q}$ etc., respectively. We denote any points on the fold line by $\mathbf{x}$. The line passing through distinct points `P` and `Q` is expressed in $\mathcal{G}_3$ as follows.

$$(\mathbf{x} - \mathbf{p}) \wedge (\mathbf{q} - \mathbf{p}) = \mathbf{0}. \tag{1}$$

Since $\mathbf{q}$ - $\mathbf{p}$ can be written as λ $\mathbf{u}$ for any scalar $\lambda \neq 0$ and a unit vector $\mathbf{u}$ whose direction is the same as $\mathbf{q}$ - $\mathbf{p}$, we have

$$(\mathbf{x} - \mathbf{p}) \wedge \mathbf{u} = \mathbf{0}. \tag{2}$$

as the standard representation of a line whose direction is $\mathbf{u}$ and passes through point $\mathbf{p}$.

To obtain the traditional form of line equations, let us recall that we represent $\mathbf{x}$ by (x_1, x_2, x_3). Thus term (Vec x_1 x_2 x_3) in Isabelle/HOL (and Vec[x_1, x_2, x_3] in *Mathematica*) is the representation of vector $\mathbf{x} = x_1 * \sigma_1 + x_2 * \sigma_2 + x_3 * \sigma_3$. Given $\mathbf{p} = (p_1, p_2, p_3)$ and $\mathbf{q} = (q_1, q_2, q_3)$, a straightforward symbolic computation on Eq. (1) yields the following three equations in the polynomial ring.

$$p_3 q_2 + q_3 x_2 + p_2 x_3 = p_2 q_3 + p_3 x_2 + q_2 x_3$$

$$p_3 q_1 + q_3 x_1 + p_1 x_3 = p_1 q_3 + p_3 x_1 + q_1 x_3$$

$$p_2 q_1 + q_2 x_1 + p_1 x_2 = p_1 q_2 + p_2 x_1 + q_1 x_2$$

Noting the conditions that `P` and `Q` are distinct, we solve the above equations for x_1, x_2 and x_3 and obtain

$$x_2 \to -\frac{x_1(q_2 - p_2)}{p_1 - q_1} - \frac{p_2 q_1 - p_1 q_2}{p_1 - q_1}, x_3 \to -\frac{x_1(q_3 - p_3)}{p_1 - q_1} - \frac{p_3 q_1 - p_1 q_3}{p_1 - q_1} \tag{3}$$

when $p_1 \neq q_1$. Equation (3) is a yet another algebraic representation of the desired line in 3D. By virtue of our choice of the basis σ_1, σ_2 and σ_3, they are the expressions of the line in Cartesian coordinate system. The other cases of the combinations of unequality and equality of p_1 and q_1, of p_2 and q_2 and of p_3 and q_3 are dealt with similarly and gives the corresponding line representation.

In practice, we can take the base plane to be $x_3 = 0$. Then we obtain the following line equation.

$$(-p_2 + q_2)(-p_1 + x_1) - (-p_1 + q_1)(-p_2 + x_2) = 0$$

from Eq. (1).

4.2 O2 in $\mathcal{G}_3$

The statement (O2) is translated to the set of $\mathcal{G}_3$ equations as follows.

1. We stipulate that the plane defined by $\mathbf{p} \wedge \mathbf{q}$ is a base plane.
2. We have a non-degeneracy condition that $P \neq Q$.
3. We define the midpoint m of P and Q, i.e. $\mathbf{m} = (\mathbf{p} + \mathbf{q})/2$.
4. The line $\mathbf{x}$ passes through m and is orthogonal to line PQ.

$$(\mathbf{x} - \mathbf{m}) \cdot (\mathbf{p} - \mathbf{q}) = \mathbf{0} \tag{4}$$

5. Line $\mathbf{x}$ is on the base plane: $\mathbf{x} \wedge \mathbf{p} \wedge \mathbf{q} = \mathbf{0}$.

Note that without Condition 5, we would have obtained an infinite number of lines. We are only interested in the line that is on the base face. The line $\mathbf{x}$ is then the set of points (x_1, x_2, x_3) that satisfies

$$\begin{aligned} &(p_1 \neq q_1 \bigvee p_2 \neq q_2 \bigvee p_3 \neq q_3) \bigwedge \\ &p_1 q_3 x_2 + p_2 q_1 x_3 + p_3 q_2 x_1 = p_1 q_2 x_3 + p_2 q_3 x_1 + p_3 q_1 x_2 \bigwedge \\ &{p_1}^2 + {p_2}^2 + {p_3}^2 + 2q_1 x_1 + 2q_2 x_2 + 2q_3 x_3 = 2p_1 x_1 + 2p_2 x_2 + 2p_3 x_3 + \\ &{q_1}^2 + {q_2}^2 + {q_3}^2 \end{aligned} \tag{5}$$

The above set of the equations can be solved numerically at the time of origami construction. To fold the origami, we need to know the fold line, which the solutions of Eq. (5) can fully specify. The symbolic form of Eq. (5) is saved and later used for proving purpose.

The following lemma is useful as the situation it specifies frequently appears when we define a line m that is orthogonal to line n.

Lemma 1 *Let $\boldsymbol{p}$, $\boldsymbol{q}$ and $\boldsymbol{r}$ be vectors denoting pairwise distinct points* P, Q *and* R*, respectively. Furthermore, we assume that* P, Q *and* R *are on the base plane whose direction is the same as that of* i_3 *and that* P *and* Q *are on line* n*. The following equation represents the line that passes through* R *and is orthogonal to the line* n*.*

$$(\boldsymbol{x} - \boldsymbol{r}) \wedge \langle (\boldsymbol{p} - \boldsymbol{q}) * \mathrm{i}_3 \rangle_1 = \boldsymbol{0} \tag{6}$$

The proof is done by a straightforward algebraic manipulation on multivectors. For the geometric interpretation of Eq. (6), see Fig. 7. The line (to the right, colored in blue) is the solution of $\mathbf{x}$ of Eq. (6).

Using Lemma 1, we immediately obtain the fold line of (O2):

$$(\mathbf{x} - \frac{1}{2}(\mathbf{p} + \mathbf{q})) \wedge \langle (\mathbf{p} - \mathbf{q}) * \mathrm{i}_3 \rangle_1 = \mathbf{0}$$

4.3 O3 in $\mathcal{G}_3$

It is natural to view (O3) as a special case of (O6). It is a degenerate case where (i) the two lines m and n are the same, (ii) points P and Q in (O6) define another line, say l, and (iii) l and m superpose (cf. Fig. 8).

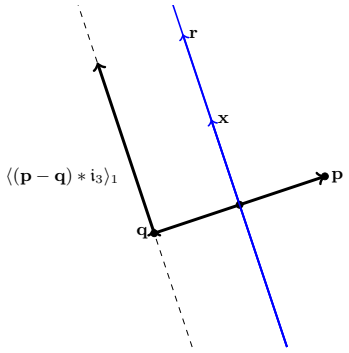
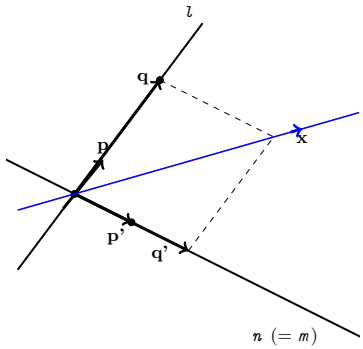


Fig. 7: Line $\mathbf{x}$ in Lemma 1



The line in blue is one of the two fold lines.

Fig. 8: Fold line in (O3) in $\mathcal{G}_3$

4.4 O4 in $\mathcal{G}_3$

Referring to Fig. 4a, we assume that distinct points Q and R define line m. Then, we immediately obtain the desired fold line by Lemma 1.

$$(\mathbf{x} - \mathbf{p}) \wedge \langle(\mathbf{q} - \mathbf{r}) * \mathbf{i}_3\rangle_1 = \mathbf{0},$$

where $\mathbf{q} \neq \mathbf{r}$.

As an example, let us consider the case where the base face is on the base plane $(x_1, x_2, 0)$ i.e. $z = 0$ plane in Cartesian coordinate system. The fold line determined by (O4) is then the set of points (x_1, x_2, x_3) that satisfies the relation

$$(q_1 - r_1)(-p_1 + x_1) - (-q_2 + r_2)(-p_2 + x_2) = 0 \bigwedge$$
$$\neg(q_1 = r_1 \bigwedge q_2 = r_2),$$

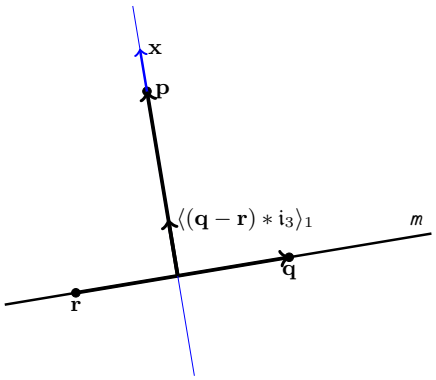


Fig. 9: Geometric configuration in (O4) in $\mathcal{G}_3$

where $\mathbf{r} = (r_1, r_2, 0)$.

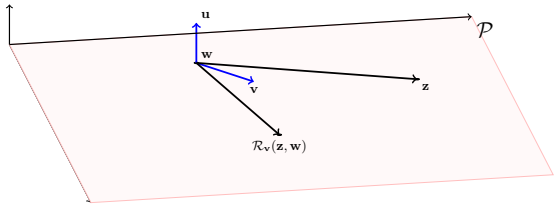
4.5 O5 in $\mathcal{G}_3$

We can view the operations (O5) and (O6) as a fold to bring a point onto a line or vice versa, as seen from Figs 13 and 14. The fold reflects the point across the fold line determined by the parameters of (O5) or (O6).

Let $\mathbf{w}$ and $\mathbf{z}$ be vectors, and $\mathbf{v}$ be a unit vector. We then define a reflection operator $\mathcal{R}_{\mathbf{v}}(\mathbf{z}, \mathbf{w})$, as follows.

$$\mathcal{R}_{\mathbf{v}}(\mathbf{z}, \mathbf{w}) = \mathbf{v} * (\mathbf{z} - \mathbf{w}) * \mathbf{v} + \mathbf{w}.$$

$\mathcal{R}_{\mathbf{v}}(\mathbf{z}, \mathbf{w})$ reflects $\mathbf{z}$ across $\mathbf{v}$ at pivot $\mathbf{w}$ (cf. Fig. 10).



The base plane $\mathcal{P}$ (colored in light grey) is specified by vector $\mathbf{u}$, which is orthogonal to $\mathcal{P}$. Vectors $\mathbf{z}$, $\mathbf{z}$' and $\mathbf{v}$ are on the base plane and $\mathbf{v}$ is unit vector.

Fig. 10: Reflection

Referring to Fig. 5a, we assume assume that distinct points S and T define line m. Since the fold line passes through point Q and its direction is determined

by a unit vector $\mathbf{u}$, we have the following equation for $\mathbf{x}$.

$$(\mathbf{x} - \mathbf{q}) \wedge \mathbf{u} = \mathbf{0},$$

satisfying the following conditions.

non-degeneracy :	$\mathbf{s} \neq \mathbf{t}$
reflected point on m :	$\langle \mathcal{R}_{\mathbf{u}}(\mathbf{p}, \mathbf{q}) - \mathbf{s} \rangle_1 \wedge (\mathbf{t} - \mathbf{s}) = \mathbf{0}$
unit vector :	$\mathbf{u} * \mathbf{u} = 1$
$\mathbf{u}$ on the base plane :	$\mathbf{u} \wedge \mathrm{i}_3 = \mathbf{0}$

Figure 11 shows the geometric configuration of (O5) before the fold. This corresponds to the origami of Fig. 5a.

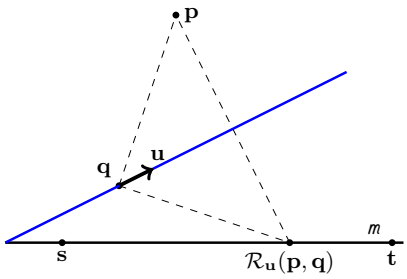


Fig. 11: Geometric configuration of (O5) in $\mathcal{G}_3$

4.6 O6 in $\mathcal{G}_3$

Finally, we derive the following equations from the statement (O6). We assume that the pairs of distinct points A and B, and of distinct points C and D determine lines m and n, respectively. Recall that we stipulate all the points A, B, C, D, P and Q of the arguments of (O6) are on the base face.

The desired fold line satisfies

$$(\mathbf{x} - \mathbf{w}) \wedge \mathbf{u} = \mathbf{0},$$

where $\mathbf{g}$ and $\mathbf{u}$ satisfy the following conditions.

non-degeneracy :	$\mathbf{a} \neq \mathbf{b}$
reflected point on m :	$\langle \mathcal{R}_{\mathbf{u}}(\mathbf{p}, \mathbf{w}) - \mathbf{a} \rangle_1 \wedge (\mathbf{b} - \mathbf{a}) = \mathbf{0}$
non-degeneracy :	$\mathbf{c} \neq \mathbf{d}$
reflected point on n :	$\langle \mathcal{R}_{\mathbf{u}}(\mathbf{q}, \mathbf{w}) - \mathbf{c} \rangle_1 \wedge (\mathbf{d} - \mathbf{c}) = \mathbf{0}$
unit vector :	$\mathbf{u} * \mathbf{u} = 1$
$\mathbf{u}$ on the base plane :	$\mathbf{u} \wedge \mathrm{i}_3 = \mathbf{0}$

Figure 12 shows the geometric configuration of (O6) before the fold. This corresponds to the origami of Fig. 6a.

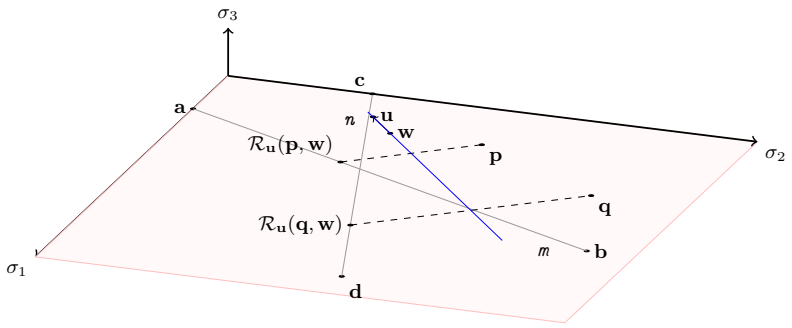


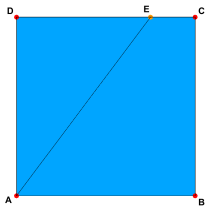
Fig. 12: Geometric configuration in (O6) in $\mathcal{G}_3$

5 Example - trisection of an arbitrary angle

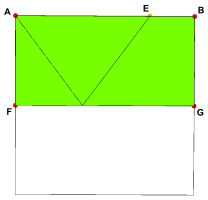
In this section, we give a simple example of trisecting an angle by origami. We consider both the construction and verification of a geometric problem of trisecting an arbitrary angle. We may regard this as an example pertinent to origami, as trisecting an arbitrary angle is impossible by the Euclidean tool of a straightedge and a compass. Abe devised a method for trisecting an arbitrarily given angle using Huzita's elementary operations [8]. Figure 13 shows the steps of his construction, excluding the steps of creating an angle $\angle EAB$. This construction is even simpler than Abe's, in that it requires only one supporting horizontal crease.

Let us see the construction in detail. Let E be an arbitrary point on the edge of DC (cf. Fig. 13a). The problem is to trisect $\angle EAB$. If we exclude the steps of unfolding, the number of the folds from step 3 is only three. At step 4 in Fig. 13b, we superpose points A and D. Step 6 in Fig. 13d is the crucial step of the construction. We superpose point D and line AE, and point A and line GF, simultaneously. We can find such fold lines that make these superpositions possible.

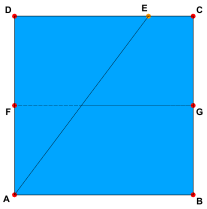
To find such fold lines, we need to solve a cubic equation whose solutions give the three possible fold lines. We choose the fold line that leads to the trisection of the interior angle $\angle EAB$. So we fold the origami along fold line MN as shown in Fig. 13d. In Fig. 13e, we project points D, F and A onto the base plane and create points L, K and J, respectively, and then unfold the origami. At step 8; we fold the origami along line AJ, and at step 9 we unfold the resulting origami. This completes the construction. Figure 13h is not a part of the construction, but is to emphasize the trisectors. Further details can be found in [9].



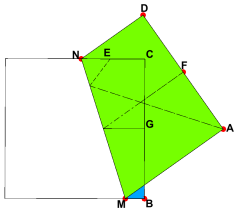
(a) Step 3: Angle $\angle EAB$



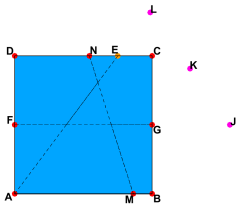
(b) Step 4: (O2)



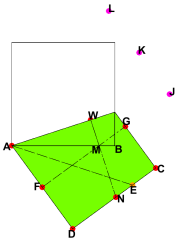
(c) Step 5: Unfold



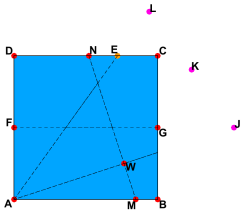
(d) Step 6: (O6)



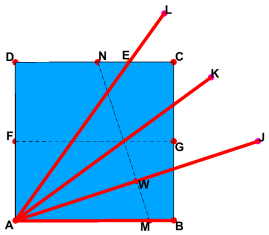
(e) Step 7: Unfold



(f) Step 8:(O1)



(g) Step 9: Unfold



(h) Two trisectors of $\angle EAB$

Fig. 13: Abe's method for trisecting angle $\angle EAB$

From these operations, we obtain the trisector AJ, which is the solution of $\mathbf{x}$of the equation

$$(\mathbf{x} - \mathbf{w}) \wedge \mathbf{u} = 0,$$

with the following relations:

$$\langle \mathcal{R}_{\mathbf{u}}(\mathbf{d}, \mathbf{w}) - \mathbf{a} \rangle_1 \wedge (\mathbf{e} - \mathbf{a}) = \mathbf{0} \tag{7}$$
$$\langle \mathcal{R}_{\mathbf{u}}(\mathbf{a}, \mathbf{w}) - \mathbf{g} \rangle_1 \wedge (\mathbf{f} - \mathbf{g}) = \mathbf{0} \tag{8}$$
$$\mathbf{w} = \frac{1}{2} \langle \mathbf{a} + \mathcal{R}_{\mathbf{u}}(\mathbf{a}, \mathbf{w})) \rangle_1 \tag{9}$$
$$\mathbf{j} = \langle \mathcal{R}_{\mathbf{u}}(\mathbf{a}, \mathbf{w}) \rangle_1 \tag{10}$$
$$\mathbf{l} = \langle \mathcal{R}_{\mathbf{u}}(\mathbf{d}, \mathbf{w}) \rangle_1 \tag{11}$$
$$\mathbf{k} = \langle \mathcal{R}_{\mathbf{u}}(\mathbf{f}, \mathbf{w}) \rangle_1 \tag{12}$$
$$\mathbf{u} * \mathbf{u} = \mathbf{1} \tag{13}$$

Point $W$is the pivot of the reflection in the above equations, and is the intersection of the fold line MN and line AJ as shown in Figs. 13f and 13h.

Next, we prove that lines AK and AJ are the trisectors of $\angle EAB$. To that end, we show that vectors $\mathbf{j}$, $\mathbf{k}$ and $\mathbf{l}$ are obtained by rotating vectors $\mathbf{b}$, $\mathbf{j}$ and $\mathbf{k}$ with $\mathbf{a}$ as the pivot, by an angle θ, respectively. We therefore have the following three goals of the proof. If the goals are proven to be correct, then we have $\theta = \frac{1}{3}\angle EAB$.

$$\langle (\mathbf{b} - \mathbf{a}) * e^{\mathrm{i}_3 \theta} \rangle_1 \wedge (\mathbf{j} - \mathbf{a}) = \mathbf{0} \tag{14}$$
$$\langle (\mathbf{j} - \mathbf{a}) * e^{\mathrm{i}_3 \theta} \rangle_1 \wedge (\mathbf{k} - \mathbf{a}) = \mathbf{0} \tag{15}$$
$$\langle (\mathbf{k} - \mathbf{a}) * e^{\mathrm{i}_3 \theta} \rangle_1 \wedge (\mathbf{l} - \mathbf{a}) = \mathbf{0} \tag{16}$$

We negate each equations (14) $\sim$ (16). Let us denote them by (14'), (15'), and (16'). We translate equations (7) $\sim$ (13), and (14') to the set of polynomial equations. The Gröbner basis computation of the polynomials thus obtained yields the reduced Gröbner base $\{1\}$. The same computing procedures for the other two goals also yield $\{1\}$. Thus, the proof is successful. Note that $e^{\mathrm{i}_3 \theta}$ is an expression of $\mathcal{G}_3$, and should be interpreted in the following way. The multiplications with $e^{\mathrm{i}_3 \theta}$ in Eqs. (14) $\sim$ (16) generate sinusoidal functions. These have to be transformed to polynomials using the identity $\sin^2 \theta + \cos^2 \theta = 1$ for arbitrary θ before they are processed by Gröbner basis computations.

6 Related works

We have two sources to recall in relating our present work to previous works, i.e., origami and geometric algebra. As for the former, we see the significant difficulties and challenges ahead in constructing 3D origamis. To circumvent the difficulties, Lang introduced search heuristics to find fold patterns in 3D [10] and Streinu et al. analyzed Lang's algorithm and made it more reliable

in more cases [11]. While our approach is axiomatic, at some stage of further development, we will explore the possibility of incorporating heuristics for finding new folds. Especially, to place appropriate constraints on the use of parameters to a fold operation is necessary for feasible 3D origami construction.

As for the latter, geometric algebra, now commonly coined as such, is a family of algebras and has the fascinating history. There exist many kinds of literature including textbooks at the undergraduate and graduate levels and research monographs on it. We will point out the publications in the context of automated geometric theorem proving, and limit ourselves to those relating directly to our present study of origami. The main motivation of the use of GA is to explore computational methods of geometric theorem proving, that are freed from the dimensions and the coordinates. The work of Wang et al [12][13], among others, is one of the pioneering ones with this motivation. They tried to apply the equations of GA systematically and successfully proves classical geometrical theorems of Euclidean geometry. Yhey went on to construct a rewrite system with nice computational properties such as strong normalization and confluence. To define a rewrite system from a large number of employable equations, a proper choice of rules and/or Knuth-Bendix completion procedure seem a formidable task.

Our approach in the context of the origami research is a hybrid approach in which we use GA in formalizing geometrical properties and use polynomial rings that we can obtain by compiling GA expressions, for computationally intensive tasks. The work was initiated to formalize 2D origami geometry in 2014. We introduced a variant of $\mathcal{G}_2$ [14]. In the similar vein, our axiomatic study of algebraic treatment of origami may benefit from a variant of geometric algebra, conformal geometric algebra, proposed by Mizoguchi et al. [15].

Another work that has similar motivation, but differs from our approach is that of Fuchs' and Théry's [16, 17]. They formalized GA in Coq using the data structure of a binary tree. Their goal is not only to formalize GA but also to derive a set of efficient functions that we can use for non-trivial computations.

As for the implementation of GA in general, depending on the applications, a wide range of software libraries and systems are available. Among them, we note the work of Aragón-Camerasa et al. on the Mathematica package implementation of GA [18]. As our GA is designed and implemented to be a part of Eos system (also implemented in Mathematica), the analysis of the package would be of benefit to our further system development. However, the direct use of the package by us is unlikely as it would involve much efforts of transforming the data structures, among others.

7 Concluding remarks

We have shown a rigorous approach to the use of the geometric algebra in computational origami. We proved in Isabelle/HOL most of the useful identities that hold among algebraic expressions of $\mathcal{G}_3$. Thus, we can state the elementary origami operations in geometric algebra and further correctly apply the equations

to compute the transformations of the objects under study both symbolically and numerically. Furthermore, we can now use $\mathcal{G}_3$ to geometric theorem proving. The example we have given in Section 5 is simple enough to show how we can use our methodology to automated theorem proving in geometry. However, it is scalable to other more sophisticated geometric problems.

We can observe the importance of the geometric algebra as the layer of abstraction between the language for the origami geometry and the algebraic expressions. We also see that the distinctions between points, lines and vectors disappear and that the derived expressions do not depend on the dimensionality of the coordinate system.

Since we want to integrate the functionality of $\mathcal{G}_3$ into the computational system tightly, we start with the formalization of $\mathcal{G}_3$ in Isabelle/HOL with the help of *Mathematica*, and then for most computational, i.e. simplification, purposes, we use *Mathematica*. In this paper, we focussed on the transformation from HO to $\mathcal{G}_3$. Further work is in progress to incorporate the obtained results to extend Eos to 3D visualization and the verification of 3D origami.

Acknowledgments

This work was supported by JSPS KAKENHI Grants Numbers 25330007 and 16K00008.

References

1. D. Hestenes: New Foundations for Classical Mechanics (Second Edition). Kluwer Academic Publishers (1986)
2. Ida, T., Takahashi, H., Marin, M., Ghourabi, F., Kasem, A.: Computational Construction of a Maximal Equilateral Triangle Inscribed in an Origami. In: Mathematical Software - ICMS 2006. Volume 4151 of Lecture Notes in Computer Science., Springer (2006) 361–372
3. Haftmann, F.: Haskell-style Type Classes with Isabelle/Isar. http://isabelle.in.tum.de/doc/classes.pdf (2014)
4. C. Doran and A. Lasenby: Geometric Algebra for Physicists. Cambridge University Press (2003)
5. L. Dorst and D. Fontijne and S. Mann: Geometric Algebra for Computer Science. Elsevier Inc. (2007)
6. Huzita, H.: Axiomatic Development of Origami Geometry. In Huzita, H., ed.: Proceedings of the First International Meeting of Origami Science and Technology. (1989) 143–158
7. Ida, T., Ghourabi, F., Takahashi, K.: Formalizing Polygonal Knot Origami. Journal of Symbolic Computation (2014)
8. Abe, H.: Trisecting an Angle by Origami. Sugaku Seminar (1980) cover page http://www.phoenix-c.or.jp/ tokioka/n!/suu(underscore)semi.html.
9. Ida, T., Kasem, A., Ghourabi, F., Takahashi, H.: Morley's Theorem Revisited: Origami Construction and Automated Proof. Journal of Symbolic Computation **46** (2011) 571 – 583

10. Lang, R.J.: A computational algorithm for origami design. In: The 12th annual ACM symposium on computational geometry. (1996) 98–105
11. Bowers, J.C., Streinu, I.: Rigidity of Origami Universal Molecules. In Ida, T., Fleuriot, J., eds.: Automated Deduction in Geometry. Volume 7993 of LNAI., Springer Heidelberg New York Dordrecht London (2013) 120–142
12. Wang, D. In: Clifford algebraic calculus for geometric reasoning. Springer Berlin Heidelberg, Berlin, Heidelberg (1997) 115–140
13. Fèvre, S., Wang, D. In: Combining algebraic computing and term-rewriting for geometry theorem proving. Springer Berlin Heidelberg, Berlin, Heidelberg (1998) 145–156
14. Ida, T.: Huzita's basic origami fold in geometric algebra. In: Post-Proceedings of the 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2014), IEEE Computer Society Conference Publishing Services (CPS) (2015) 11 – 13
15. Kondo, M., Matsuo, T., Mizoguchi, Y., Ochiai, H.: A mathematica module for conformal geometric algebra and origami folding. In Davenport, J.H., Ghourabi, F., eds.: SCSS 2016: 7th International Symposium on Symbolic Computation in Software Science. Volume 39 of EPiC Series in Computing., EasyChair (2016) 68–80
16. Fuchs, L., Théry, L. In: A Formalization of Grassmann-Cayley Algebra in COQ and Its Application to Theorem Proving in Projective Geometry. Springer Berlin Heidelberg, Berlin, Heidelberg (2011) 51–67
17. Fuchs, L., Théry, L.: Implementing geometric algebra products with binary trees. Advances in Applied Clifford Algebras **24** (2014) 589–611
18. Arag/'on-Camarasa, G., Arag/'on-Gonz/'alez, G., Arag/'on, J.J., Rogr/'iguez-Andrade, M.A.: Clifford Algebra with Mathematica. http://arxiv.org/dbs/0810.2412 (2008)

Automatic Rewrites of Input Expressions in Complex Algebraic Geometry Provers

Extended Abstract

Zoltán Kovács[1], Tomás Recio[2] and Csilla Sólyom-Gecse[3]

[1] Private University College of Education
of the Diocese of Linz, Austria
zoltan@geogebra.org
[2] Depto. de Matemáticas, Estadística y Computación
Universidad de Cantabria, Santander
tomas.recio@unican.es
[3] Babeş-Bolyai University, Cluj-Napoca, Romania
solyom-csilla@yahoo.com

Abstract. We present an algorithm to help converting expressions having non-negative quantities (like distances) in Euclidean geometry theorems to be usable in a complex algebraic geometry prover. The algorithm helps in refining the output of an existing prover, therefore it supports immediate deployment in high level prover systems.

Keywords: Automatic Theorem Proving, Automatic Theorem Deduction, Complex Algebraic Geometry, Elementary Geometry, Dynamic Geometry Software, GeoGebra

1 Introduction

Many dynamic geometry systems support covering textbook problems by offering an intuitive interface to construct geometry diagrams. When combined with automated theorem proving tools, the first steps are the graphical construction of the hypotheses and the final step is the definition of the thesis. In many textbook theorems in elementary geometry the thesis is a relation between some geometry objects in the diagram, but sometimes the thesis is an equation containing variables representing lengths, angles, areas or volumes. Such an example is the Pythagorean theorem, among many others, including Ptolemy's theorem or Heron's formula.

The most adequate way for entering such expressions is not yet completely discussed. Some software tools like JGEX [13] or OpenGeoProver [11] have their own user interface and rewrite mechanism to reformulate the input equation (e.g. $a = 2b$, here a and b are distances) to a more suitable form (e.g. $a^2-4b^2 = 0$). The problem with the equation $a = 2b$ is that complex algebraic geometry (CAG) solvers cannot assume that the quantities a and b are non-negative since

the variables are always chosen from the set of complex numbers. By using the more suitable form $a^2 - 4b^2 = 0$ we can implicitly ignore the difference between positive and negative values for the quantities.

The first references about the need to reformulate the thesis expression are in Chou's book [4], by specifying different rewrites for the various input expressions to be more suitable for the CAG method.

In this work we propose a general way to handle a large number of possible algebraic input expressions defined in the thesis equation. We use computer algebra to determine the more suitable expression instead of using the input equation, namely by elimination of certain variables. We claim that the number of terms of the more suitable expression is double exponential in the number of odd powers in the term form of the input—the suggested way for the computation is still achievable for the usual classroom situations.

We also suggest a modification of the typical workflow for a CAG proof of an elementary geometry theorem by factorizing the more suitable expression. The notion of "essential conditions" of a statement will be introduced, which may play an important role similarly to non-degeneracy conditions. As an illustration we recall Viviani's theorem and its one-dimensional specific case by demonstrating our algorithm implemented in GeoGebra.

2 The problem

Chou's book [4] presents a list of cases when translating statements into unordered geometry requires further considerations. In its Part II, section 2 (p. 97) Chou mentions

- the length of a segment (which should be substituted by its square),
- the equality of length of two segments (for $a = b \iff a - b = 0$ the polynomial $a^2 - b^2$ is used),
- the equality of product of two segments (for $a \cdot b = c \cdot d \iff ab - cd = 0$ the polynomial $a^2b^2 - c^2d^2$ will be utilized),
- a ratio of length of two segments (for $3a = 7b$ the expression $9a^2 - 49b^2$ is used),
- the sum of length of two segments is a third length (for $a + b = c$ the polynomial $(a - b - c) \cdot (a - b + c) \cdot (a + b - c) \cdot (a + b + c)$ is used).

Here the last case is the most difficult one. Let us illustrate it by a simple statement:

Example 1. Let a be the length of the segment joining the free points A and B. Define point C as an arbitrary point of this segment and and let the length of segment AC be b and that of BC be c. Now $a = b + c$.

Let us try to prove this easy theorem with the usual CAG approach. A possible translation of the geometric diagram into algebraic equations is to use variables $v_1, v_2, v_3, v_4, v_5, v_6$ to describe the Cartesian coordinates of points A,

B and C: $A = (v_1, v_2)$, $B = (v_3, v_4)$, $C = (v_5, v_6)$. Now we can state that $v_1v_4 + v_3v_6 + v_5v_2 - v_1v_6 - v_3v_2 - v_5v_4 = 0$, and we can also claim that $a^2 = (v_1 - v_3)^2 + (v_2 - v_4)^2$, $b^2 = (v_1 - v_5)^2 + (v_2 - v_6)^2$, $c^2 = (v_3 - v_5)^2 + (v_4 - v_6)^2$ by using the Pythagorean theorem for distances of the vertices. Here 4 equations describe the hypotheses. By following [8] and the Gröbner bases method, let us use the negated form of $a = b + c$ as $z(a - b - c) = 1$ to describe the thesis. Now we use the prover algorithm from [5] (chapter 6, §4), implemented in Singular 4.0.2 [6], and enter the following program code:

```
ring r=(0,v1,v2,v3,v4,v5),(v6,a,b,c,z),dp;
ideal i=v1*v4+v3*v6+v5*v2-v1*v6-v3*v2-v5*v4,
 a^2-(v1-v3)^2-(v2-v4)^2,
 b^2-(v1-v5)^2-(v2-v6)^2,
 c^2-(v3-v5)^2-(v4-v6)^2,
 z*(a-b-c)-1;
groebner(i);
```

We recall that (in the standard framework for automatic theorem proving as in [4]) the output is $\langle 1 \rangle$ if and only if the statement is true interpreted in an algebraic closed field. But our statement is not true in the complex numbers: in fact, the output is an ideal (a reduced Gröbner basis) with several elements. On the other hand, in the complex numbers the following can be proved:

$$(a - b - c) \cdot (a - b + c) \cdot (a + b - c) \cdot (a + b + c) = 0$$

as one can check by changing in the above list, the thesis polynomial to

```
z*((a-b-c)*(a-b+c)*(a+b-c)*(a+b+c))-1
```

In this case we have the expected result, the reduced minimal Gröbner basis is $\langle 1 \rangle$.

One important note must be remarked: the proven statement is actually the following:

Theorem 1. *Let us denote by a the length of the segment AB by joining the free points A and B. Define point C as an arbitrary point of the line going through A, B, and let length$(AC) = b$ and length$(BC) = c$. Now $a = b + c$, unless $b = a + c$ or $c = a + b$.*

It is easy to see that the last condition $a + b + c(= 0)$ in our modified input has no essential geometrical meaning, it is a degenerate case, therefore it can be omitted from the precise statement. On the other hand, the other excluded conditions do have geometrical meanings:

1. $b = a + c$, in this case C is not on the segment, but on a line which joins A and B; C is on the red dotted ray in Fig. 1. Here we must admit that our problem setting is not even compatible with CAG since we cannot distinguish a segment from a line. That is, in the geometric approach we need to clarify if we allow points outside the segment (but still on the line) or not. It is quite straightforward that in the CAG approach the only possible approach is to extend the meaning of a segment to be a line.

2. $c = a + b$, this is a similar case, but C is on the blue dotted ray.

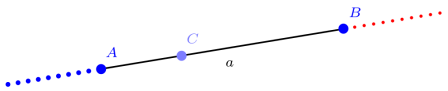


Fig. 1. Example 1

We call these two extra assumptions *essential conditions* to highlight their geometrical importance and distinguish them from the degeneracy conditions which are geometrically a "much smaller" set.

3 A detailed example

Given the input polynomial equation $p = 0$ where p is squarefree, we perform the following computation by using a computer algebra system (here Giac [7] will be used). In our example, let $p = a - b - c$.

```
>> factor(eliminate([a-b-c,a^2=A^2,b^2=B^2,c^2=C^2],[a,b,c]))
```

that is, we eliminate all terms from p which are not of even powers of a, b, c and the obtained output is:

```
[(A-B-C)*(A-B+C)*(A+B-C)*(A+B+C)]
```

This technique works in general, and as special cases it covers all instances reported by Chou. We tested this idea for several elementary geometry theorems and successfully proved them without any manual computations or further challenges. Our tests include the Pythagorean theorem, the cathetus, the geometric mean, the angle bisector, the intercept, Ceva's, Menelaus' and Ptolemy's theorems, and Heron's formula. A detailed list of our tests can be found at `http://tinyurl.com/adg16-formula-rewrite` which is generated on a daily basis automatically from the latest source code of the open dynamic geometry system GeoGebra.

As an illustration, we recall Viviani's theorem (see [12]). It is a generalization of our example in two dimensions. On the other hand, here 7 of the appearing 8 factors are essential, while one is degenerate.

Example 2. Let ABC be a regular triangle and D an internal point of it. Let i, j and k be the distance of D from the sides of the triangle, respectively. Then $i + j + k$ is a constant (it is actually the height m of the triangle, see Fig. 2).

Similarly to Example 1 here we cannot restrict D to be an internal point of the triangle. We can divide the plane to 7 different areas and enumerate them from **1** to **7**. In area **1** the equation $i + j + k - m = 0$ holds, in **2**–**4** the equations

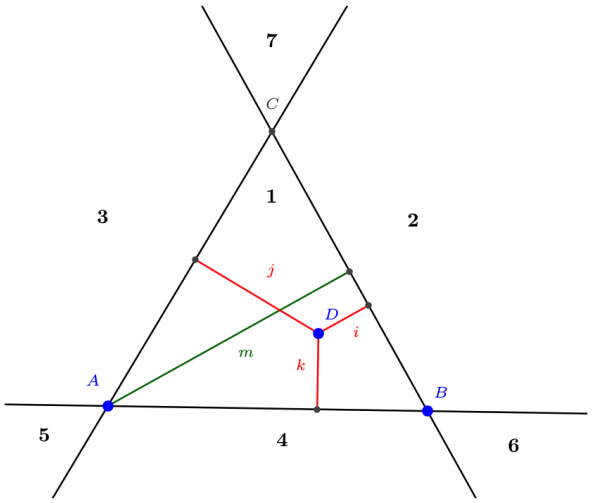


Fig. 2. Example 2: Viviani's theorem

$i-j-k+m=-(-i+j+k-m)=0$, $i-j+k-m=0$, $i+j-k-m=0$, respectively, and in **5**–**7** the equations $i-j-k-m=-(-i+j+k+m)=0$, $i-j+k+m=0$, $i+j-k+m=0$, respectively.

When using our method with input as the left hand side of any of these 7 equations, the output expression is a product of the 7 ones and an extra one, $i+j+k+m$. It is clear that this last 8th one cannot be zero, only if the triangle collapses to a point, that is, $A=B=C(=D)$, which is obviously a degenerate case.

Our method finally modifies Viviani's theorem in the following form:

Theorem 2. *Let ABC be a regular triangle and D another point on the plane. Let i, j and k be the distance of D from the sides of the triangle, respectively. Let m be the height of the triangle. Then, provided that none of the conditions*

- $i+m=j+k$,
- $i+k=j+m$,
- $i+j=k+m$,
- $i=j+k+m$,
- $j=i+k+m$,
- $k=i+j+m$

hold, $i+j+k=m$ follows.

This process is purely automatic, since both the rewrite and also adding the essential conditions can be done without any human intervention. The essential conditions will be those factors of the output which are not appearing in the input p, except the only factor which is a sum of the quantities.

Viviani's theorem can also be generalized for 3 dimensions, see [1].

4 Other uses, future work

The same idea can be applied when putting equations containing quantities among the hypotheses. An important question in the CAG approach how to distinguish between an ellipse and a hyperbola, since their definition—by using the foci A, B and a circumpoint C—cannot be separated. Actually, some theorems can be even generalized by using this fact (see [9] for an example). On the other hand, a better understanding can be obtained with the help of the factorized elimination.

For instance, given a hyperbola with foci A and B and point C, another point P is an element of the hyperbola if and only if $|AC - CB| = |AP - PB|$, that is, $(AC - CB)^2 = (AP - PB)^2$. Now let $p_h = (AC - CB)^2 - (AP - PB)^2 = (AC-CB-AP+PB)\cdot(AC-CB+AP-PB)$. Similarly, for an ellipse described with the same points, the equation $AC + CB = AP + PB$ can be assumed, so we set $p_e = AC + CB - AP - PB$. By using our method for the inputs p_h and p_e we get the same output $p_h \cdot p_e \cdot (AC + CB - AP - PB) \cdot (AC + CB + AP - PB) \cdot (AC + CB + AP + PB) \cdot (AC - CB - AP - PB) \cdot (AC - CB + AP + PB)$. Here the last 5 factors are geometrically degenerate cases, that is, the hyperbola and the ellipse are undistinguishable, but there are no other geometrical curves which can be mixed with them in the CAG approach.

This example also supports the fact that given an input polynomial p with l terms which are not of even power, (independently of the number of even powers in p) the output polynomial will consist of 2^l (or eventually 2^{l-1}) factors. In other words, the expansion of the output polynomial will consist of doubly exponential number of terms of the number of not even powers. For example, for $l = 6$ the computation will not finish in a feasable time in Giac:

```
factor(eliminate([x1+x2+x3+x4+x5+x6,x1^2=X1^2,x2^2=X2^2,x3^2=X3^2,
    x4^2=X4^2,x5^2=X5^2,x6^2=X6^2],[x1,x2,x3,x4,x5,x6]))
```

The opportunity to type arbitrary expressions (involving distances, lengths, volumes, etc.) is, in our opinion, a desirable feature of theorem provers in a dynamic geometry system, allowing the user to access new horizons in studying, discovering and enjoying Euclidean geometry.

Acknowledgments

Computing the output formula by elimination was suggested by Bernard Parisse, inventor of Giac.

We are thankful to Predrag Janičić, Julien Narboux and Francisco Botana for their suggestions to improve the text of this paper.

Second author is partially supported by the Spanish Research Project 'Construcciones Algebra-geométricas: fundamentos, algoritmos y aplicaciones' (MTM2014-54141-P).

References

1. A. Bogomolny. Viviani's 3D Analogue from Interactive Mathematics Miscellany and Puzzles. Downloaded from `http://www.cut-the-knot.org/triangle/VivianiTetrahedron.shtml`, accessed in April 2016.
2. F. Botana, M. Hohenwarter, P. Janičić, Z. Kovács, I. Petrović, T. Recio and S. Weitzhofer. Automated Theorem Proving in GeoGebra: Current Achievements. *J Autom Reasoning*, 55(1):39–59, March 2015.
3. K. Brown. Polynomials For Sums of Square Roots. Downloaded from `http://www.mathpages.com/home/kmath111/kmath111.htm`, accessed in February 2016.
4. S.-C. Chou. *Mechanical Geometry Theorem Proving*. Springer Science + Business Media, 1987.
5. D. Cox, J. Little and D. O'Shea. *Ideals Varieties and Algorithms*. Springer New York, 2007.
6. W. Decker, G.-M. Greuel, G. Pfister and H. Schönemann. SINGULAR 4-0-2 — A computer algebra system for polynomial computations. 2015. `http://www.singular.uni-kl.de`.
7. Z. Kovács and B. Parisse. Giac and GeoGebra – Improved Gröbner basis computations. *Computer Algebra and Polynomials*, Volume 8942 of the series Lecture Notes in Computer Science, 126–138. Springer, 2015.
8. D. Kapur. Using Gröbner bases to reason about geometry problems. *Journal of Symbolic Computation*, 2(4):399–408, December 1986.
9. Z. Kovács, C. Sólyom-Gecse. GeoGebra Tools with Proof Capabilities, 2016. `http://arxiv.org/abs/1603.01228`.
10. B. Kutzler and S. Stifter. On the application of Buchberger's algorithm to automated geometry theorem proving. *Journal of Symbolic Computation*, 2(4):389–397, December 1986.
11. I. Petrović and P. Janičić. Integration of OpenGeoProver with GeoGebra, 2012. `http://argo.matf.bg.ac.rs/events/2012/fatpa2012/slides/IvanPetrovic.pdf`.
12. T. J. Recio Muñiz. *Cálculo simbólico y geométrico*. Editorial Síntesis. Madrid, 1998.
13. Z. Ye, S.-C. Chou and X.-S. Gao. An Introduction to Java Geometry Expert. In *Automated Deduction in Geometry*, pages 189–195. Springer Science + Business Media, 2011.

Two Ways of Using Rabinowitsch Trick for Imposing Non-degeneracy Conditions

Manuel Ladra[1], Pilar Páez-Guillán[1], and Tomás Recio[2]

[1] Universidad de Santiago, Santiago de Compsotela, Spain
manuel.ladra@usc.es,
pilarpaez06@hotmail.com
[2] Depto. de Matemáticas, Estadística y Computación, Universidad de Cantabria, Santander, Spain
tomas.recio@unican.es

Abstract. In the algebraic-geometry-based theory of automated proving and discovery, it is usually required that the user includes, as complementary hypotheses, some intuitively obvious non-degeneracy conditions. There are two main procedures to introduce such conditions into the hypotheses set, and both of them rely on the well known Rabinowitsch trick. But it happens that each of these very close methods can lead to very different situations (truth/false statements). The aim of this extended abstract is to present and to discuss in detail this usually forgotten dilemma,highlighting the need to do some further work on this issue.

We present one example, in the context of automatic discovery, where it will be checked that, after imposing the thesis to be true and eliminating the non-independent variables, different additional hypotheses appear, depending on the chosen approach when including the non-degeneracy conditions; this fact is directly related to the change in the way we proceed eliminating variables and introducing hypotheses. All the calculations have been carried out using the software Singular in the FinisTerrae 2 supercomputer.

Keywords: Automated discovery, non-degeneracy conditions, Rabinowitsch trick, saturation

1 Introducing non-degeneracy conditions as hypotheses

The framework of this paper is the automated theorem proving theory initiated, forty years ago, by Wu on his seminal paper 'On the decision problem and the mechanization of theorem-proving in elementary geometry' [8]. Its goal is to provide computer algebra algorithms to automatically decide if a given geometric statement is generally true or not, i.e. true except for some exceptional cases, usually not explicitly declared, such as the collinearity of the three vertices of a generic triangle or dealing with circle of radius zero.

Roughly speaking, this method proceeds by assigning coordinates and equations to the elements (points, lines, circles, etc.) and conditions (perpendicularity, incidence, etc.) of the involved geometric hypotheses H and theses T, creating, in this way, two systems of polynomial equations H, T. Then it declares the validity of the geometric statement $H \Longrightarrow T$ over an algebraically closed field K by considering the inclusion $V(H) \subseteq V(T)$ between the solution set of the system of equations H and that of the corresponding system T. Finally, this inclusion is tested by some computational algebraic geometry methods (triangularization, Gröbner basis, etc.).

Now, it is often the case that in many elementary geometric statements the algebraic variety defined by the hypotheses is not totally contained in the one defined by the thesis. In fact, in most cases the output of an automatic proving algorithm is that the tested statement is false unless the hypotheses variety is further restricted by adding some complementary constraints such as: these two points must be different, this triangle should not collapse to a line, etc. These additional hypotheses are known as non-degeneracy conditions and statements that are true except in theses cases are called generally true. See [2, Chapter 6, Section 4] or [7] for further details on this issue.

As shown long time ago by the quantity and quality of the examples in [1], it is clear that this computer algebra approach to mechanical theorem proving is quite successful. Yet, it is also well known the high complexity of the involved polynomial Gröbner basis algorithms [2]. For this reason it is often the case that the user attempts to prevent the failure of the computation by manually introducing, before starting to run the proving algorithm, some easy-to-guess non-degeneracy conditions.

The main problem here is that the methods we are describing are, basically, designed only to deal with equalities, not with inequations. Thus we have got to find a way to introduce non-degeneracy conditions that both reflects (as closely as possible) the geometric meaning of the condition (i.e. to avoid some degenerate cases) and expresses it by means of equations.

2 Different ways...

Traditionally (at least since [5]) the refutation of a given property $f = 0$, i.e. three points not aligned, two points not coincident, etc., is handled as an equation by adding some auxiliary variable t and considering the equation $f \cdot t - 1 = 0$ as representing $\neg\{f = 0\}$, emulating Rabinowitsch trick[3].

[3] Originally, this "trick" was designed to deduce the general Hilbert Nullstellensatz (about the vanishing of a polynomial f on the solution set of some system of polynomial equations) from the Weak Nullstellensatz (about the existence of solution for a systems of equations), by introducing some extra variable and considering $f \cdot t - 1$. Thus, if f vanishes on some solution set, $f \cdot t - 1$ can not have a solution on this set. Thus, the Weak Nullstellensatz implies that 1 is a combination of the polynomials in the set of equations and of $f \cdot t - 1$. Next, replace t by $1/f$ in this expression of

For instance, if we want to declare that points $A(x_1, x_2), B(x_3, x_4), C(x_5, x_6)$ should not lie on a line, we can proceed as follows. First notice that $(x_1x_4 - x_1x_6 - x_2x_3 + x_2x_5 + x_3x_6 - x_4x_5) = 0$ is the condition for the alignment of the three points A, B, C. Then notice that every solution to the equation $(x_1x_4 - x_1x_6 - x_2x_3 + x_2x_5 + x_3x_6 - x_4x_5) \cdot t - 1 = 0$ has to verify $(x_1x_4 - x_1x_6 - x_2x_3 + x_2x_5 + x_3x_6 - x_4x_5) \neq 0$, i.e. that A, B, C are not aligned. And conversely, for every choice of coordinates for A, B, C such that A, B, C are not aligned, we will have $(x_1x_4 - x_1x_6 - x_2x_3 + x_2x_5 + x_3x_6 - x_4x_5) \neq 0$, and, then, there will be a value of $t = 1/(x_1x_4 - x_1x_6 - x_2x_3 + x_2x_5 + x_3x_6 - x_4x_5)$ such that $(x_1, x_2, x_3, x_4, x_5, x_6, t)$ is a solution to $(x_1x_4 - x_1x_6 - x_2x_3 + x_2x_5 + x_3x_6 - x_4x_5) \cdot t - 1 = 0$.

Speaking in general, if $f(x_1, \ldots, x_n) \cdot t - 1 = 0$, then the inequality $f(x_1, \ldots, x_n) \neq 0$ holds, and conversely, if $f(x_1, \ldots, x_n) \neq 0$, then we can take $t = 1/f(x_1, \ldots, x_n)$ and so $f(x_1, \ldots, x_n) \cdot t - 1 = 0$. Therefore, if we are dealing with an ideal of hypotheses H, and we want to add the condition $\neg\{f = 0\}$ to it, we must consider the ideal $H + (f \cdot t - 1)$.

It is easy to generalize this procedure for the case of having to negate the conjunction or the disjunction of several conditions. Clearly, the assertion $\neg\{f_1 = 0 \wedge \ldots \wedge f_r = 0\}$ translates into $\neg\{f_1 = 0\} \vee \ldots \vee \neg\{f_r = 0\}$. So, we will have to deal with the ideal $H + (f_1 \cdot t_1 - 1) \cdots (f_r \cdot t_r - 1)$ (although we could also employ the same slack variable t for all the conditions). Regarding $\neg\{f_1 = 0 \vee \ldots \vee f_r = 0\}$, it can be expressed as $\neg\{f_1 = 0\} \wedge \ldots \wedge \neg\{f_r = 0\}$. Therefore, the ideal which we must consider is $H + (f_1 \cdot t_1 - 1, \ldots, f_r \cdot t_r - 1)$. In this case, we cannot use a single auxiliary variable t. However, in practice we can express $\neg\{f_1 = 0\} \wedge \ldots \wedge \neg\{f_r = 0\}$ as $(f_1 \cdots f_r \cdot t - 1)$.

But imposing, as part of the hypotheses, the avoidance of some condition $f = 0$, i.e. requiring that our hypothesis solution set $V(H)$ verifies $f \neq 0$, can be done in a different, although quite similar, way. Say, by considering the Zariski closure of the difference $V(H) \setminus V(f)$, i.e. by considering as new hypotheses the polynomial equations expressing the smallest set that verifies the given hypotheses and not the condition $f = 0$. As it is well known ([2]) this smallest algebraic set is defined, when H is radical, by means of the zeroes of the quotient ideal $(H) : (f)$. On the other hand let us remark that, geometrically, it is the same to consider $V(f)$ or $V(f^n)$, for whatever n-th power of f; thus, we can deal simultaneously with the collection of ideals $(H) : (f)^n$. And this is, precisely, the idea of saturation of the ideal H by the ideal (f). More generally, let I, J be ideals of a polynomial ring $K[X]$. Recall that $I : J = \{x \mid xJ \subset I\}$. Then, the *saturation of I by J* is defined as $\texttt{Saturate}(I, J) = I : J^\infty = \cup_n (I : J^n)$ (see [2,3]), and it satisfies $V(\texttt{Saturate}(I, J)) = \overline{V(I) \setminus V(J)}$. In summary, the other option we are considering here to include non-degeneracy conditions $\neg\{f = 0\}$ is to saturate the ideal of hypotheses by the condition f. Again, it is straightforward the generalization of this idea of saturation to the case of several conditions (see [3]).

This second option, by means of saturation, could seem, at first glance, more sophisticated than the first option, the one relying on the implementation of

1 and clear denominators to obtain some power of f as a combination of the given polynomials (see [6]).

Rabinowitsch trick. But there is not a big difference. In fact, [3, Proposition 6 and Corollary 2 of Appendix] as well as in [2, Theorem 14 of Chapter 4, Section 4] show that the saturation of ideal I by polynomial f is equal to the result of eliminating the variable t in the ideal $I + (f \cdot t - 1)$.

Thus, the actual dilemma is: do we want to add non degeneracy conditions as in Rabinowitsch trick, by carrying around an extra, alien, variable, which should be eliminated at the end of the theorem proving or discovery process; or should we deal, from the beginning, with the non degeneracy condition expressed in terms of the "real" variables of our statement, by saturation? Warning: elimination implies, in geometric terms, taking the Zariski closure of the projection [2]; while, as reported above, saturation involves taking the closure of the difference of two sets by eliminating the variable t in the ideal $I + (f \cdot t - 1)$.

Thus, we can summarize the characteristic feature of both approaches as follows: do we want to deal with inequations by doing an early closure or elimination step or is it better to postpone it until the end of the proving method?

3 . . . different consequences

We think the analysis of the different ways of introducing non-degeneracy conditions is an important issue, seemingly forgotten by the automated proving and discovery community. Important because the decision about which one of the two described approaches is more suitable (i.e more efficient to prove or more successful to discover interesting theorems) could have relevant consequences.

Thus, in [3] it is presented one specific example of how both methods differ in a common context, yielding, if non-degeneracy is introduced by means of Rabinowitsch trick, an interesting theorem discovering the conditions for the orthic triangle of a given triangle with non-collinear vertices, to be equilateral. On the other hand, if the non-collinearity of the vertices is introduced by saturation, there is no discovery at all. It is remarkable that the approaches to discovery in [3] and the one we employ here are slightly different.

Generally speaking, we have the following relation between both approaches:

Proposition 1. *Let H be the ideal of hypotheses of a given statement, and let $f = 0$ be a new degenerate condition. Let $I_1 = H + (f \cdot t - 1)$ and let $I_2 = \mathtt{Saturate}(H, f)$ be the new ideals of hypotheses corresponding to the two possibilities of introducing the non-degeneracy conditions. Then $I_2 \subseteq I_1$.*

Proof. In fact, as mentioned above, $\mathtt{Saturate}(H, f) \subset H + (f \cdot t - 1)$, since the saturation is equal to the elimination of the variable t in the ideal at the right side, and, thus, it is contained in it.

Example 1. The following trivial example $H = (0), f = x$ shows that, in general, the inclusion $\mathtt{Saturate}(H, f) \subset H + (f \cdot t - 1)$ is strict, since in this case $\mathtt{Saturate}(H, f) = (0)$ and $H + (f \cdot t - 1) = (x \cdot t - 1)$.

Now let us recall (c.f. [7]) that a statement $H \Longrightarrow T$ is called *generally true* if it is not zero the elimination of all variables, except the independent ones ruling our statement (say $x_1, \ldots, x_s$) , in the ideal $H + (T \cdot t - 1)$; and it is called *generally false* if it is not zero the elimination of all variables, except the independent ones ruling our statement, in the ideal $H + T$. Obviously, if two ideals J, I verify $J \subseteq I$, the result of adding to each one the same new ideal and intersecting the sum with $K[x_1, \ldots, x_s]$ will preserve the inclusion. So, if the result of some elimination yields zero for an ideal, the same happens for all smaller ideals contained in it. Thus:

Corollary 1. *Notation as above. If the statement $I_1 \Longrightarrow T$ is not generally true, then, the same happens to the statement $I_2 \Longrightarrow T$; and if $I_1 \Longrightarrow T$ is not generally false, then, the same happens to the statement $I_2 \Longrightarrow T$. That is,*

$$\{I_2 \Longrightarrow T \text{ generally false}\} \Longrightarrow \{I_1 \Longrightarrow T \text{ generally false}\} \qquad \text{(GF)}$$

$$\{I_2 \Longrightarrow T \text{ generally true}\} \Longrightarrow \{I_1 \Longrightarrow T \text{ generally true}\}. \qquad \text{(GT)}$$

We can informally say that I_2 can be a more "exigent" hypotheses set for a statement to be true than I_1, for proving (see statement (GT)) and for discovery (see (GF)).

It is also interesting to remark that, in the theory of automated discovery, the polynomials in $(H + T) \cap K[x_1, \ldots, x_s]$ are seen as necessary conditions for the thesis to be true (see [7]), and must be added to the set of hypotheses. As we stated before,

$$(I_2 + T) \cap K[x_1, \ldots, x_s] \subseteq (I_1 + T) \cap K[x_1, \ldots, x_s]. \qquad (1)$$

So, the Rabinowitsch trick is able to provide more additional conditions for discovery than the saturation, increasing the differences between the two sets of hypotheses.

Example 2. Again, we present a simple example to illustrate that the inclusion in (1) is, in general, strict. Consider the ideal $H = (x + y)$, and let x be an independent variable, while y is non-independent. Consider also the non-degeneracy condition $x \neq 0$, and the thesis $x \cdot y = 0$. With the above notation, $I_1 = (x + y, x \cdot t - 1)$ and $I_2 = (x + y)$. If we add T to our ideals and eliminate variables y, t, we obtain, respectively, (x^2) and (1).

Situations as the previous one can happen, as already remarked, because the saturation method involves the early consideration of the closure. For instance, considering Example 1, the saturation of (0) by (x) is (0), and its zero set –the whole line– surely includes a point where $x = 0$, although we wanted to avoid such instances!

4 An example

Finally, we would like to present a particular example in order to show in some detail the described situation. Our example is based on the already cited theorem

about the orthic triangle, but with a different approach to the one presented in [3]–the conclusions will be different too. We wish to show that the orthic triangle associated to an equilateral triangle is also equilateral (see Fig. 1). Since we want to address the theorem from the point of view of discovery, we decide to ignore the hypothesis about the original triangle being equilateral and take a completely arbitrary one, with the purpose of obtaining the condition for it to be equilateral as necessary.

So, we take $A = (0, 0)$, $B = (u_1, 0)$ and $C = (u_2, u_3)$ as the vertices of the main triangle, and set $D = (u_2, 0)$, $E = (x_1, x_2)$ and $F = (x_3, x_4)$ the vertices of the orthic one. We force the segments $\overline{AE}$ and $\overline{BC}$ to be perpendicular, as well as E to be collinear with B and C; analogously, $\overline{BF}$ and $\overline{AC}$ must be perpendicular, and F must be aligned with A and C. By construction, it is obvious that the point D is collinear with A and B, and that $\overline{CD}$ is perpendicular to $\overline{AC}$. As for our desired conclusion, we state it using two polynomials, each one forcing two sides of the orthic triangle to have the same length. We deal with them separately.

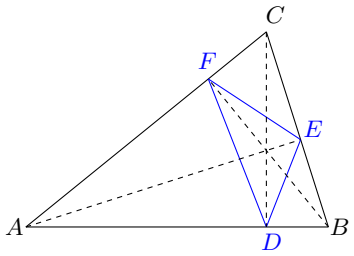


Fig. 1. Orthic triangle

Besides the main hypotheses, we choose as non-degeneracy conditions those which force the triangle ABC not to collapse to a line, i.e., $u_1 \neq 0$ and $u_3 \neq 0$. These two conditions can be summarised in just one: $u_1 u_3 \neq 0$. We introduce this new hypothesis in the two ways described above: by adding to the primitive hypotheses the polynomial $u_1 u_3 \cdot t - 1$ or by saturating the hypotheses by the ideal $(u_1 u_3)$. It can be shown that the resulting ideals are different in the sense of Proposition 1. Moreover, with the choice of saturation it is easy to see that the assertion stating that the orthic triangle is equilateral is generally false. Thus, by Corollary 1, it is also generally false with the choice of Rabinowitsch trick. After that, we continue to find new hypotheses by imposing the thesis –the orthic triangle to be equilateral– to be true and eliminating the non-independent variables.

As we expected, we obtain necessary conditions for the thesis to verify related, in both cases, with the main triangle to be equilateral. What is interesting is that we find here another example of the inclusion 1 being strict. On the one

hand, using saturation we obtain as generators the polynomials

$$-u_1(u_1-2u_2)(u_1u_2-u_2^2+u_3^2)(-u_1u_2+u_2^2+u_3^2) \tag{2}$$

and

$$u_2(-u_1^2+u_2^2+u_3^2)(-u_1^2u_2+2u_1u_2^2+2u_1u_3^2-u_2^3-u_2u_3^2), \tag{3}$$

corresponding to the two polynomials in the thesis ideal. On the other hand, using the traditional approach, we find a very similar output: two polynomials, the first of them having the same factors as (2), except u_1, and the second one equal to (3). As a consequence of the choice of the non-degeneracy condition, the factor u_1 does not give any new information. Is is easy to check that if the triangle ABC is equilateral, the polynomials vanish. Nevertheless, there are more possible configurations of that triangle which make them vanish, and should be carefully studied.

The next step consists on including the "discovered" polynomials in the set of hypotheses, and check if the theorem holds under these new conditions. Using the approach of saturation, the calculations present no difficulties, and we easily check that the theorem is generally true–more non-degeneracy conditions are needed. Regarding the traditional approach, one of the polynomials in the thesis ideal follows directly from the augmented set of hypotheses, without needing more non-degeneracy conditions. As for the other one, no calculations are needed: in particular, with the approach of saturation, it holds when taking as hypotheses just the primitive ones and (3). Therefore, and since polynomial (3) appears again with this focus, Corollary 1 assures that the statement is also generally true employing Rabinowitsch trick.

Thus, we have discovered two "different" (in a formal sense) theorems concerning the posed problem, although their geometrical meanings are the same.

The calculations have been carried out using the software Singular [4] in the FinisTerrae 2 supercomputer.

Acknowledgement

The first author was supported by Ministerio de Economía y Competitividad (Spain), grant MTM2013-43687-P (European FEDER support included) and by Xunta de Galicia, grant GRC2013-045 (European FEDER support included). The third author was partially supported by the Spanish Research Project 'Construcciones Algebra-geométricas: fundamentos, algoritmos y aplicaciones', grant MTM2014-54141-P (European FEDER support included). We also gratefully thank CESGA (Centro de Supercomputación de Galicia, Santiago de Compostela, Spain) for providing access to the FinisTerrae 2 supercomputer.

References

1. Chou, S.C.: Mechanical geometry theorem proving, Mathematics and its Applications, vol. 41. D. Reidel Publishing Co., Dordrecht (1988), with a foreword by Larry Wos

2. Cox, D.A., Little, J., O'Shea, D.: Ideals, varieties, and algorithms. An introduction to computational algebraic geometry and commutative algebra. 4th revised ed. Undergraduate Texts in Mathematics, Springer, Cham (2015)
3. Dalzotto, G., Recio, T.: On protocols for the automated discovery of theorems in elementary geometry. J. Automat. Reason. 43(2), 203–236 (2009)
4. Decker, W., Greuel, G.M., Pfister, G., Schönemann, H.: SINGULAR 4-0-2 — A computer algebra system for polynomial computations. `http://www.singular.uni-kl.de` (2015)
5. Kapur, D.: A refutational approach to geometry theorem proving. Artificial Intelligence 37(1-3), 61–93 (1988)
6. Rabinowitsch trick. Bronawell, W. D. (originator), Encyclopedia of Mathematics, `http://www.encyclopediaofmath.org/index.php?title=Rabinowitsch_trick&oldid=12019`
7. Recio, T., Vélez, M.P.: Automatic discovery of theorems in elementary geometry. J. Automat. Reason. 23(1), 63–82 (1999)
8. Wu, W.T.: On the decision problem and the mechanization of theorem-proving in elementary geometry. Sci. Sinica 21(2), 159–172 (1978), Reprinted in: Automated theorem proving: After 25 years (Bledsoe, W. W., Loveland, D. W., eds.), Contemporary Mathematics, 29. AMS, Providence, RI, pp. 213–234, 1984

Portfolio Methods in Theorem Proving for Elementary Geometry*

Extended Abstract

Vesna Marinković[1], Mladen Nikolić[1], Zoltán Kovács[2], and Predrag Janičić[1]

[1] Faculty of Mathematics, University of Belgrade, Serbia
[2] The Private University College of Education of the Diocese of Linz, Austria

Abstract. Portfolio problem solving is an approach in which for an individual instance of a specific problem, one particular solver or one particular solving technique is selected among several available ones and used. This approach has given many successes in recent years, in many areas, including automated theorem proving, especially in SAT solving. We report on our first attempts at using the portfolio approach in proving geometry theorems. There are several challenges in this, including: domains of the popular geometry provers are not the same (some provers support some constructs and some not), there is no standard format for storing geometry theorems, available geometry provers are implemented in different programming languages, etc. However, the biggest challenge is defining a set of relevant features that characterize a specific theorem and that should serve for choosing an appropriate prover. Our first preliminary experiments are promising and we believe there is a big room for further progress in this direction.

1 Introduction

Theorems in geometry are often given in terms of a construction and a goal [1,2]. Hence, conjectures of this form can also be seen as correctness statements about specific constructions (i.e., statements that the constructions meet the given constraints). For proving such theorems, one can use a number of automated theorem provers developed so far.

Portfolio problem solving is an approach in which for an individual instance of a specific problem, one particular solver or one particular solving technique is selected among several available ones (based on some specifics of the input instance) and used. An expected gain is that for a number of future input instances the percentage of solved problems will be higher than for any individual solver and/or that overall solving time will be lower than for any individual solver. In that, it is assumed that choosing of the appropriate solvers takes negligible computing time (compared to solving time). This approach has given many successes in recent years, in many areas, including automated theorem proving

* The first, the second, and the fourth authors have been partly supported by the grant 174021 of the Ministry of Science of Serbia.

and related problems, and especially in SAT solving [3,4]. One of the most important challenges in building successful portfolio system is identifying suitable *features*—properties of input instances that influence performance of individual solvers.

The portfolio approach in proving geometry theorems has not been used so far. However, there are already systems that need and actually can support portfolio theorem proving in geometry. For instance, tools GeoGebra [5] and GCLC [6] have several theorem provers built-in and should be able to automatically choose an appropriate one given a goal to be proved. Using the portfolio approach in geometry seems like a promising idea, but there are several challenges with integrating several automated geometry provers within one portfolio system (challenges not present in some other domains, such as SAT). One problem is that their domains are not the same—some provers support some constructs and some not (say, intersections of two circles). The second problem is that there is no standard format for storing geometry theorems. The third problem is that available geometry provers are implemented in different programming languages. However, probably the hardest challenge is identifying a set of appropriate features, harder than in problems such as SAT. In this paper we present our first experiences and experiments with portfolio proving in geometry. The steps of portfolio development we will follow consist of choosing: the corpus of instances, the set of provers, the set of instance features, and the prover selection mechanism.

2 Corpora of Theorems

The corpus of theorems that we consider are theorems obtained as correctness theorems for solutions of 560 triangle construction problems from Wernick's list [7]. The solutions (i.e. the constructions) and the corresponding correctness theorems were generated automatically by a system ArgoTriCS [8,9,10].

In order to prove that the generated construction is correct, one has to prove that the points given by the problem setting (for each three) are indeed the corresponding points of the constructed triangle ABC. One example of such (automatically generated) conjecture follows. It states that for a given construction of the triangle ABC given its vertex A, circumcenter O and midpoint M_a of the side BC, the point M_a is indeed a midpoint of the side BC of the constructed triangle ABC (this conjecture follows from problem 28:$\{A, O, M_a\}$ from Wernick's corpus). The list of primitive constructions supported in ArgoTriCS is given in the Appendix.

Example 1. Given a point A, a point O and a point M_a, construct the triangle ABC.

Construction:

1. Using the point A and the point M_a, construct a point G (rule W01);
2. Using the point O and the point G, construct a point H (rule W01);
3. Using the point A and the point H, construct a line h_a (rule W02);

4. Using the point A and the point O, construct a circle $k(O, C)$ (rule W06);
5. Using the point M_a and the line h_a, construct a line a (rule W10);
6. Using the circle $k(O, C)$ and the line a, construct a point C and a point B (rule W04);
7. Using the point B and the point C, construct a point $\_M_a$ (rule W01);
8. Using the point A and the point C construct the line $\_b$ (rule W02);
9. Using the point C and the point A, construct a point $\_M_b$ (rule W01);
10. Using the point B and the point C construct the line $\_a$ (rule W02);
11. Using the point $\_M_a$ and the line $\_a$ construct the line $\_m_a$ (rule W10);
12. Using the point $\_M_b$ and the line $\_b$ construct the line $\_m_b$ (rule W10);
13. Using the line $\_m_a$ and the line $\_m_b$ construct the point $\_O$ (rule W03);

Statement: Prove that the point M_a is identical to the point $\_M_a$.

The illustration of the generated construction is given in Figure 1.

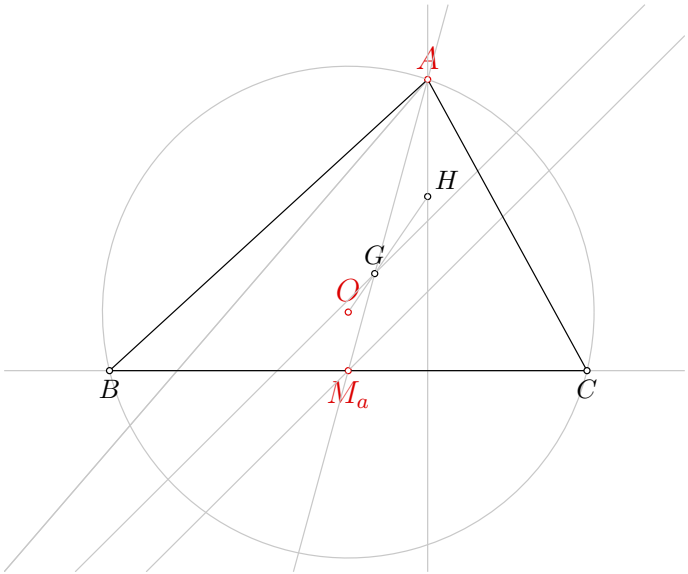


Fig. 1. Illustration of the construction

We tried to prove all conjectures generated this way using four theorem provers: three provers integrated in GCLC [6] (there is support for three methods: the area method, simple Wu's method, and the Gröbner bases method) and OpenGeoProver [11] (it currently provides support only for simple Wu's method) with a timeout set to 5 minutes. However, some conjectures are beyond the scope of specific provers (for instance, conjectures which include intersections of circles in the area method), for some of them the prover concludes that it

can neither prove nor disprove, while for some the timeout is reached without getting any answer. Here we focus only on solvable construction problems that ArgoTriCS succeeded to solve and for each of these problems 3 conjectures are generated.[3] There were 828 conjectures altogether, 537 theorems were proved by at least one prover, Wu's method implemented within OpenGeoProver proved 486 conjectures, the same method in GCLC proved 295, area method 350, while the Gröbner bases method proved 264 conjectures. So, the prover that showed the best performance is Wu-OpenGeoProver, but the total number of conjectures proved is 537 which shows that there is a room for the portfolio approach.

3 Set of Features and Portfolio Design

A construction will be represented by a vectors of some numbers—values of selected features. The basic features should reflect complexity of the construction and of the construction steps used. The additional statistics over these basic features will be inspired by similar statistics used in SAT portfolios. For most of them there is no corresponding geometrical meaning—still, we consider them mimicking the SAT portfolios and hoping that we might capture some relevant property not covered by the basic features.

We define a *construction graph* as a directed labeled graph consisting of a set of labeled nodes and a set of edges. Each node corresponds to one step of the construction and is labeled by a rule applied and an object constructed in that step. Edges are pairs of nodes such that first node corresponds to the step in which an object is derived which is used in the step corresponding to the second node. The construction graph of the construction given in Example 1 is illustrated in Figure 2.

We consider the following features over the construction graph:

- the number of nodes in the graph,
- the number of nodes whose in-degree is zero,
- the number of edges in the graph,
- the ratio of number of nodes and edges and its reciprocal,
- the ratio of longest path length and number of nodes and its reciprocal,
- the ratio of longest path length and number of edges and its reciprocal,
- the node degree statistics,
- the node in-degree statistics,
- the node out-degree statistics,
- the number of nodes whose in-degree is zero and that are not labeled by A, B, or C,
- the rule application frequencies (normalized) and their statistics,
- the object type frequencies (normalized) and their statistics,
- statistics of normalized frequencies of referencing each specific object,
- the statement size.

[3] Some of these conjectures are trivial—in cases when one of the points given by the problem setting is one of the vertices $\{A, B, C\}$.

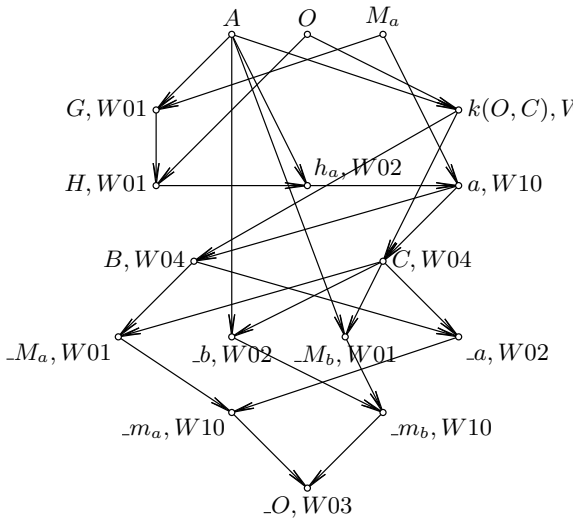


Fig. 2. Example construction graph

We consider the following statistics over the set of basic features: the mean, variation coefficient, minimum, maximum, and entropy of the distributions of those statistics.

Alltogether, currently there are 67 features. For instance, the construction graph illustrated in Figure 2 has 17 nodes, 28 edges, the mean of in-degrees of its nodes is 1.647, while their variation is 0.581.

The first portfolio system for the prover selection that we formulate is based on the k nearest neighbors technique [12] and is similar to existing portfolio systems for SAT [3,4]. The method assumes that the training set is available that consists of feature vectors and proving times for each prover. If the prover did not finish in some fixed predetermined cutoff time (5 minutes in our case), then for that instance ten times the cutoff time is recorded. For a given instance to be proven, its feature vector is computed and k instances from the training set, with feature vectors closest to the input one, are found. The prover with the best performance on these k instances is invoked for the input instance.

The second portfolio system is based on multinomial logistic regression (MLR) [13] as a second natural approach to multiclass classification, where each prover is treated as a separate class and MLR models the probability over provers for each instance. In the training phase, instead of using clear-cut assignment of provers to instances, as is usual in classification, MLR allows assigning relative desirability of different provers based on the amount of time provers saved with respect to the cutoff time. In the exploitation phase, the feature vector is computed for the input instance, and the prover with highest probability predicted by the MLR model is invoked.

4 Results and Portfolio Performance

We evaluate the above two portfolios on our set of conjectures. Both the proving and the exploitation cutoff are set to be 300 seconds. Evaluation is performed by 5-fold nested crossvalidation [14]. This procedure is generally used for evaluation of machine learning models with metaparameters (e.g. k in k nearest neighbors). One of its qualities is that the whole data set is used for evaluation and yet training and test instances are always separated. Since the crossvalidation splits the training set randomly into subsets (each of which will at one point serve as the testing set, while the others will serve the training set), there might be some variation in the results due to particular split. In order to avoid that, we average the results over 100 such splits, having performed crossvalidation for each of them. As the reference prover, we used the *best prover* among component provers of portfolios and the *virtual best prover* which represents the best performance one could hope for with the given provers. The results are given in Table 1. The proposed portfolios perform very well—they cover around 80% of the gap between reference solvers both in terms of number of solved instances and in terms of proving time. Two portfolio approaches perform roughly the same. The CPU time needed by the portfolios to select a prover for an input instance is less than 0.001 second, so it is virtually negligible. The CPU time needed for computing feature values for input instance is also negligible (less than 0.01s).

Prover	# proved	σ	% increase	Time (s)	σ	% decrease
Best prover	486.0	0	0	15831	0	0
MLR portfolio	525.4	2.44	77	4250	724	77
k-nn portfolio	526.5	2.47	79	3776	731	80.1
Virtual best prover	537.0	0	100	776	0	100

Table 1. Results of evaluation of two portfolios compared to two reference provers (best single prover and, idealized, virtual best prover). For each portfolio we provide average number of instances proved and its standard deviation over 100 runs, percentage of gap between two reference provers which is covered, average time per one run on whole corpus and its standard deviation over 100 runs, and percentage of gap between two reference solvers which is covered.

Aside of selecting an appropriate prover, in practical contexts, it is important to provide some information to the user regarding the time required for the prover to finish. The simplest requirement would be to estimate whether the prover will finish in the given cutoff time. We model this problem as a classification problem. One class consists of conjectures for which the prover finishes in 300 seconds and the other class consists of all other instances. For classification we use regularized logistic regression and evaluate it using nested crossvalidation. We enriched the feature set by adding products of features to the base feature set. That way we model feature interactions—presence of product $x_i x_j$ allows the change in feature x_i to contribute to the model value proportionally to x_j.

The classification accuracy obtained ranges from 94% to 98%, depending on the prover, which is very high (especially considering that the classes are relatively well balanced), meaning that for future conjectures we can predict if they will be proven in the given time with very high probability.

We also deal with prediction of prover runtime and its logarithm. Considering that provable formulae are easily identified by classification, we evaluate runtime prediction only on the subset of conjectures which are proven within cutoff time. For prediction we use ridge regression and for evaluation we again use nested crossvalidation. The root mean square error [13] for runtime prediction ranges from 0.07 to 3.07s, depending on the prover, which is good, except for one prover for which it is 13.16s. The root mean square error for prediction of logarithm of runtime ranges from 0.04 to 0.39s which we consider to be a good result. In almost all cases, r^2 (coefficient of determination [15]) testifies that the variance of prediction by the model is reduced more then 80% compared to the variance of prediction using the test set mean as a predictive model, meaning that the learning process was successful. We conclude that for future conjectures we expect to be able to predict proving time roughly with error of several seconds, and to predict the order magnitude of the proving time rather accurately.

5 Current and Future Work

As we hoped, the portfolio approach in theorem proving in geometry gives good performance, at least in the domain of theorems obtained from construction problems. There are gains, even without any geometrical considerations, apart from considerations within designing the set of features.

The experiments performed do not exhaust opportunities in this domain. The GeoGebra system currently has at its disposal several geometry provers [16,17] and could, therefore, benefit from our portfolio approach (a response will be more quickly given to the user). Currently, we are constructing a similar portfolio over those provers. Preliminary evaluation on a corpus of classroom oriented geometry statements yields encouraging results. We also plan to consider other corpora of problems, such as those obtained on the basis of Connelly's corpus of construction problems [18].

Portfolio systems are built expecting gains in proving performance. However, one may be interested in explaining a deeper relationship between feature values and the provers which were selected. Considering the high dimension of feature vectors and statistical nature of the models, such relationships are very hard to establish, but we will look at this problem it in future. In this context, we intend to explore performance of portfolios using subsets of currently used 67 features. Feature selection techniques used in machine learning may be useful in finding most relevant ones.

It would be also interesting to explore if some features can be used for setting parameters of the provers (in order to make it more efficient for that particular instance). However, most of provers used in geometry do not have adjustable parameters. Instead of setting parameters, one can use the portfolio in the phase

of constructing a theorem (for choosing a most promising formulation), but such approach is yet to be considered.

References

1. Chou, S., Gao, X., Zhang, J.: Machine Proofs in Geometry. World Scientific, Singapore (1994)
2. Chou, S.C.: Mechanical geometry theorem proving. Kluwer Academic Publishers Norwell, MA, USA (1987)
3. Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Non-model-based algorithm portfolios for SAT. In: Theory and Applications of Satisfiability Testing (SAT). (2011)
4. Nikolić, M., Marić, F., Janičić, P.: Simple algorithm portfolio for sat. Artificial Intelligence Review (2012) 1–9
5. Hohenwarter, M.: GeoGebra: Ein Softwaresystem für dynamische Geometrie und Algebra der Ebene. Master's thesis, Paris Lodron University, Salzburg, Austria (2002)
6. Janičić, P.: GCLC – A Tool for Constructive Euclidean Geometry and More than That. In Takayama, N., Iglesias, A., Gutierrez, J., eds.: Proceedings of International Congress of Mathematical Software (ICMS 2006). Volume 4151 of Lecture Notes in Computer Science., Springer-Verlag (2006) 58–73
7. Wernick, W.: Triangle constructions vith three located points. Mathematics Magazine **55** (1982) 227–230
8. Marinković, V.: ArgoTriCS – Automated Triangle Construction Solver. Journal of Experimental & Theoretical Artificial Intelligence (2016)
9. Marinković, V., Janičić, P.: Towards Understanding Triangle Construction Problems. In Jeuring et al., J., ed.: Intelligent Computer Mathematics - CICM 2012. Volume 7362 of Lecture Notes in Computer Science., Springer (2012)
10. Marinković, V., Janičić, P., Schreck, P.: Computer Theorem Proving for Verifiable Solving of Geometric Construction Problems. In: 10th International Workshop on Automated Deduction in Geometry (ADG 2014). Volume 9201 of Lecture Notes in Computer Science., Springer (2015) 72–93
11. Marić, F., Petrović, I., Petrović, D., Janičić, P.: Formalization and implementation of algebraic methods in geometry. In Quaresma, P., Back, R.J., eds.: Proceedings First Workshop on CTP Components for Educational Software, Wrocław, Poland, 31th July 2011. Volume 79 of Electronic Proceedings in Theoretical Computer Science., Open Publishing Association (2012) 63–81
12. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer New York Inc. (2001)
13. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press (2012)
14. Krstajic, D., Buturovic, L.J., Leahy, D.E., Thomas, S.: Cross-validation pitfalls when selecting and assessing regression and classification models. Journal of Cheminformatics **6** (2014) 1–15
15. Rao, C.R.: Linear Statistical Inference and its Applications. Wiley (1973)
16. Botana, F., Hohenwarter, M., Janičić, P., Kovács, Z., Petrović, I., Recio, T., Weitzhofer, S.: Automated theorem proving in GeoGebra: Current achievements. Journal of Automated Reasoning **55** (2015) 39–59 `http://dx.doi.org/10.1007/s10817-015-9326-4`.

17. Kovács, Z.: The portfolio prover in GeoGebra 5. In Botana, F., Quaresma, P., eds.: Proceedings of the 10th International Workshop on Automated Deduction in Geometry (ADG 2014), 9-11 July 2014. University of Coimbra, Portugal (2014) 191–205
18. Connelly, H.: An Extension of Triangle Constructions from Located Points. Forum Geometricorum **9** (2009) 109–112

A List of primitive constructions used by ArgoTriCS

1. Given points X, Z, and W, and a rational number r one can construct a point Y for which holds: $\overrightarrow{XY}/\overrightarrow{ZW} = r$; NDG condition is that the points Z and W are distinct[4].
2. Given points X and Y one can construct a line XY; DET condition is that the points X and Y are distinct[5].
3. Given two lines, it is possible to construct their intersection point; NDG condition is that lines are not parallel, while DET condition is that lines are not equal.
4. Given a line and a circle, it is possible to construct their intersection points; NDG condition is that they do intersect.
5. Given a line, a circle, and one intersection point, it is possible to construct their second intersection point; NDG condition is that the line and the circle intersect.
6. Given two distinct points X and Y it is possible to construct a circle $k(X, Y)$ centered at point X which passes through the point Y; NDG condition is that the points X and Y are distinct.
7. Given two circles, one can construct their intersection points; NDG condition is that the circles intersect, while DET condition is that the circles are distinct.
8. Given two circles and one intersection point, one can construct their second intersection point; NDG condition is that the circles intersect, while DET condition is that the circles are distinct.
9. Given points X and Y one can construct a circle with diameter $\overline{XY}$; NDG condition is that points are distinct.
10. Given a point X and a line p one can construct a line q which passes through the point X and which is perpendicular to the line p.
11. Given a line p and a point X which does not belong to the line p one can construct a circle k centered at point X which touches the line p; NDG condition is that the point X does not belong to the line p.
12. Given a circle k and a point X outside the circle k, one can construct two tangents from the point X to the circle k; NDG condition is that the point X is outside the circle k.

[4] NDG conditions denote non-degeneracy conditions which guarantee that the constructed objects indeed exist.

[5] DET conditions denote determination conditions which enable that the constructed objects are uniquely determined.

13. Given a circle k, a point X outside the circle k, and one tangent from the point X to the circle k, one can construct the second tangent from the point X to the circle k; NDG condition is that the point X is outside the circle k.
14. Given points X and Y one can construct bisector of the segment $\overline{XY}$; NDG condition is that points X and Y are distinct.
15. Given a point X, a line p and a rational number r, one can construct a line which is an image of the line p in homothety with center at the point X with coefficient r.
16. Given a point X and a line p one can construct a line which passes through the point X and which is parallel to the line p.
17. Given points X and Y and an angle α one can construct the line q such that that $\angle(\overline{XY}, q) = A \cdot \alpha/2^B + C \cdot \pi/2^D$ holds.[6]
18. Given points X and Y and an angle α one can construct a line q such that $\angle(q, \overline{XY}) = A \cdot \alpha/2^B + C \cdot \pi/2^D$ holds.
19. Given points X, Y, and Z one can construct a point W which is harmonic conjugate of Z with respect to X and Y; NDG conditions are that points X and Y are distinct, points Y and Z are distinct and point Y is not a midpoint of the segment XZ.
20. Given points X and Y and an angle α one can construct a locus of points such that the segment $\overline{XY}$ can be seen at angle $A \cdot \alpha/2^B + C \cdot \pi/2^D$.
21. Given a point X and a line p one can construct a foot of the perpendicular from the point X to the line p.
22. Given a point X and a circle k_1 one can construct a circle k_2 centered at point X which internally touches the circle k_1; NDG conditions are that the point X is inside the circle k_1 and that the point X is not the center of the circle k_1.
23. Given a point A, a circle C, and a rational number r one can construct an image of the circle C in homothety with a center at the point A with coefficient r.

[6] Here we use the fact that if an angle α is constructible, then the angle $K_1 \cdot \alpha/2^{K_2} + K_3 \cdot \pi/2^{K_4}$ is constructible as well, where K_1, K_2, K_3, and K_4 are natural numbers.

On a Certain Class of Cubic Surfaces Related to the Simson–Wallace Theorem

Extended Abstract

Pavel Pech

Faculty of Education, University of South Bohemia,
České Budějovice, Czech Republic
pech@pf.jcu.cz

Abstract. Properties of a cubic surface which is related to the Simson–Wallace theorem are studied. Given a skew quadrilateral in the real Euclidean 3D-space $\mathbb{E}^3$, then the locus of a point whose orthogonal projections onto the sides of the quadrilateral are coplanar is a cubic surface. Properties of this locus such as decomposability, structure of lines on the surface and existence of singular points are investigated using computer aided analytical method.

Keywords: Simson–Wallace locus. Skew quadrilaterals. Cubic surfaces.

1 Introduction

The well-known Simson–Wallace theorem reads [5]:

Feet of perpendiculars from a point P onto the lines AB, BC and CA of a triangle ABC are collinear iff P lies on the circumcircle of ABC.

There are several generalizations of the Simson–Wallace theorem, see [2], [3], [5], [9], [10]. The following is a generalization on skew quadrilaterals [6–8]:

Theorem 1. *Let K, L, M, N be orthogonal projections of a point P onto the sides AB, BC, CD, AD of a skew quadrilateral $ABCD$ respectively. Let $A = (0,0,0)$, $B = (a,0,0)$, $C = (b,c,0)$ and $D = (d,e,f)$. Then the locus of $P = (p,q,r)$ such that the tetrahedron $KLMN$ has a constant volume s is a cubic surface F in (2).*

Outline of the proof [8]: Suppose that $acf \neq 0$ since otherwise the quadrilateral is planar. Denote $K = (k_1,0,0)$, $L = (l_1,l_2,0)$, $M = (m_1,m_2,m_3)$, $N = (n_1,n_2,n_3)$ and $P = (p,q,r)$. Then

$PK \perp AB \Leftrightarrow h_1 := a(p-k_1) = 0,$

$L \in BC \Leftrightarrow h_2 := l_2(b-a) - c(l_1-a) = 0,$

$PL \perp BC \Leftrightarrow h_3 := (p-l_1)(b-a) + c(q-l_2) = 0,$

$M \in CD \Leftrightarrow h_4 := (d-b)(m_2-c) - (e-c)(m_1-b) = 0,$

$h_5 := (e-c)m_3 - (m_2-c)f = 0, h_6 := (m_1-b)f - m_3(d-b) = 0,$

$PM \perp CD \Leftrightarrow h_7 := (p-m_1)(d-b) + (q-m_2)(e-c) + (r-m_3)f = 0,$

$N \in DA \Leftrightarrow h_8 := dn_2 - en_1 = 0,\ h_9 := dn_3 - fn_1 = 0,$

$h_{10} := fn_2 - en_3 = 0,$

$PN \perp DA \Leftrightarrow h_{11} := (p-n_1)d + (q-n_2)e + (r-n_3)f = 0,$

Volume of $KLMN = s \ \Leftrightarrow$

$$h_{12} := \begin{vmatrix} k_1, & 0, & 0, & 1 \\ l_1, & l_2, & 0 & 1 \\ m_1, & m_2, & m_3, & 1 \\ n_1, & n_2, & n_3, & 1 \end{vmatrix} - 6s = 0. \tag{1}$$

Elimination of $k_1, \ldots, n_3$ in the system $h_1 = 0$, $h_2 = 0$, $\ldots$, $h_{12} = 0$ yields the formula[1]

$$F := cfH + sR = 0, \tag{2}$$

where

$H = p^3(c^2d(d-a) - (be^2 + bf^2 - 2cde)(a-b)) - p^2qc(ae(c-e) + f^2(a-2b)) - p^2rcf(ac-2cd-2e(a-b)) + pq^2(c^2(d^2+f^2-ad) + e(2cd-be)(a-b)) + 2pqrf(cd-be)(a-b) - pr^2f^2(ab-b^2-c^2) - q^3ace(c-e) - q^2racf(c-2e) + qr^2acf^2 + p^2(cd(a^2(c-2e) + e(ab+b^2+c^2)) + (e^2+f^2)(ab+b^2+c^2)(a-b) - c(e^2+f^2+d^2)(cd+ae-be)) + pq(cd(d-a)(ab-b^2-c^2-ad+bd) - de(a-b)(b^2+c^2) + a^2ce(c-e) - cf^2(ab+b^2+c^2-a^2) + (a-b)((e^2+f^2)(be-cd) + bd^2e)) - prf((ab-b^2-c^2)(bd+ce-d^2-e^2-f^2) - ac(2be+ac-2cd-2ae)) + q^2ae(c(bd+ce-d^2-e^2-f^2) - (c-e)(ab-b^2-c^2)) + qra(cf(bd+ce-d^2-e^2-f^2) - f(c-2e)(ab-b^2-c^2)) + r^2af^2(ab-b^2-c^2) - pa(cd(c(ad-d^2+ce) - (be+de)(a-b)) + (e^2+f^2)((b^2+c^2-ce)(a-b) - c^2d)) + (qe+rf)a(bd+ce-d^2-e^2-f^2)(ab-b^2-c^2)$

and

$R = 6(d^2+e^2+f^2)((b-d)^2 + (c-e)^2 + f^2)((a-b)^2 + c^2).$

Hence $P \in F$ is the necessary condition for the feet K, L, M, N to be coplanar. Similarly, with the use of program Epsilon [12, 13], we prove that $P \in F$ is the sufficient condition as well [8]. □

We see that $F = 0$ describes a cubic surface.

We can also proceed in another way to find F. Expressing $k_1, \ldots, n_3$ from the system above we get:

$k_1 = p,$

[1] We use software CoCoA which is freely distributed at `http://cocoa.dima.unige.it` and Epsilon library which is freely distributed at `http://www-calfor.lip6.fr/~wang/epsilon/`

$l_1 = (p(a-b)^2 + qc(b-a) + ac^2)/((a-b)^2 + c^2)$,

$l_2 = (pc(b-a) + c^2q + ac(a-b))/((a-b)^2 + c^2)$,

$m_1 = (p(b-d)^2 + q(b-d)(c-e) + rf(d-b) + c(cd - be - de) + b(e^2 + f^2))/((b-d)^2 + (c-e)^2 + f^2)$,

$m_2 = (p(b-d)(c-e) + q(c-e)^2 + fr(e-c) - bcd + cd^2 + b^2e - bde + cf^2)/$

$((b-d)^2 + (c-e)^2 + f^2)$,

$m_3 = (pf(d-b) + qf(e-c) + f^2r + f(b^2 + c^2 - bd - ce))/$

$((b-d)^2 + (c-e)^2 + f^2)$,

$n_1 = (d^2p + deq + dfr)/(d^2 + e^2 + f^2)$,

$n_2 = (dep + e^2q + efr)/(d^2 + e^2 + f^2)$,

$n_3 = (dfp + efq + f^2r)/(d^2 + e^2 + f^2)$.

Substitution for $k_1, l_1, l_2, \ldots, n_3$ into (1) gives F in the form (2), where

$H = c(dp + eq + fr)(p(d-b) + (e-c)q + fr - (d-b)d - (e-c)e - f^2)(cp + q(a-b) - ac) + (p(b-a) + cq + a(a-b))((-p(e^2 + f^2) + qde + rdf)(p(d-b) + q(e-c) + rf + b^2 + c^2 - bd - ce) + (dp + eq + fr)(p((c-e)^2 + f^2) - q(b-d)(c-e) - rf(d-b) - c(cd - be - de) - b(e^2 + f^2)))$,

i.e., $H = K_1K_2K_3 + K_4(K_5K_6 + K_1K_7)$, where $K_1, K_2, K_3, K_4, K_5, K_6$ and K_7 are linear factors in p, q, r.

Later we will express F even in the more concise form.

2 Properties of the locus

In the following suppose that $s = 0$ in (2). Then $F = H$, since $cf \neq 0$. In this section some properties of the cubic H which is associated with a skew quadrilateral $ABCD$ are investigated.

Particularly the following properties of H are studied:

a) decomposability,
b) structure of lines on the cubic,
c) singular points.

2.1 Decomposability

The next theorem is on decomposability of the locus H:

Theorem 2. *The cubic surface which is associate with a skew quadrilateral $ABCD$ is decomposable iff two pairs of sides — either adjacent or opposite*

— of $ABCD$ are of equal lengths p, q.
If $p \neq q$ the cubic decomposes into a plane and a one–sheet hyperboloid,
if $p = q$, i.e., if $ABCD$ is equilateral, the cubic decomposes into three mutually orthogonal planes.

In the following example a cubic H is associated with the quadrilateral $ABCD$ whose two pairs of opposite sides are of equal lengths.

Example 1. For a skew quadrilateral $ABCD$ with $a = 1$, $b = 0$, $c = 1$, $d = 0$, $e = 1$, $f = 1$ we get a cubic surface

$$(pq - q^2 - pr - qr + q + r)(p + r - 1) = 0,$$

which decomposes into a plane and hyperboloid, Fig. 1.

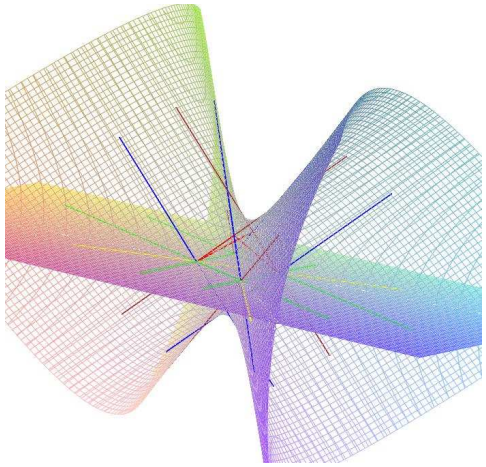


Fig. 1. A cubic $(pq - q^2 - pr - qr + q + r)(p + r - 1) = 0$ decomposes into a plane and hyperboloid

2.2 Structure of lines on the cubic

The well-known Salmon–Cayley theorem [1], [11] states that a smooth cubic surface over algebraic closed field contains exactly 27 lines. In the following the number of real lines which lie on the cubic H are investigated.

Planes $A_1, A_2, A_3, A_4, A_5, A_6, A_7$ and A_8 which are perpendicular to the sides of $ABCD$ and pass through its vertices are crucial for investigation of the structure

of lines on the cubic:

$$\begin{aligned} &A_1:\ A_1 \perp DA,\ D \in A_1, && A_5:\ A_5 \perp BC,\ B \in A_5,\\ &A_2:\ A_2 \perp DA,\ A \in A_2, && A_6:\ A_6 \perp BC,\ C \in A_6,\\ &A_3:\ A_3 \perp CD,\ C \in A_3, && A_7:\ A_7 \perp AB,\ A \in A_7,\\ &A_4:\ A_4 \perp CD,\ D \in A_4, && A_8:\ A_8 \perp AB,\ B \in A_8. \end{aligned}$$

We will see later that they belong to the system of tritangent planes which intersect the cubic H in three lines.

We can easily verify that it holds

$$H = A_1A_3A_5A_7 - A_2A_4A_6A_8, \tag{3}$$

or

$H = (dp + eq + fr - d^2 - e^2 - f^2)((d-b)p + (e-c)q + fr - (d-b)b - (e-c)c)((b-a)p + cq - (b-a)a)p - (dp + eq + fr)((d-b)p + (e-c)q + fr - (d-b)d - (e-c)e - f^2)((b-a)p + cq - (b-a)b - c^2)(p-a)$.

Note that this is one of the most concise forms of H.

The importance of (3) appears by searching for lines lying on the cubic. Namely from $H = 0$ and (3) we get that the line $A_i \cap A_j$, $i = 1,3,5,7$, $j = 2,4,6,8$ belongs to H.
From (3) we obtain the following 12 lines which belong to the cubic surface:

$$\begin{aligned} &a = A_2 \cap A_7,\ b = A_8 \cap A_5,\ c = A_6 \cap A_3,\ d = A_4 \cap A_1,\\ &e = A_2 \cap A_5,\ f = A_8 \cap A_3,\ g = A_6 \cap A_1,\ h = A_4 \cap A_7,\\ &i = A_7 \cap A_6,\ j = A_2 \cap A_3,\ k = A_8 \cap A_1,\ l = A_5 \cap A_4. \end{aligned}$$

Another 6 tritangent planes given by pairs of parallel lines:

$$\begin{aligned} &A_9 = a \cup k, \quad A_{10} = b \cup i, \quad A_{11} = c \cup l,\\ &A_{12} = d \cup j, \quad A_{13} = e \cup g, \quad A_{14} = f \cup h. \end{aligned}$$

Denote:

$$\begin{aligned} C_1 &= ab - bd - ce,\\ C_2 &= b^2 + c^2 - ab - bd - ce,\\ C_3 &= d^2 + e^2 + f^2 - a^2 + ab - bd - ce. \end{aligned} \tag{4}$$

It holds:

Proposition 1.

$$\begin{aligned} &a)\ \textit{The lines } m,n \textit{ coincide} \Leftrightarrow C_1 = 0 \ \textit{ and } \ C_2 \neq 0, C_3 \neq 0.\\ &b)\ \textit{The lines } m,o \textit{ coincide} \Leftrightarrow C_2 = 0 \ \textit{ and } \ C_1 \neq 0, C_3 \neq 0.\\ &c)\ \textit{The lines } n,o \textit{ coincide} \Leftrightarrow C_3 = 0 \ \textit{ and } \ C_1 \neq 0, C_2 \neq 0. \end{aligned} \tag{5}$$

If $C_1 \neq 0, C_2 \neq 0, C_3 \neq 0$ then we obtain another three lines m, n, o

$$m = A_{10} \cap A_{12}, \quad n = A_9 \cap A_{11}, \quad o = A_{13} \cap A_{14}$$

which belong to the cubic H.

In the Fig. 2 the cubic with 15 real lines is depicted.

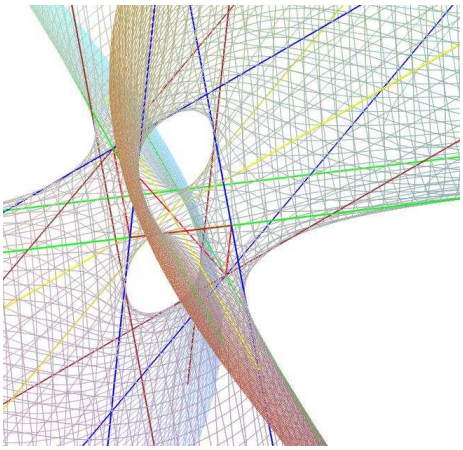


Fig. 2. The cubic $p^2q + pq^2 - p^2r - q^2r + pr^2 + qr^2 - 2pq - r^2 + r = 0$ contains exactly 15 real lines

Finally we add the plane

$$A_{15} = m \cup n \cup o.$$

Note that A_{15} passes through the center S of the circumsphere of $ABCD$.

The planes above yield the following 10 canonical forms[2] of the cubic H:

$$\begin{aligned}
H &= A_2A_4A_{10} + A_5A_7A_{12}, & H &= A_1A_3A_{10} + A_6A_8A_{12},\\
H &= A_4A_8A_{13} + A_1A_5A_{14}, & H &= A_3A_7A_{13} + A_2A_6A_{14}\\
H &= A_1A_7A_{11} + A_4A_6A_9, & H &= A_2A_8A_{11} + A_3A_5A_9,\\
H &= A_1A_2A_{15} + A_9A_{12}A_{13}, & H &= A_3A_4A_{15} + A_{11}A_{12}A_{14},\\
H &= A_5A_6A_{15} + A_{10}A_{11}A_{13}, & H &= A_7A_8A_{15} + A_9A_{10}A_{14}.
\end{aligned} \tag{6}$$

[2] The cubic H is expressed in a canonical form if $H = abc + def$, where a, b, c, d, e, f are linear factors.

2.3 27 lines on the cubic

So far we have investigated cubics H which contain 15 lines. Is there a case when a cubic H contains 27 real lines? The answer gives the following theorem:

Theorem 3. *Let $C_1 \neq 0$, $C_2 \neq 0$, $C_3 \neq 0$. Then a cubic H contains exactly 27 distinct real lines iff*

$$(C_1C_2 - C_2C_3 + C_3C_1)^2 - 4a^2C_1C_2C_3 \ > \ 0 \ . \tag{7}$$

Example 2. For a skew quadrilateral $a = 1$, $b = -2$, $c = 1$, $d = 2$, $e = -1$, $f = 1$ we get the cubic

$2p^3 - 3p^2q - 3pq^2 + 2q^3 - 3p^2r - 3q^2r + 7pr^2 + qr^2 + 24p^2 + 24pq - 3q^2 - 74pr + 10qr - 7r^2 - 26p - 77q + 77r = 0.$

It holds $C_1 = 3, C_2 = 12, C_3 = 8$ and

$$(C_1C_2 - C_2C_3 + C_3C_1)^2 - 4a^2C_1C_2C_3 = 144 \ > \ 0 \ .$$

Then by the Theorem 3 there exist 27 real lines on the cubic, Fig. 3. The com-

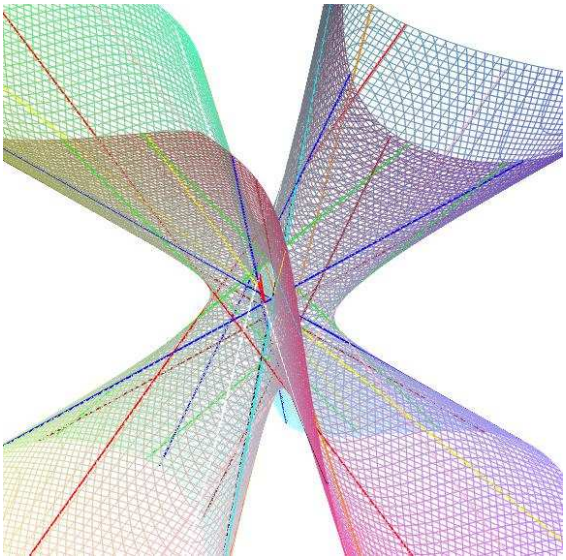


Fig. 3. The cubic $2p^3 - 3p^2q - 3pq^2 + 2q^3 - 3p^2r - 3q^2r + 7pr^2 + qr^2 + 24p^2 + 24pq - 3q^2 - 74pr + 10qr - 7r^2 - 26p - 77q + 77r = 0$ contains 27 real lines

putation of all 27 lines was done in Maple.

2.4 Singular points

The following theorem holds:

Theorem 4. *If* $C_1 = 0$, $C_2 \neq 0$, $C_3 \neq 0$ *or* $C_2 = 0$, $C_1 \neq 0$, $C_3 \neq 0$ *or* $C_3 = 0$, $C_1 \neq 0$, $C_2 \neq 0$ *then* H *possesses 2 nodes.*

Proof. First assume that $C_1 = 0$, $C_2 \neq 0$, $C_3 \neq 0$. By (1) the lines m and n coincide. As $m = A_{10} \cap A_{12}$ and $n = A_9 \cap A_{11}$ then the planes A_9, A_{10}, A_{11} and A_{12} have a common line. Further since $A_9 = a \cup k$, $A_{10} = b \cup i$, $A_{11} = c \cup l$ and $A_{12} = d \cup j$, then the lines a, k, b, i, c, l, d, j intersect the common line $m = n$. It is easy to verify that the lines a, c, i, j, m meet at

$$S_1 = \Big[0, \frac{b^2 + c^2 - ab}{c}, \frac{e(ab - b^2 - c^2)}{cf}\Big],$$

and the lines b, d, k, l, m at

$$S_2 = \Big[a, 0, \frac{d^2 + e^2 + f^2 - ad}{f}\Big].$$

Since neither a, c, i, j, m nor b, d, k, l, m are coplanar, the points S_1 and S_2 are conical singular points — nodes.

Similarly we proceed if $C_2 = 0$ or $C_3 = 0$. □

Remark 1. Note that the singular points S_1 and S_2 are intersections of opposite normals a, c and b, d, and also of the lines i, j and k, l. This is guaranteed by the condition $C_1 = 0$ which means that the diagonals AC and BD are orthogonal.

Example 3. For $a = 1$, $b = 0$, $c = 1$, $d = 0$, $e = 0$, $f = 2$ we get the cubic

$$2p^2q - 2pq^2 + p^2r + q^2r - 2pr^2 - 2qr^2 - 2p^2 + 3pr + 3qr + 2r^2 + 2p - 4r = 0.$$

Since $C_1 = 0$, $C_2 = 1$, $C_3 = 3$ then the cubic has two nodes at the points $(0, 1, 0)$ and $(1, 0, 2)$, Fig. 4.

Remark 2. How to find singular points of H in general? Solving the system $\{H = 0, \frac{\partial H}{\partial p} = 0, \frac{\partial H}{\partial q} = 0, \frac{\partial H}{\partial r} = 0\}$ together with the condition $C_i = 0$ for $i = 1, 2, 3$ does not give any result at the moment.

3 Concluding remarks

In the text some properties of the cubic surface which is associated with a skew quadrilateral in $\mathbb{E}^3$ were investigated. A few remarks and questions:

Some properties of H are still explored (e.g. Eckhards points, the set of all singular points, etc.).

We used 6 parameters a, b, c, d, e, f to describe a cubic H. Is it possible to decrease the number of parameters, for instance by C_1, C_2, C_3 and "an unknown

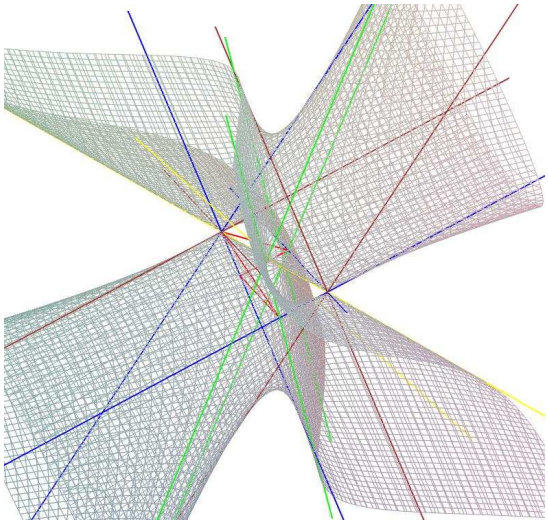


Fig. 4. The cubic $2p^2q-2pq^2+p^2r+q^2r-2pr^2-2qr^2-2p^2+3pr+3qr+2r^2+2p-4r=0$ with two nodes.

expression"?

So far we explored the case when orthogonal projections of a locus point onto the sides of a skew quadrilateral were coplanar (the case $s=0$ in (2)). Similarly we can explore the case when orthogonal projections of a locus point onto the sides of a skew quadrilateral form a tetrahedron of a constant non-zero volume s. What are the properties of the locus (the case $s\neq 0$ in (2))?

What is the locus if we take 4 an *arbitrary* lines in 3D real Euclidean space instead of 4 lines which form a skew quadrilateral?

The cubic H could serve as a model for demonstration of some types of cubic surfaces.

References

1. Dolgachev, I. V.: Classical Algebraic Geometry: A Modern View. Cambridge Univ. Press, Cambridge (2012)
2. Giering, O.: Affine and Projective Generalization of Wallace Lines. J. Geometry and Graphics 1, 119–133 (1997)
3. Guzmán, M.: An Extension of the Wallace–Simson Theorem: Projecting in Arbitrary Directions. Amer. Math. Monthly 106, 574–580 (1999)
4. Holzer, S., Labs, O.: Illustrating the Classification of Real Cubic Surfaces. In: Elkadi, Mourrain, M. B., Piene R. (eds.) Algebraic Geometry and Geometric Modeling, pp. 119–134 (2006)

5. Johnson, R.: Advanced Euclidean Geometry. Dover, New York (1960)
6. Pech, P.: On Simson–Wallace Theorem and Its Generalizations. J. Geometry and Graphics 9, 141–153 (2005)
7. Pech, P.: On a 3D Extension of the Simson–Wallace Theorem. J. Geometry and Graphics 18, 205–215 (2014)
8. Pech, P.: Extension of Simson–Wallace Theorem on Skew Quadrilaterals and Further Properties. LNAI, vol. 9201, pp. 108–118. Springer, Heidelberg (2015)
9. Riesinger, R.: On Wallace Loci from the Projective Point of View. J. Geometry and Graphics 8, 201–213 (2004)
10. Roanes–Lozano, E. M., Roanes–Lozano, M.: Automatic Determination of Geometric Loci. 3D-Extension of Simson–Steiner Theorem. LNAI, vol. 1930, pp. 157–173. Springer, Heidelberg (2000)
11. Schläfli, L.: On the Distribution of Surfaces of the Third Order into Species, in Reference to the Presence or Absence of Singular Points, and the Reality of their Lines. Philos. Trans. Royal Soc. 153, 193–241 (1863)
12. Wang, D.: Elimination Methods. Springer, Wien New York (2001)
13. Wang, D.: Elimination Practice. Software Tools and Applications. Imperial College Press, London (2004)

Automated Generation of Keywords from Images for Geometric Information Search

Dan Song and Xiaoyu Chen*

SKLSDE–LMIB–School of Mathematics and Systems Science,
Beihang University, Beijing 100191, China

Abstract. We outline an approach for automated generation of keywords in a domain of interest from an image to characterize the domain information and knowledge that the image may imply. We take geometry of Euclidean plane as our domain of study and show how keywords with geometric meanings may be generated and be used to search for geometric information on the web with image query. The approach is based on our previous work on automated retrieval of geometric information from images of diagrams. It works by first retrieving basic geometric entities and relations from the query image by means of pattern recognition and numerical verification respectively and then generating advanced geometric information introduced along the hierarchical definitions of derived geometric concepts. Finally, the keywords of different levels are weighted for their influences on the characterization of the image and are combined to generate several groups of keywords ordered according to their weights. Experiments with a preliminary implementation of the approach illustrate the feasibility of automated generation of keywords for searching the web for geometric information, in particular geometric theorems, which query images may imply.

Keywords: geometric feature, pattern matching, image search, geometric knowledge management

1 Introduction

To search for information or knowledge in a domain of interest from web resources via popular search engines such as Google and Bing, one usually needs to characterize queries in a group of keywords in the domain [5]. However, the choice of keywords may heavily influence the quality of searching results. For example, it is not easy to efficiently find appropriate keywords to search for geometric theorems on the web. Images are widely used to represent information in a domain. An interesting question is how to search for domain information on the web with image query. Answers to this question may lead to more friendly and robust interface for general search engines.

* Corresponding author.

A lot of work has been dedicated to domain information retrieval with image query. For example, to retrieve mathematical expressions, an expression-level TF-IDF (term frequency-inverse document frequency) approach was proposed using keyword search, where queries and indexed expressions are represented by keywords taken from LaTeX strings in [7]. For general image retrieval, an efficient approach was proposed in [4] by combining color information and keyword information. Content-based image retrieval approaches have been studied to search for images and perform automatic annotation [3].

In this paper, we will take geometry of Euclidean plane as our domain of study and show how keywords with geometric meanings may be generated and be used to search for deep geometric information on the web by taking images of diagrams as queries. The approach is based on our previous work on automated retrieval of basic geometric information from images of diagrams [2, 6], and then generate advanced geometric information introduced along the hierarchical definitions of derived geometric concepts. Finally, the keywords of different levels are weighted for their influences on the characterization of the image and are combined to generate several groups of keywords ordered according to their weights. In this way, one is able to take an image of diagram as query and obtain geometric information relevant to the query on the web.

The rest of the paper is organized as follows. In Section 2, a general approach is outlined for automated generation of keywords in a domain of interest from an image. The domain $\mathfrak{E}$ of geometry of Euclidean plane is briefly specified in Section 3. Then particular strategies and rules developed for deriving advanced geometric entities and relations from the retrieved basic information are presented in Section 4 followed by methods of generating groups of keywords from the derived geometric information in Section 5. Discussions on implementation issues with experimental results are given in Section 6. The paper is concluded with some remarks in Section 7.

2 Generation of Keywords from Images: A General Approach

In this section, we outline a general approach for automated generation of keywords in a domain of interest from an image to characterize the domain information and knowledge that the image may imply. The approach may lead to methods of searching for domain information on the web with image query. It consists of four main steps as shown in Fig. 1.

1. *Specifying the domain of interest.* Identify a set $\mathbb{E}$ of domain entities, a set $\mathbb{Q}$ of quantities, and a set $\mathbb{R}$ of domain relations and represent the entities, quantities, and their relations formally. Let an image $\mathtt{I}$ in the domain involving the elements of $\mathbb{E}$, $\mathbb{Q}$, and $\mathbb{R}$ be given.
2. *Retrieving basic domain information.* First use pattern recognition and machine learning techniques to detect domain objects from $\mathtt{I}$ and then use numerical computation to mine domain relations among the detected objects to obtain basic domain information that the image may imply.

3. *Deriving advanced domain information.* Along the hierarchical definitions of derived domain concepts, advanced domain information may be introduced from the retrieved basic domain information.
4. *Generating appropriate keywords.* From the derived domain information, keywords of different levels are weighted for their influences on the characterization of the image and are combined to generate groups of keywords ordered according to their weights.

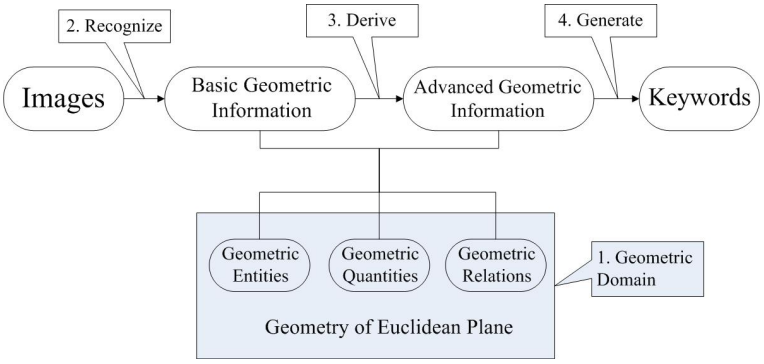


Fig. 1: Overview of a general approach for generating keywords from images

The general approach may be applied to any concrete domain by implementing the tasks proposed in each step. In the following sections, we take geometry of Euclidean plane as the domain of our study and show how keywords with geometric meanings may be generated and be used to search for deep geometric information on the web with image query.

3 Specification of Plane Euclidean Geometric Information

To retrieve geometrically meaningful information from image data, it is necessary to identify in which domain the data should be interpreted and to specify the domain with formal representation of geometric concepts, so that geometric information becomes meaningful and processable.

3.1 Representation of Basic Geometric information

Let the sets $\mathbb{E}_0$, $\mathbb{Q}_0$, and $\mathbb{R}_0$ of concepts of basic geometric entities, basic geometric quantities, and basic geometric relations, respectively, in $\mathfrak{E}$ be

determined first. For example:

$\mathbb{E}_0 = \{$`point`(a::Number, b::Number), `line`(A::Point, B::Point), `halfline`(A::Point, B::Point), `segment`(A::Point, B::Point), `circle`(A::Point, B::Point, C::Point), `circle`(A::Point, B::Point), `circle`(A::Point, r::Distance)$\}$;
$\mathbb{Q}_0 = \{$`distance`(A::Point, B::Point), `angle`(A::Point, B::Point, C::Point), `size`(α::Angle)$\}$;
$\mathbb{R}_0 = \{$`incident`(A::Point, l::Line), `pointOnC`(A::Point, o::Circle), `parallel`(l::Line, m::Line), `perpendicular`(l::Line, m::Line), `equal`(a::Distance, b::Distance), `equal`(a::Size, b::Size)$\}$.

Instances of the concepts in $\mathbb{E}_0$, $\mathbb{Q}_0$, and $\mathbb{R}_0$ are given with types, representations, explanations, and keywords as follows.[1] Note that the keyword(s) of each instance is/are ordered in a list with respect to the frequency of each keyword appearing in geometric documents on the web, such as articles in Wikipedia and other web pages. The higher the frequency of one keyword is, the top the keyword is arranged.

Basic geometric entities

1. **Type**: Point
   Representation: `point`(x, y)
   Meaning: a point with coordinates (x, y)
   Keywords: {`point`}
2. **Type**: Line
   Representation: `line`(A, B) or `halfline`(A, B) or `segment`(A, B)
   Meaning: a straight line or a halfline or a segment passing through two different points A and B
   Keywords: {`line`, `halfline`, `segment`}
3. **Type**: Circle
   Representation: `circle`(A, r) or `circle`(A, B) or `circle`(A, B, C)
   Meaning: a circle with center A and radius r or a circle with center A and passing through another point B or a circle passing through three different points A, B, C
   Keywords: {`circle`}

Basic geometric quantities

1. **Type**: Distance
   Representation: `distance`(A, B)
   Meaning: the distance between A and B where A and B are two points
   Keywords: {`distance`}
2. **Type**: Angle
   Representation: `angle`(A, B, C)
   Meaning: $\angle ABC$ where A, B, and C are three different points
   Keywords: {`angle`}

[1] Given an instance I, we use I.`type`, I.`representation`, I.`meaning`, and I.`keywords` to represent the type, representation, meaning, and keywords of I, respectively.

3. **Type**: Size
   Representation: $\texttt{size}(\alpha)$
   Meaning: the size of α where α is an angle
   Keywords: {`size`}

Basic geometric relations

1. **Type**: Boolean
   Representation: $\texttt{incident}(A, l)$
   Meaning: a point A lies on a line l
   Keywords: {`collinear, incident`}
2. **Type**: Boolean
   Representation: $\texttt{pointOnC}(A, o)$
   Meaning: a point A is on a circle o
   Keywords: {`incident`}
3. **Type**: Boolean
   Representation: $\texttt{parallel}(l_1, l_2)$
   Meaning: a line l_1 is parallel to a line l_2
   Keywords: {`parallel`}
4. **Type**: Boolean
   Representation: $\texttt{perpendicular}(l_1, l_2)$
   Meaning: a line l_1 is perpendicular to a line l_2
   Keywords: {`perpendicular`}
5. **Type**: Boolean
   Representation: $\texttt{equal}(\texttt{distance}(A, B), \texttt{distance}(C, D))$
   Meaning: the distance between two points A and B is equal to the distance between two points C and D
   Keywords: {`equidistant`}
6. **Type**: Boolean
   Representation: $\texttt{equal}(\texttt{size}(\texttt{angle}(A, B, C)), \texttt{size}(\texttt{angle}(D, E, F)))$
   Meaning: the size of $\angle ABC$ is equal to the size of $\angle DEF$
   Keywords: {`equal angle`}

The sets of basic geometric entities, quantities, and relations in $\mathfrak{E}$ given above are not meant to be complete and they can be diminished or enlarged as necessary. For instance, the set of basic geometric quantities can be enlarged by adding the quantity Area.

3.2 Representation of Advanced Geometric information

Based on the concepts in $\mathbb{E}_0, \mathbb{Q}_0$, and $\mathbb{R}_0$, advanced geometric concepts may be derived with formal definitions. For example, the following advanced concepts can be defined and represented in the formal Geometry Description Language (GDL [1]).

Advanced geometric entities

1. Advanced geometric entities with type Point

(a) **Name**: Midpoint
Representation: $M :=$ `midpoint`(A, B)
Definition: `midpoint`(A::Point, B::Point) $\triangleq$ [M::Point where `incident`(M, `line`$(A, B)) \wedge$ `equal(distance`(M, A), `distance`(M, B))]
Keywords: {`midpoint, bisect`}

(b) **Name**: Intersection
Representation: $P :=$ `intersection`(l_1, l_2)
Definition: `intersection`(l_1::Line, l_2::Line) $\triangleq$ [P::Point where `incident`(P, l_1) $\wedge$ `incident`(P, l_2)]
Keywords: {`intersection, intersect`}

(c) **Name**: Foot
Representation: $A :=$ `foot`(l_1, l_2)
Definition: `foot`(l_1::Line, l_2::Line) $\triangleq$ [A::Point where $A :=$ `intersection`$(l_1, l_2) \wedge$ `perpendicular`(l_1, l_2)]
Keywords: {`closest, foot`}

2. Advanced geometric entities with type Line

(a) **Name**: Triangle
Representation: $t :=$ `triangle`(A, B, C)
Definition: `triangle`(A::Point, B::Point, C::Point) $\triangleq$ [{`segment`(A, B), `segment`(B, C), `segment`(C, A)}, where $\neg$`incident`$(A,$ `segment`$(B, C))$]
Keywords: {`triangle`}

(b) **Name**: Quadrilateral
Representation: $q :=$ `quadrilateral`(A, B, C, D)
Definition: `quadrilateral`(A::Point, B::Point, C::Point, D::Point) $\triangleq$ [{`segment`(A, B), `segment`(B, C), `segment`(C, D), `segment`(D, A)}, where $\neg$`incident`$(A,$ `segment`$(B, C)) \wedge \neg$`incident`$(B,$ `segment`$(C, D)) \wedge \neg$`incident`$(C,$ `segment`$(D, A)) \wedge \neg$`incident`$(B,$ `segment`$(D, A))$]
Keywords: {`quadrilateral`}

(c) **Name**: Parallelogram
Representation: $p :=$ `parallelogram`(A, B, C, D)
Definition: `parallelogram`(A::Point, B::Point, C::Point, D::Point) $\triangleq$ [`quadrilateral`(A::Point, B::Point, C::Point, D::Point, where `parallel`(`segment`(A, B), `segment`$(C, D)) \wedge$ `parallel`(`segment`(B, C), `segment`$(A, D))$]
Keywords: {`parallelogram`}

(d) **Name**: Isoscelestrapezoid
Representation: $i :=$ `trapezoid`(A, B, C, D)
Definition: `trapezoid`(A::Point, B::Point, C::Point, D::Point) $\triangleq$ [`quadrilateral`(A::Point, B::Point, C::Point, D::Point), where `parallel`(`segment`(A, B), `segment`$(C, D)) \wedge$ $\neg$`parallel`(`segment`(B, C), `segment`$(A, D)) \wedge$ `equal(distance`(A, B), `distance`$(C, D))$]
Keywords: {`isosceles trapezoid`}

(e) **Name**: n-Polygon ($n \geq 5$)
Representation: $t :=$ `npolygon`$(P_1, \ldots, P_n)$
Definition: `npolygon`$(P_1$::Point$, \ldots, P_n$::Point$) \triangleq$ [{`segment`(P_i, P_{i+1}), $i = 1, \ldots, n-1$, `segment`(P_n, P_1)}, where $\neg$`incident`$(P_i,$ `segment`$(P_j, P_k))$, $i, j, k \in \{1, \ldots, n\}$]
Keywords: {`polygon`}

3. Advanced geometric entities with type Circle

(a) **Name**: Circumcircle
Representation: $c :=$ `circumcircle`(t)
Definition: `circumcircle`$(t$::Triangle$) \triangleq$ [`circle`(A, B, C) where $\{A, B, C\} :=$ `vertex`(t)]
Keywords: {`circumcircle`}

(b) **Name**: Incircle
Representation: $c :=$ `incircle`(t)
Definition: `incircle`$(t$::Triangle$)$
$\triangleq$ [$c :=$ `circle`$(O$::Point$, r$::Length$)$ where $t :=$ `triangle`(A, B, C), `tangent`(`segment`$(A, B), c) \wedge$ `tangent`(`segment`$(A, C), c)$ $\wedge$`tangent`(`segment`$(B, C), c)$]
Keywords: {`incircle`}

Advanced geometric relations

1. **Name**: Tangent
Representation: `tangent`(l, c)
Definition: `tangent`$(l$::Line$, c$::Circle$) \triangleq$ [`pointOnC`(`foot`(`center`$(c), l), c)$]
Keywords: {`tangent`}
2. **Name**: Bisector
Representation: `bisect`(A, B, C, D)
Definition: `bisect`$(A$::Point$, B$::Point$, C$::Point$, D$::Point$) \triangleq$
[`equal`(`size`(`angle`$(A, B, D)),$ `size`(`angle`$(C, B, D)))$]
Keywords: {`angle bisector`}
3. **Name**: Trisector
Representation: `trisect`(A, B, C, D)
Definition: `trisect`$(A$::Point$, B$::Point$, C$::Point$, D$::Point$, E$::Point$) \triangleq$
[`equal`(`size`(`angle`$(A, B, D)),$ `size`(`angle`$(D, B, E)))\wedge$
`equal`(`size`(`angle`$(D, B, E)),$ `size`(`angle`$(E, B, C)))$])
Keywords: {`trisector, trisect`}
4. **Name**: Concurrent
Representation: `concurrent`(l_1, l_2, l_3)
Definition: `concurrent`$(l_1$::Line$, l_2$::Line$, l_3$::Line$) \triangleq$
[`incident`(`intersection`$(l_1, l_2), l_3)\vee$
`incident`(`intersection`$(l_2, l_3), l_1)\vee$
`incident`(`intersection`$(l_1, l_3), l_2)$]
Keywords: {`concurrent`}

4 Derivation of Advanced Geometric Information

By using the approaches described in [2, 6], a set $\mathbb{F}$ of basic geometric entities and relations can be retrieved from a given image of diagram. In this section, we explain how to derive advanced geometric entities and relations from the obtained basic information.

4.1 Strategies for deriving geometric information

From a set of basic geometric entities and relations, different advanced entities and relations can be derived according to their definitions. For instance, from basic relations $\texttt{incident}(P, l)$, $\texttt{incident}(P, \texttt{line}(C, D))$, and $\texttt{equal}(\texttt{distance}(C, P), \texttt{distance}(D, P))$, two advanced entities $P := \texttt{intersection}(l, \texttt{line}(C, D))$ and $P := \texttt{midpoint}(C, D)$ can be obtained. In order to help efficiently derive necessary advanced geometric information inquiry, the following notations and strategies are introduced.

For each geometric instance O, we introduce a property *level* (denoted as $O.\texttt{level}$), which is an integer in $[0, +\infty)$, to characterize the priority of O in the derivation process.

1. If O is a basic geometric entity or a basic geometric relation, then $O.\texttt{level} = 0$;
2. if O is an advanced geometric entity $\mathcal{P}(\overline{\mathcal{O}}_1, \ldots, \overline{\mathcal{O}}_n)$ with definition in the form that $\mathcal{P}(\mathcal{O}_1, \ldots, \mathcal{O}_n) \triangleq [\mathcal{O}_0 \text{ where } S(\mathcal{R}_1, \ldots, \mathcal{R}_m)]$, then $O.\texttt{level} = \max_{i=1}^{n} \overline{\mathcal{O}}_i.\texttt{level} + 1$;
3. if O is an advanced geometric relation $\mathcal{R}(\overline{\mathcal{O}}_1, \ldots, \overline{\mathcal{O}}_n)$ with definition in the form that $\mathcal{R}(\mathcal{O}_1, \ldots, \mathcal{O}_n) \triangleq [S(\mathcal{R}_1, \ldots, \mathcal{R}_m)]$, then $O.\texttt{level} = \max_{i=1}^{m} \mathcal{R}_i.\texttt{level} + 1$.

With the property *level* of each geometric instance, advanced geometric entities and relations can be derived according to the following strategies:

1. derive advanced geometric entities and relations in order from lower level to higher level;
2. for each geometric instance with the same level, first derive advanced geometric entities, and then derive advanced geometric relations;
3. when deriving advanced geometric entities, first generate derived entities with type Point, then generate derived entities with type Line, and finally generate derived entities with type Circle;
4. given geometric relations $\overline{\mathcal{R}}_1, \ldots, \overline{\mathcal{R}}_m \in \mathbb{F}$, create a new label P, derive an advanced geometric entity $P := \mathcal{P}(\overline{\mathcal{O}}_1, \ldots, \overline{\mathcal{O}}_n)$ by definition in the form that $\mathcal{P}(\mathcal{O}_1, \ldots, \mathcal{O}_n) \triangleq [\mathcal{O}_0 \text{ where } S(\mathcal{R}_1, \ldots, \mathcal{R}_m)]$, and add $P := \mathcal{P}(\overline{\mathcal{O}}_1, \ldots, \overline{\mathcal{O}}_n)$ into $\mathbb{F}$; then remove $\overline{\mathcal{R}}_1, \ldots, \overline{\mathcal{R}}_m$ from $\mathbb{F}$;
5. given geometric relations $\overline{\mathcal{R}}_1, \ldots, \overline{\mathcal{R}}_m \in \mathbb{F}$, derive an advanced geometric relation $\mathcal{R}(\overline{\mathcal{O}}_1, \ldots, \overline{\mathcal{O}}_n)$ by definition in the form that $\mathcal{R}(\mathcal{O}_1, \ldots, \mathcal{O}_n) \triangleq [S(\mathcal{R}_1, \ldots, \mathcal{R}_m)]$, add add $\mathcal{R}(\overline{\mathcal{O}}_1, \ldots, \overline{\mathcal{O}}_n)$ into $\mathbb{F}$; then remove $\overline{\mathcal{R}}_1, \ldots, \overline{\mathcal{R}}_m$ from $\mathbb{F}$.

Remark 1. The reasons of removing relations $\overline{\mathcal{R}}_1, \ldots, \overline{\mathcal{R}}_m$ in the strategies 4 and 5 are as follows: (1) prevent $\mathbb{F}$ from explosion of geometric information; (2) geometric information implied by $\overline{\mathcal{R}}_1, \ldots, \overline{\mathcal{R}}_m$ has been already contained in an instance of derived entity or relation, therefore $\overline{\mathcal{R}}_1, \ldots, \overline{\mathcal{R}}_m$ are redundant for characterizing the given image of diagram and for the generation of keywords. The feasibility of these strategies can be demonstrated by our experiments presented in Section 6. However, other strategies may be adopted according to specific requirements.

4.2 Rules for generating advanced geometric information

In this subsection, we present procedural rules of deriving advanced geometric entities and relations.[2]

Derive geometric entities with type Point

- **Intersection**
  1. For each point $P \in \mathbb{F}$ with type Point, create a set $\mathbb{G}_P = \{\texttt{incident}(P, l_i) | \texttt{incident}(P, l_i) \in \mathbb{F}\}$;[3]
  2. For each set $\mathbb{G}_P$, if the cardinal number $\|\mathbb{G}_P\| < 2$, then break; if $\mathbb{G}_P = \{\texttt{incident}(P, l_1), \texttt{incident}(P, l_2)\}$,
     - if P.`level` > 0, then create an advanced geometric entity $P :=$ $\texttt{intersection}(l_1, l_2)$ and add $P := \texttt{intersection}(l_1, l_2)$ into $\mathbb{F}$;
     - otherwise, update $P := \texttt{intersection}(l_1, l_2)$.

     otherwise, $\|\mathbb{G}_P\| > 2$, for each l_i, count the number $\texttt{Count}(l_i)$ of lines in $\mathbb{F}$ with the same endpoints of l_i, and then
     - if P.`level` > 0, create an advanced geometric entity $P :=$ $\texttt{intersection}(l_t, l_r)$ and add $P := \texttt{intersection}(l_t, l_r)$ into $\mathbb{F}$;
     - otherwise, update $P := \texttt{intersection}(l_t, l_r)$,

     where $\texttt{Count}(l_t)$ and $\texttt{Count}(l_r)$ are the maximal two numbers among $\{\texttt{Count}(l_i) | \texttt{incident}(P, l_i) \in \mathbb{G}_P\}$.

Derive geometric entities with type Line

1. Re-represent lines in $\mathbb{F}$.
   (a) For each entity $l \in \mathbb{F}$ with type Line, create a set $\mathbb{P}_l$ collecting all the points $P_1, P_2, \ldots, P_N$ $(2 \leq N)$ incident to l. Then for each point P_i in $\mathbb{P}_l$, we introduce two weights to measure the importance of P_i: (1) P_i.`cw` denotes a weight for P_i which counts the number of circles in $\mathbb{F}$ that P_i is incident to ; (2) P_i.`lw` denotes a weight for P_i which counts the number of lines in $\mathbb{F}$ with P_i as an endpoint. With the two weights, an order $\prec$ may be induced on $\mathbb{P}_l$ in the following rules. Given any two points $P_i, P_j \in \mathbb{P}_l$,

[2] Since different kinds of advanced geometric entities and relations can be derived in a similar way, we will illustrate the rules for deriving some typical advanced geometric entities and relations.

[3] If $i = 0$, then $\mathbb{G}_P = \varnothing$.

i. if (P_i.cw = 1 and P_i.lw = 0) or (P_j.cw = 1 and P_j.lw = 0) or (P_i.cw = P_j.cw), then if P_i.cw + P_i.lw ≤ P_j.cw + P_j.lw, then $P_i \prec P_j$; else $P_j \prec P_i$;
ii. else if P_i.cw < P_j.cw, then $P_i \prec P_j$;
iii. else $P_j \prec P_i$.

(b) All the points in $\mathbb{P}_l$ can be ordered with respect to $\prec$. Suppose $P_1 \prec P_2 \prec \cdots \prec P_N$, then re-represent l into $l :=$ Segment(P_{N-1}, P_N).

2. If there exists a nonempty set $\mathbb{S}$ of lines in $\mathbb{F}$ such that all the endpoints of the lines in $\mathbb{S}$ are incident to a circle, then retrieve a set of n-polygons ($n \geq 3$) $\mathbb{S}$ by Algorithm 1 and add all the obtained n-polygons into $\mathbb{F}$;
3. Retrieve a set of n-polygons ($n \geq 3$) from $\mathbb{F} - \mathbb{S}$ by Algorithm 1 and add all the obtained n-polygons into $\mathbb{F}$.

Derived geometric entities with type Circle

- **Circumcircle**
  Among all the derived n-polygons ($n \geq 3$), if there exist m polygons $p_1, p_2, \ldots, p_m$ which are inscribed to a circle c, then
  1. if there exists p_i ($1 \leq i \leq m$) such that the number of edges of p_i is greater than that of any p_j ($1 \leq j \leq m$), then replace the representation of c with $c :=$ circumcircle(p_i);
  2. otherwise, select p_j ($1 \leq j \leq m$) such that there exists no other p_i the number of edges of which is greater than that of p_j ($1 \leq j \leq m$), and replace the representation of c with $c :=$ circumcircle(p_j).

5 Generation of Keywords for Web and Image Search

In this subsection, we outline our approach of generating keywords from retrieved geometric information for a given image of diagram. Given a set $\mathbb{F}$ of geometric information and an positive integer N, the following procedure outputs a set $\mathcal{K}$, whose element is a set of keywords with cardinal number N.

1. Select a geometric entity O_l with type Line from $\mathbb{F}$ such that O_l.level ≥ O.level for all $O \in \mathbb{F}$ where O is with type Line. Then $\mathcal{K}' := \{\emptyset\}$. For each $\mathrm{K}_i \in \mathcal{K}$, $\mathbf{K}_i := \{\mathrm{K}_i \cup \{k\} \mid k \in \{O_l.\texttt{keywords}\}\}$, $\mathcal{K}' := \mathcal{K}' \cup \mathbf{K}_i$. Then $\mathcal{K} := \mathcal{K}'$ and $\mathbb{F} := \mathbb{F} - \{O_l\}$.
2. Select a geometric entity O_c with type Circle from $\mathbb{F}$ such that O_c.level ≥ O.level for all $O \in \mathbb{F}$ where O is with type Circle. Then $\mathcal{K}' := \emptyset$. For each $\mathrm{K}_i \in \mathcal{K}$, $\mathbf{K}_i := \{\mathrm{K}_i \cup \{k\} \mid k \in \{O_c.\texttt{keywords}\}\}$, $\mathcal{K}' := \mathcal{K}' \cup \mathbf{K}_i$. Then $\mathcal{K} := \mathcal{K}'$ and $\mathbb{F} := \mathbb{F} - \{O_c\}$.
3. Select a geometric entity O_p with type Point from $\mathbb{F}$ such that O_p.level > 0 and O_p.level ≥ O.level for all $O \in \mathbb{F}$ where O is with type Point. Then $\mathcal{K}' := \emptyset$. For each $\mathrm{K}_i \in \mathcal{K}$, $\mathbf{K}_i := \{\mathrm{K}_i \cup \{k\} \mid k \in \{O_p.\texttt{keywords}\}\}$, $\mathcal{K}' := \mathcal{K}' \cup \mathbf{K}_i$. Then $\mathcal{K} := \mathcal{K}'$ and $\mathbb{F} := \mathbb{F} - \{O_p\}$.

Algorithm 1 `detectPolygon`(S_g)

Require: A set S_g of segments.
Ensure: P, a set of set of segments, each $P \in \mathrm{P}$ is a set of segments, which implies a polygon $\mathcal{P}$ by just containing all the edges of $\mathcal{P}$.

```
1: P := ∅;
2: Collect all the endpoints of segments in S_g to V, and set n := ‖V‖, visited[i] :=
   False, i = 1, ..., n, curV := 0, E := ∅ denotes the used edges from original start
   vertex to current one;
3: visited[curV] := True, ncount := 0, m := ‖S_g‖;
4: for i = 1, ..., m do
5:    if curV is an endpoint of S_g[i] and all the segments in E lie on the
          same side of S_g[i] then
6:       idx := the index of the other endpoint of S_g[i] in ‖V‖ except V[curV];
7:       if visited[idx] = True then
8:          Construct the set C ⊂ E which denote a loop from V[idx] to
              V[idx], P := P ∪ {C};
9:          S_g := S_g ∪ E − {S_g[i]};
10:         go to line 3;
11:      else
12:         E := E ∪ {S_g[i]}, S_g := S_g − {S_g[i]}, curV := idx;
13:         go to line 3;
14:      end if
15:   else ncount := ncount + 1;
16:   end if
17: end for
18: if ncount = m then
19:    n_E := E, visited[curV] := False, curV := the other endpoint of
         E[n_E] except V[curV], E := E − {E[n_E]};
20:    go to line 3;
21: end if
22: Combine polygons in P if they share some common edges;
23: return P;
```

4. Select a geometric relation R with type Boolean from $\mathbb{F}$ such that $R.\texttt{level} \geq R'.\texttt{level}$ for all $R' \in \mathbb{F}$ where R' is with type Boolean. Then $\mathcal{K}' := \emptyset$. For each $\mathrm{K}_i \in \mathcal{K}$, $\mathbf{K}_i := \{\mathrm{K}_i \cup \{k\} \mid k \in \{R.\texttt{keywords}\}\}$, $\mathcal{K}' := \mathcal{K}' \cup \mathbf{K}_i$. Then $\mathcal{K} := \mathcal{K}'$ and $\mathbb{F} := \mathbb{F} - \{R\}$.
5. In each of the above four steps, if for any $\mathrm{K} \in \mathcal{K}$, $\|\mathrm{K}\| \geq N$, then the procedure stops; otherwise, repeat the four steps.

6 Implementation and Experiments

The techniques of keyword generation from an image of diagram described above have been implemented in C++ development environment. Diagram images used for our experiments were produced by using GeoGebra.

We have made some preliminary experiments with a number of selected diagram images, for which keyword generation was carried out on a PC with

2.83 GHz CPU and 4.00 GB of memory. In what follows, we give a complete example to illustrate our approach of generating keywords from images of diagrams for searching geometric information on the web. Selected experimental results are listed in Appendix (see Table 1, where the form "A/B" in the "Keywords" column means either A or B can be selected as the keyword for web search).

Example 1. Given an image of diagram of Pascal Theorem (or Nine-point Circle Theorem) as shown in Fig. 2.

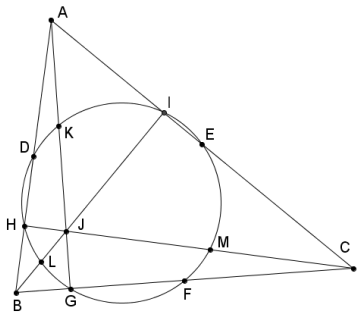


Fig. 2: An image of diagram of Nine-Point Theorem

1. The following basic geometric information can be retrieved from Fig. 2.
   - The set $\mathbb{P}$ of points of interest:

     $G := (108, 412)$, $C := (484, 384)$, $B := (36, 417)$, $J := (103, 336)$,
     $A := (82, 58)$, $H := (48, 329)$, $I := (231, 179)$, $N := (176, 299)$,
     $F := (261, 401)$, $K := (93, 194)$, $M := (294, 360)$, $E := (285, 223)$,
     $L := (68, 378)$, $D := (61, 231)$.

   - The set $\mathbb{L}$ of lines:

     $a := \texttt{segment}(B, C)$, $b := \texttt{segment}(A, G)$, $c := \texttt{segment}(H, C)$,
     $d := \texttt{segment}(A, C)$, $e := \texttt{segment}(B, I)$, $f := \texttt{segment}(B, A)$.

   - The set $\mathbb{C}$ of circles: $g := \texttt{circle}(N, 133)$.

– The set $\mathbb{R}$ of geometric relations:

$\texttt{incident}(G, a)$, $\texttt{incident}(J, b)$, $\texttt{incident}(J, c)$, $\texttt{incident}(J, e)$,
$\texttt{incident}(H, f)$, $\texttt{incident}(I, d)$, $\texttt{incident}(F, a)$, $\texttt{incident}(K, b)$,
$\texttt{incident}(M, c)$, $\texttt{incident}(E, d)$, $\texttt{incident}(L, e)$, $\texttt{incident}(D, f)$,
$\texttt{pointOnC}(G, g)$, $\texttt{pointOnC}(H, g)$, $\texttt{pointOnC}(I, g)$,
$\texttt{pointOnC}(F, g)$, $\texttt{pointOnC}(K, g)$, $\texttt{pointOnC}(M, g)$,
$\texttt{pointOnC}(E, g)$, $\texttt{pointOnC}(L, g)$, $\texttt{pointOnC}(D, g)$,
$\texttt{perpendicular}(a, b)$, $\texttt{perpendicular}(c, f)$, $\texttt{perpendicular}(d, e)$,
$\texttt{equal}(\texttt{distance}(C, F), \texttt{distance}(B, F))$,
$\texttt{equal}(\texttt{distance}(J, K), \texttt{distance}(A, K))$,
$\texttt{equal}(\texttt{distance}(C, M), \texttt{distance}(J, M))$,
$\texttt{equal}(\texttt{distance}(C, E), \texttt{distance}(A, E))$,
$\texttt{equal}(\texttt{distance}(B, L), \texttt{distance}(J, L))$,
$\texttt{equal}(\texttt{distance}(B, D), \texttt{distance}(A, D))$.

2. Advanced geometric information can be derived by using strategies and rules presented in Section 4.
   – The set $\mathbb{E}$ of advanced geometric entities:

$C := (484, 384), B := (36, 417), A := (82, 58), N := (176, 299),$
$F := \texttt{midpoint}(C, B), K := \texttt{midpoint}(J, A), M := \texttt{midpoint}(C, J),$
$E := \texttt{midpoint}(C, A), L := \texttt{midpoint}(B, J), D := \texttt{midpoint}(B, A),$
$J := \texttt{intersection}(b, c), H := \texttt{foot}(c, \texttt{segment}(B, A)),$
$I := \texttt{foot}(\texttt{segment}(A, C), e), G := \texttt{foot}(\texttt{segment}(B, C), b),$
$b := \texttt{halfline}(A, G), c := \texttt{segment}(H, C), e := \texttt{segment}(B, I),$
$g := \texttt{circle}(N, 133), h := \texttt{triangle}(A, B, C).$

   – The set $\mathbb{R}$ of advanced geometric relations:

$\texttt{concurrent}(b, c, e)$,
$\texttt{pointOnC}(G, g)$, $\texttt{pointOnC}(H, g)$, $\texttt{pointOnC}(I, g)$,
$\texttt{pointOnC}(F, g)$, $\texttt{pointOnC}(K, g)$, $\texttt{pointOnC}(M, g)$,
$\texttt{pointOnC}(E, g)$, $\texttt{pointOnC}(L, g)$, $\texttt{pointOnC}(D, g)$.

3. When set N to be 5, the following four groups of keywords can be generated by using the procedure shown in Section 5.
   (a) {triangle, circle, closest, concurrent, midpoint};
   (b) {triangle, circle, foot, concurrent, midpoint};
   (c) {triangle, circle, closest, concurrent, bisect};
   (d) {triangle, circle, foot, concurrent, bisect};

4. By using the generated keywords, geometric information can be searched from web via Google, as shown in Figs. 3 and 4. From the results, we found that the second (or the fourth) group of keywords lead to more relevant information than the first (or the third) group of keywords. The keyword `foot` has more influences on the results than `midpoint`.

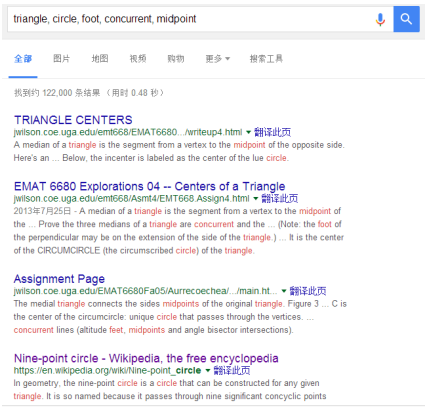


Fig. 3: Searching for web pages via Google by using the generated keywords

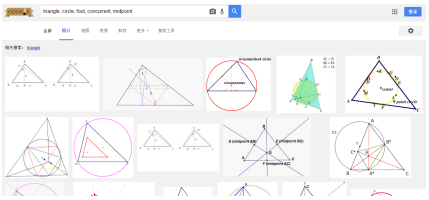


Fig. 4: Searching for images via Google by using the generated keywords

7 Concluding Remarks

We have presented an approach for automated generation of keywords in geometry of Euclidean plane from an image to characterize geometric information and knowledge that the image may imply. This approach can be used to search the web for geometric information, in particular geometric theorems, which a query image may imply. The approach may be also generalized to the generation of keywords in any other domain of interest from an image in the domain.

Acknowledgments. The authors wish to thank Professor Dongming Wang for his insightful ideas on the problem and the referees for their helpful comments on improving the paper. This work has been supported by the Foundation of the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2016ZX-18 and the Fundamental Research Funds for the Central Universities under Grant No. YWF-16-SXXY-01.

References

1. X. Chen. Representation and automated transformation of geometric statements. Journal of Systems Science & Complexity 27(2):382–412, 2014
2. X. Chen, D. Song, and D. Wang. Automated generation of geometric theorems from images of diagrams. Geometric Reasoning — Special issue of Annals of Mathematics and Articial Intelligence 74(3-4):333–358, 2015
3. R. Datta, J. Li, and J. Wang. Content-based image retrieval approaches and trends of the new age. Multimedia Information Retrieval, 2005
4. Y. Jiang, Z. Zhou, and M.U. Chowdhury. Image retrieval based on the combination of color and keyword. In: Proceedings of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business and Applications, pp. 450–453. International Society for Computers and Their Applications (ISCA), 2003
5. S. Oyama, T. Kokubo, and T. Ishida. Domain-specific web search with keyword spices. IEEE Transactions on Knowledge and Data Engineering 16(1):17–27, 2004
6. D. Song, D. Wang, and X. Chen. Discovering geometric theorems from scanned and photographed images of diagrams. In: Automated Deduction in Geometry (F. Botana and P. Quaresma, eds.), Lecture Notes in Computer Science 9201, pp. 149–165. Springer, Berlin Heidelberg, 2015
7. R. Zanibbi and B. Yuan. Keyword and image-based retrieval of mathematical expressions. In: Proceedings of SPIE 7874, Document Recognition and Retrieval XVIII, 78740I, 2011

Appendix.

Table 1: Selected experimental results

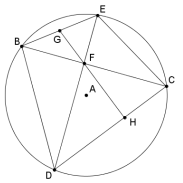
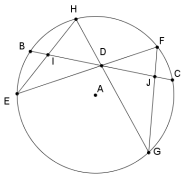
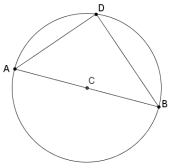
ID	Diagram image	Keywords
1		{quadrilateral, circumcircle, closest/foot, midpoint/bisect}
2		{segment, circle, midpoint/bisect, equidistant}
3		{triangle, circumcircle, midpoint/bisect, perpendicular}

Table 1: Selected experimental results

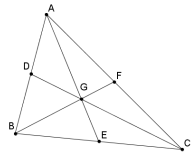
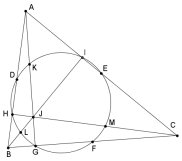
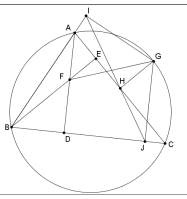
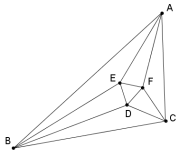
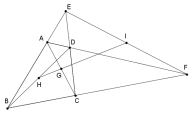
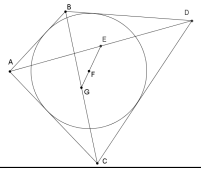
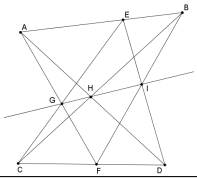
ID	Diagram image	Keywords
4		{triangle, midpoint/bisect, concurrent, intersection/intersect}
5		{triangle, circle, closest/foot, concurrent, midpoint/bisect}
6		{triangle, circumcircle, concurrent, closest/foot, midpoint/bisect}
7		{triangle, trisector, equidistant}
8		{triangle, midpoint/bisect, collinear/incident, intersection/intersect}
9		{quadrilateral, incircle, midpoint/bisect, collinear/incident}
10		{quadrilateral, intersection/intersect, collinear/incident}

Table 1: Selected experimental results

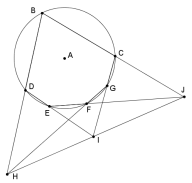
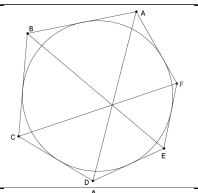
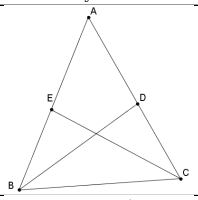
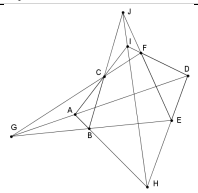
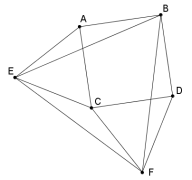
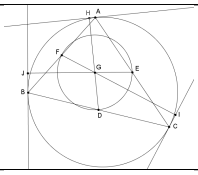
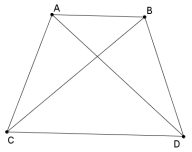
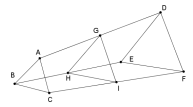
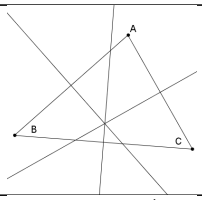
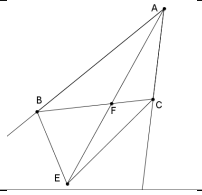
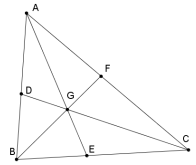
ID	Diagram image	Keywords
101		{polygon, circumcircle, intersection/intersect, concurrent}
11		{polygon, incircle, concurrent}
12		{triangle, intersection, angle bisector, equidistant}
13		{polygon, intersection, collinear/incident, triangle}
14		{polygon, intersection, perpendicular, equidistant}
15		{triangle, circumcircle, midpoint, tangent}
16		{isosceles trapezoid, midpoint/bisect, equidistant, intersection/intersect}

Table 1: Selected experimental results

ID	Diagram image	Keywords
17		{polygon, midpoint/bisect, equidistant, intersection/intersect}
18		{triangle, midpoint/bisect, concurrent, closest/foot}
19		{quadrilateral, intersection/intersect, angle bisector,collinear/incident}
20		{triangle, concurrent,angle bisector}

Formalization of a Surface Subdivision Allowing a Region with Holes without Coordinates

Kazuko Takahashi, Sosuke Moriguchi, and Mizuki Goto*

School of Science&Technology, Kwansei Gakuin University,
2-1, Gakuen, Sanda, 669-1337, JAPAN
ktaka@kwansei.ac.jp, chiguri@acm.org, izconnect705@gmail.com

Abstract. This paper discusses how a surface subdivision is formalized symbolically. We formalize a subdivision of a finite two-dimensional plane allowing a region with holes. We use a data structure called PLCA. A PLCA expression is defined using four constructors and represents a surface subdivision without coordinates. We show construction of a configuration for a subdivision and representation of a subdivision using PLCA. Formalization and proofs are performed within Coq proof assistant.

Keywords: surface subdivision, formalization, PLCA, planarity, Coq

1 Introduction

A number of studies have been undertaken previously on symbolic treatment of geometric or topological properties. Some of these studies used proof assistants to give a computational model representing spatial data, and to certify its formalization with a mechanical proof. They sometimes find drawbacks or oversights in pen-and-paper proofs.

Surface subdivision is a basic concept in computational geometry and topology. Intuitively, it is an embedding of a surface with a finite number of connected regions. There are a number of geometric or topological topics related to surface subdivision, for example, graph embedding, four color theorem and Jordan's simple curve theorem [1]. Surface subdivision determines locations of regions for a given set of points, and it commonly uses a region without holes such as Delaunay triangulation [1]. On the other hand, we consider a case in which a region with holes can be regarded as a subdivision. That is, we admit all configurations shown in Figure 1 as a subdivision of a plane. Configuration here means that a representation of a figure showing its geometric or topological characteristics. In these figures, a red part shows a region with a hole and a green part shows a region as a hole. When we admit such cases, the border of each region is not always a Jordan curve, that is, a simple closed curve without a self-loop [10] (e.g, Figure 1(c)(d)).

* Currently, JFE Advantech Co., Ltd.

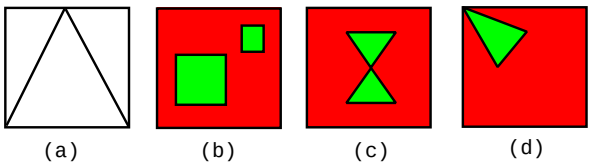


Fig. 1. Surface subdivision: (a) triangulation (no holes), (b) a region with holes, (c) a region with holes which are connected with a point, and (d) a region with a hole which is connected to itself with a point.

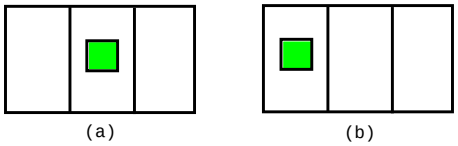


Fig. 2. Different configurations with a disconnected component.

There are few symbolic expressions focusing on topological aspects without the use of coordinates and, to the best of our knowledge, none gives a mechanical proof of the algorithms for handling disconnected components.

Disconnected components can be regarded as regions that are embedded into the holes of another region. Therefore, when we admit disconnected components, we have to explicitly represent the region with holes in which the component is located, since no coordinate information is provided. For example, the two configurations shown in Figure 2 should be considered as being different, which makes formalization difficult.

In this paper, we describe formalization with PLCA to represent a surface subdivision that allows disconnected components. PLCA is a data structure in which all incidence relations between objects are stored [14]. Topological and geometric aspects can be distinguished in this structure. Previously, we have described its inductive construction and proved that the obtained class is planar [15]. We first gave a pen-and-paper proof and then a mechanical proof using a proof assistant Coq [2]. As a rigorous specification is required in a symbolic formalization, it was necessary to solve subtle problems that can be ignored in a pen-and-paper proof [11].

In these studies, "planarity" means not only embedding a PLCA expression in a two-dimensional plane, but also forming a subdivision of a plane. However, we did not refer to subdivision explicitly, nor did we explain in detail the meaning of the preconditions of each constructor; nevertheless, those are significant factors in the symbolic treatment of geometric/topological data. In this paper, we describe construction of PLCA from the viewpoint of subdivision, and how conditions for surface subdivision are represented with PLCA.

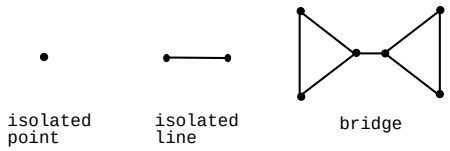


Fig. 3. Disallowed configurations of a subdivision

We consider a surface subdivision as a configuration in which both side of each line always belong to distinct regions. It means that there are no isolated points or isolated lines, and no bridges (Figure 3). In addition, a configuration can include a region with holes as a piece of subdivision and regions connected with a point, which are shown in Figure 1. We represent these conditions in the form of PLCA planarity. We give a more specific expression for a subdivision and show that a planar PLCA forms a subdivision. Our final goal is to implement computational geometry algorithms related to surface subdivision in PLCA, based on the formalization described in this paper.

All the formalization is implemented in Coq. Although we will avoid describing the messy Coq code here, the entire formalization and proofs have been uploaded to [12].

The remainder of this paper is organized as follows. In Section 2, we describe a data structure of PLCA. In Section 3 we show how a surface subdivision is constructed with PLCA in Coq. In Section 4, we discuss how surface subdivision is represented in our formalization. In Section 5 we compare our work with related work, and Section 6 concludes the paper.

2 PLCA

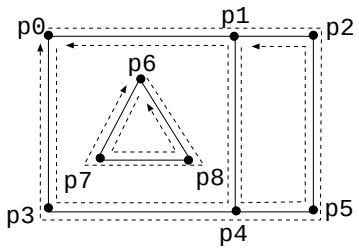
2.1 PLCA Expression

A PLCA data structure was originally designed to give a qualitative representation of a spatial object. The term "qualitative" implies no use of numerical data such as coordinates, sizes or ratios etc.; instead, we perform reasoning focusing on certain characteristics of a spatial object. So far, several qualitative spatial representations have been proposed [13, 4]. Among them, PLCA is designed to distinguish connection patterns of regions [14].

Definition 1 (PLCA expression). *A PLCA expression is defined as a five-tuple,* $\langle P, L, C, A, o\rangle$*, where* P *is a set of points,* $L \subseteq P^2$*,* $C \subseteq L^n$ *(*$n \geq 3$*),* $A \subseteq C^m$ *(*$m \geq 1$*),* $o \in C$*.*

In PLCA, there are four kinds of object: points, lines, circuits, and areas.

– Points are the most primitive objects. Points are distinguishable from each other. We use p as a variable for points.



$P = \{p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$
$L = \{l_0, l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9\}$
$C = \{o, c_0, c_1, c_2, c_3\}$
$A = \{a_0, a_1, a_2\}$

$l_0 = (p_0, p_1)$
$l_1 = (p_1, p_2)$
$l_2 = (p_0, p_3)$
$l_3 = (p_1, p_4)$
$l_4 = (p_2, p_5)$
$l_5 = (p_3, p_4)$
$l_6 = (p_4, p_5)$
$l_7 = (p_6, p_7)$
$l_8 = (p_7, p_8)$
$l_9 = (p_8, p_6)$

$c_0 = [l_2^+, l_5^+, l_3^-, l_0^-]$
$c_1 = [l_3^+, l_6^+, l_4^-, l_1^-]$
$c_2 = [l_9^-, l_8^-, l_7^-]$
$c_3 = [l_7^+, l_8^+, l_9^+]$
$o = [l_0^+, l_1^+, l_4^+, l_6^-, l_5^-, l_2^-]$

$a_0 = \{c_0, c_2\}$
$a_1 = \{c_1\}$
$a_2 = \{c_3\}$

Fig. 4. An example of a PLCA expression.

- A line represents segments between two points. It is defined as a pair of distinct points, like (p_0, p_1)[1]. We use l as a variable for lines. Each line has a direction from the first to the second element of the pair. The direction of a line l is denoted by l^+, and its inverse direction by l^-.
- A circuit represents a closed outline. It is defined as a list of lines, like $[l_0, l_1, \ldots, l_n]$. Each circuit is closed, i.e., the first element of the first element l_0 and the second element of the last element l_n are the same. We use c as a variable for circuits.
- An area represents a region enclosed with circuits. It is defined as a set of circuits, like $\{c_0, c_1, \ldots, c_n\}$. We use a as a variable for areas.

Additionally, we use a specific circuit in the outermost side of the figure denoted by *outermost*. We use o as a variable for *outermost*.

We show an example of a PLCA expression and an instance of its corresponding configuration in a two-dimensional plane in Figure 4. Note that this expression has three areas: area a_1 is an area without a hole, area a_0 is an area with a hole and area a_2 fits in that hole.

[1] The original PLCA allows curved lines. Our definition does not allow (p, p) as a line.

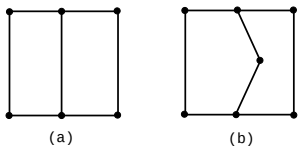


Fig. 5. Same subdivision, different data structures.

2.2 Equivalence on objects

We use a list as a data type to implement P, L, C, A, each line, circuit and area, as our first implementation in Coq. It causes us to define the equivalence relation on lists [2].

Hereafter, the symbol '+' denotes an operation of list contcatenation. For example, when $V = [v, w]$, $[u] + V$ returns $[u, v, w]$.

Two symbolic expressions that are not identical may stand for the same configuration. For example, circuits $[l_1, l_2, l_3]$ and $[l_2, l_3, l_1]$ represent the same circuit, and areas $[c_1, c_2, c_3]$ and $[c_3, c_2, c_1]$ represent the same area. This is due to the cyclic structure of a circuit, and the data structure of a list for an area. In this case, we should regard them as equivalent.

On the other hand, we have to distinguish between two expressions when they stand for the same subdivision but have different configurations. For example, the expressions for the two configurations shown in Figure 5 should be distinguished, since the number of points and lines are different.

Considering the above points, we define an equivalence relation over PLCA expressions.

As P, L, C and A are defined as lists, they are equivalent to permutations of themselves. Each element of L is equivalent to its inverse, and each element of C is equivalent to its rotation.

Definition 2 (PLCA-equivalence). *Two expressions are* PLCA-equivalent *iff the pair of them is in the transitive closure of the following binary relation* $\mathcal{R}_{eq}$*.*

- $\langle P, [l^+] + L, C, A, o\rangle\ \mathcal{R}_{eq}\ \langle P, [l^-] + L, C, A, o\rangle$.
- $\langle P, L, [c] + C, [[c] + a] + A, o\rangle\ \mathcal{R}_{eq}\ \langle P, L, [c'] + C, [[c'] + a] + A, o\rangle$*, where* c' *is a rotated circuit of* c*.*
- $\langle P, L, C, A, o\rangle\ \mathcal{R}_{eq}\ \langle P', L', C', A', o\rangle$*, where* P'*,* L'*,* C' *and* A' *are permutations of* P*,* L*,* C *and* A*, respectively.*
- $\langle P, L, C, [a] + A, o\rangle\ \mathcal{R}_{eq}\ \langle P, L, C, [a'] + A, o\rangle$*, where* a' *is a permutation of* a*.*
- $\langle P, L, [o] + C, A, o\rangle\ \mathcal{R}_{eq}\ \langle P, L, [o'] + C, A, o'\rangle$*, where* o' *is a rotated circuit of* o*.*

[2] If we use data type from a Coq library, for example, $MSet$, then we may omit these definitions.

We call the above relation PLCA-equivalence.

Note that $\mathcal{R}_{eq}$ is reflexive because of the third condition of $\mathcal{R}_{eq}$ and the reflexivity of permutations. Therefore, PLCA-equivalence is reflexive and transitive.

2.3 PLCA consistency

A PLCA expression is too permissive to find a corresponding topological space. For example, if there exists more than one area that contain the same circuit, such an expression does not make sense. And we should also avoid duplication in a list. Therefore, we put consistency on this data structure so that it makes sense.

Definition 3 (consistent PLCA). *Let $\langle P, L, C, A, o\rangle$ be a PLCA expression. The PLCA expression is said to be* a consistent PLCA expression *iff it satisfies all of the following conditions.*

- *For each point $p \in P$, there exists $l \in L$ such that $p \in l$.*
- *For each line $l \in L$, all points in l are in P.*
- *For each line $l \in L$, there exist circuits $c, c' \in C$ such that $l^+ \in c$ and $l^- \in c'$*[3]*.*
- *For each circuit $c \in C$, each line in c or its inverse line is in L.*
- *For each circuit $c \in C$ except for o, there exists an area $a \in A$ such that $c \in a$.*
- *For each area $a \in A$, all circuits in a are in C.*
- *For any area $a \in A$, $o \notin a$.*
- *Each point $p \in P$ appears only once in P.*
- *Each line $l^+ \in L$ appears only once in L and $l^- \notin L$.*
- *Each circuit $c \in C$ appears only once in C and any rotated circuits other than c do not appear in C.*
- *Each area $a \in A$ appears only once in A and any equivalent area other than a does not appear in A.*

3 Encoding PLCA

We show PLCA specification in Coq.

3.1 Construction of PLCA

We construct PLCA so that its configuration is a subdivision of a finite two-dimensional plane allowing a region with holes. We use four constructors, and explain each constructor according to the meaning of their preconditions.

[3] In the formalization in Coq, we define this condition as "for each line l such that $l^+ \in L$ or $l^- \in L$, there exists the circuit c such that $l \in c$." These conditions are the same.

Initially, we make a finite area, then apply constructors sequentially. We divide an existing area into two areas and as a result, the number of areas is incremented by exactly one each time a constructor is applied. At this time, we either add a connected component or a disconnected component depending on whether we cut an existing circuit or not. We introduce a new object *path* to divide an area. As PLCA has no coordinate information, we have to specify the area in which such a path is added. As a symbolic operation, we delete one existing area and add two new areas, and also reconfigure all related objects. The difficulty on re-composition of a circuit is that we have to find the beginning of a list, since a circuit has a cyclic structure.

3.2 Objects

Objects in PLCA are defined as follows, where `nat` denotes a natural number and `prod` denotes a product of the two specified arguments.

```
Definition Point := nat.
Definition Line := prod Point Point.
Definition Circuit := list Line.
Definition Area := list Circuit.
```

3.3 Basics

Here we provide some definitions of basic functions. Functions *reverse* and *reverse_linelist* are prepared to make the reverse circuit (lists of lines), i.e., reverse_linelist$([l_1^+, l_2^+, ..., l_n^+]) = [l_n^-, ..., l_2^-, l_1^-]$). Proposition *Rot* is defined on a pair of circuits to check whether one circuit is obtained by rotating the other, i.e., Rot$([l_1^+, l_2^+, ..., l_n^+], [l_2^+, l_3^+, ..., l_n^+, l_1^+])$ is *true*, which is used to define PLCA-equivalence. InL is a proposition that checks whether a line or its inverse is in L.

```
Definition reverse(l : Line) := (snd l, fst l).

Definition reverse_linelist(ll : list Line) := rev (map reverse ll).

Inductive Rot : list A -> list A -> Prop :=
| rotSame : forall (l : list A), Rot l l
| rotNext : forall (l l' : list A) (a : A), Rot l (a :: l')
    -> Rot l (l' ++ [a]).

Definition InL (l: Line)(L: set Line) : Prop :=
  set_In l L \/ set_In (reverse_linelinst l) L.
```

PLCA-equivalence is defined as a proposition on a pair of lists of points, a pair of lists of lines, a pair of lists of circuits, a pair of lists of areas, and a pair of circuits. Each condition of Definition 2 is encoded. For example, the last condition, showing that the rotated outermost is equivalent to the original one,

is encoded as follows: assume that one PLCA expression has `C` as a list of circuits and `o` as *outermost*, and the other PLCA expression has `C'` as a list of circuits and `o'` as *outermost*; if `o'` is a rotated circuit of `o`, and `C'` also contains `o'` instead of `o`, then these PLCA expressions are equivalent.

```
rotateOutermost : forall(P : list Point)(L : list Line)
                        (C : list Circuit)(A : list Area)
                        (o o' : Circuit),
   Rot o o'
   -> PLCAequivalence P L C A o P L (map (replace_circuit o o') C) A o'
```

We define *a path* as a sequence of lines with non-negative length. A path does not have a point that appears more than once. We define another object *trail* that is allowed to have a point that appears more than once but is not allowed to have a line that appears more than once. A path is used as a divider of a circuit to make a new area, while a trail shows a property of a segment of a circuit. Note that a circuit is not always a Jordan curve. We can make a closed path (or trail) by adding a line connecting their start and end points, which corresponds to a circuit.

Definition 4 (path). *A list of lines* $[(p_0, p_1), (p_1, p_2), \ldots, (p_{n-1}, p_n)]$ $(n > 0)$ *is said to be* a path *if* $p_0, \ldots, p_n$ *are distinct.*

Definition 5 (trail). *A list of lines* $[(p_0, p_1), (p_1, p_2), \ldots, (p_{n-1}, p_n)]$ $(n > 0)$ *is said to be* a trail*, if there exists no* i, j $(0 < i \neq j < n)$ *such that* $(p_{i-1}, p_i) = (p_{j-1}, p_j)$ *or* $(p_{i-1}, p_i) = (p_j, p_{j-1})$.

For example, in Figure 6, a path and a trail are depicted by a red line and a green line, respectively. Circuits in Figure 6(a) are both closed paths, while circuits in Figure 6(b)(c) are two closed paths and one closed trail.

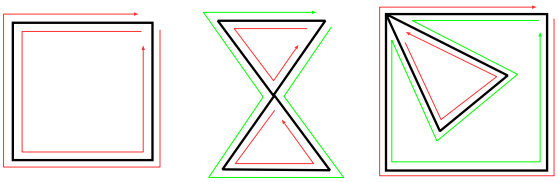


Fig. 6. Circuits constructed by paths (red) and trails (green): (a) two closed paths (b)(c) two closed paths and one closed trail.

An object trail is inductively defined as follows. The arguments show the starting point, the ending point, inner points and inner lines. As a base case,

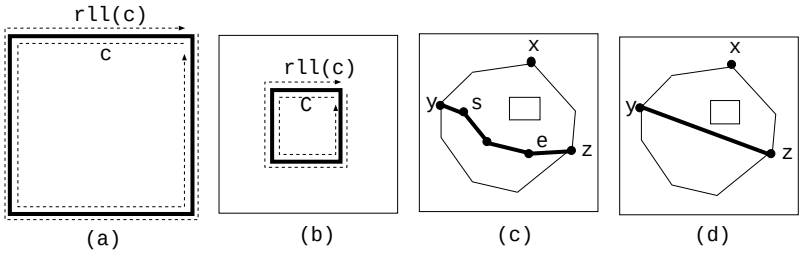


Fig. 7. Constructors: (a) single_loop, (b) add_loop, (c) add_inpath, and (d) add_inline. In these figures, 'rll' denotes *reverse_line_list*.

make a nil_trail that consists of a single point without lines [4]. Make a step_trail by adding a point that is not included in the existing inner points, and a line connecting to the new point and the ending point of the existing path. The function also checks that the new line is not included in the existing inner lines.

```
Inductive trail : Point -> Point -> list Point -> list Line -> Prop :=
| nil_trail  : forall(p : Point), trail p (p::nil) nil
| step_trail : forall(p1 p2 p3 : Point)(pl : list Point)(ll : list Line),
   trail p2 p1 pl ll
-> p3 <> p2
-> ~LIn (p3, p2) ll
-> trail p3 p1 (p3::pl) ((p3, p2)::ll).
```

3.4 Constructors

We show four constructors together with their preconditions: *single_loop*, *add_loop*, *add_inpath* and *add_inline*. A path is added in the inner side of the *outermost*. In the previous work, we consider another constructor *add_outpath* that adds a path on the outer side of the *outermost* [11]. In that case, we allow a trail as an *outermost*. We can construct PLCA that admits such a configuration, but when focusing on subdivision, it is not suitable to consider such a case. Therefore, the definition is little different from the one shown in the previous work.

single_loop: Constructor *single_loop* is an initialization that corresponds to add a Jordan curve to make *outermost*. It makes a simple path and adds a new line that connects its start and end points (Figure 7(a)).

The preconditions are simple. (1) The length of a path is more than two to make directions of its inner lines deterministic. (2) Points in the path should not be existing ones.

[4] A data structure $[(p, p)]$ is taken as a base case in this implementation, which is not allowed in the definition of a trail or path. And when trail or path is used in a constructor in the implementation, condition on the length is added.

add_loop: Constructor *add_loop* adds a Jordan curve to the specified area. It makes a simple path and adds a new line that connects its start and end points (Figure 7(b)).

The preconditions are the same as those of *single_loop*. (1) The length of a path is more than two to make directions of its inner lines deterministic. (2) Points in the path should not be existing ones.

A new configuration is obtained after putting the closed path onto the existing area.

add_inpath: Constructor *add_inpath* is rather complicated. This constructor corresponds to the division of an area by cutting two points of the same existing circuit that belongs to that area (Figure 7(c)).

The preconditions are as follows. (1) The length of a path is more than zero to make directions of its inner lines deterministic. (2) Points in the path should not be existing ones. (3) Two points (that may be the same) connected to the start and end points of the path should be in the same circuit to avoid a bridge.

A new configuration is obtained after making an area by re-composing an existing circuit with a new path. In this case, at least two lines are added to L.

add_inline: Constructor *add_inline* is similar to *add_inpath* but simpler, since no new points are added. Intuitively, it makes a shortcut between the two existing, different points in the same circuit (Figure 7(d)).

The precondition is: (1) There is no line that connects two different points in the same circuit to guarantee planarity.

A new configuration is obtained after making an area by re-composing an existing circuit with a new line. In this case, only one line is added to L.

3.5 Inductive PLCA

The four described constructors are formalized below. A PLCA expression constructed in this way is called *an inductive PLCA expression*. In the following, 'rll' denotes a function *reverse_linelist*.

single_loop: Let $p_1, p_2, ..., p_n$ $(n \geq 3)$ be distinct points, $P = [p_1, p_2, ..., p_n]$, $L = [(p_n, p_1), (p_1, p_2), ..., (p_{n-1}, p_n)]$, $c = [(p_n, p_1), (p_1, p_2), ..., (p_{n-1}, p_n)]$, $C = [c, \mathrm{rll}(c)]$ and $A = [[c]]$. Then, $\langle P, L, C, A, \mathrm{rll}(c)\rangle$ is an inductive PLCA expression.

add_loop: Let $\langle P, L, C, A, o\rangle$ be an inductive PLCA expression. If $p_1, p_2, ..., p_n$ $(n \geq 3)$ are distinct, p_i for any i does not appear in P, $c = [(p_n, p_1), (p_1, p_2), ..., (p_{n-1}, p_n)]$ and $a \in A$, then $\langle P', L', C', A', o\rangle$ is an inductive PLCA expression, where P', L', C', A' are as follows.

- $P' = [p_1, ..., p_n] + P$
- $L' = [(p_n, p_1), (p_1, p_2), ..., (p_{n-1}, p_n)] + L$
- $C' = [c, \mathrm{rll}(c)] + C$
- $A' = [[c]] + [[\mathrm{rll}(c)] + a] + (A - [a])$

add_inpath: Let $\langle P, L, C, A, o\rangle$ be an inductive PLCA expression. Suppose that $s, p_1, ..., p_n, e$ are distinct, s, e and p_i for any i do not appear in P. Let $ap = [s, p_1, ..., p_n, e]$, $al = [(s, p_1), ..., (p_n, e)]$. Note that if $s = e$, then $ap = [s]$ and $al = [\,]$. Also suppose that $c \in a$, $a \in A$, and $c = lxy + lyz + lzx$ where lxy, lyz and lzx start with points x, y and z and end with points y, z and x, respectively. At least one of y and z or s and e are different. $\langle P', L', C', A', o\rangle$ is an inductive PLCA expression, where P', L', C', A' are as follows.

- $P' = ap + P$
- $L' = [(y, s), (e, z)] + al + L$
- $C' = [[(s, y)] + lyz + [(z, e)] + \mathrm{rll}(al), lxy + [(y, s)] + al + [(e, z)] + lzx] + (C - [c])$
- $A' = [[[(s, y)] + lyz + [(z, e)] + \mathrm{rll}(al)], [lxy + [(y, s)] + al + [(e, z)] + lzx] + (a - [c])] + (A - [a])$

add_inline: Let $\langle P, L, C, A, o\rangle$ be an inductive PLCA expression. Suppose that $c \in a$, $a \in A$, and $c = lxy + lyz + lzx$ where lxy, lyz and lzx start with points x, y and z and end with points y, z and x respectively. If either (y, z) or its reverse line is not in L, $\langle P, L', C', A', o\rangle$ is an inductive PLCA expression, where L', C', A' are as follows.

- $L' = [(y, z)] + L$
- $C' = [[(z, y)] + lyz, lxy + [(y, z)] + lzx] + (C - [c])$
- $A' = [[[(z, y)] + lyz], [lxy + [(y, z)] + lzx] + (a - [c])] + (A - [a])$

Figure 8 shows examples of an area division by constructor *add_inpath*. We put a path on the specified area and connect its start and end points to the circuit. Note that y and z are the cutting points of an existing circuit, while x is the starting point of a circuit. Since the circuit has a cyclic structure, and the rotation of a cycle is equivalent to the original one, reconfiguration of a cycle begins with the original starting point. Figure 8(a) is a case to put a path on the area without a hole, Figure 8(b) is a case to put a path on the area with a hole, and Figure 8(c) is a case in which y and z are the same point, which corresponds to adding a loop to the existing point.

We show the corresponding Coq specification of each constructor in the Appendix.

4 Representation of Subdivision

We show that inductive PLCA represents a surface subdivision of a finite two-dimensional plane that allows a region with holes as a subdivision.

The condition for subdivision is represented in a form of PLCA planarity which consist of three properties: PLCA-constraint, PLCA-connectedness and PLCA-euler.

Definition 6 (PLCA-constraints). *Let $\langle P, L, C, A, o\rangle$ be a PLCA expression. It is said that the expression satisfies* PLCA-constraints *iff it satisfies all of the following conditions.*

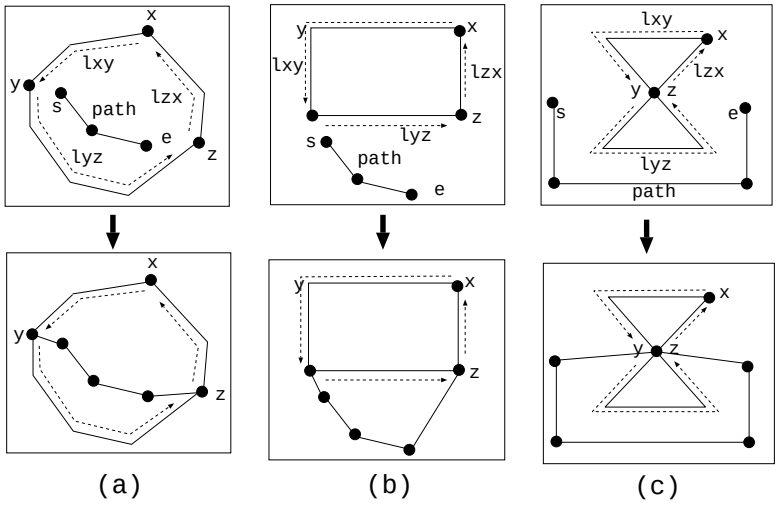


Fig. 8. Examples of dividing an area by *add_inpath*: (a) put a path on the area without a hole, (b) put a path on the area with a hole, (c) two points in the circuit are the same.

1. *For each line* $(p, p') \in L$, $p \neq p'$.
2. *For each circuit* $c \in C$, $|c| \geq 3$ *and* c *is a closed trail.*
3. *For each area* $a \in A$,
   - *if* $c, c' \in a$ *and* $c \neq c'$, *then* c *and* c' *have no shared points or lines.*
   - *if* $c \in a$, *then* c *appears in* a *only once and no rotated circuit of* c *appear in* a.
   - $|a| \geq 1$.
4. o *is a closed path.*

The first condition of PLCA-constraints guarantees that there is no isolated point, and the second condition guarantees that there is no bridge between points. The third condition is for handling duplicated points in a single circuit. For example, in Figure 9, a circuit $[(p_0, p_1), (p_1, p_2), (p_2, p_3), (p_3, p_1), (p_1, p_4), (p_4, p_5), (p_5, p_0)]$, that passes point p_1 more than once, is included only in the green area. Note that it is not a combination of two circuits. The fourth condition is to guarantee that our target plane to be divided is a finite plane encircled with a Jordan curve.

```
Definition PLCAconstraints(L : list Line)(C : list Circuit)
                          (A : list Area)(o : Circuit) : Prop :=
  Lconstraint L /\ Cconstraint C /\ Aconstraint A
               /\ Circuit_constraints o.
```

Each constraint is defined as a proposition. For example, the second condition of Definition 6 is encoded as follows:

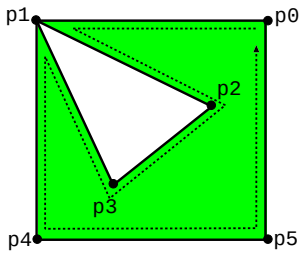


Fig. 9. Treatment of circuit including a duplicated point.

```
Definition Circuit_constraints(c : Circuit) : Prop :=
 (exists x : Point, exists pl : list Point, trail x x pl c)
  /\ 3 <= length c.
Definition Cconstraint(C : list Circuit) : Prop :=
  forall(c : Circuit), In c C -> Circuit_constraints c.
```

Definition 7 (PLCA-connect). *Let $\langle P, L, C, A, o\rangle$ be a PLCA expression. A pair of objects in the expression is said to be* PLCA-connect *iff it is in the symmetric transitive closure of the following relation $\mathcal{R}_c$.*

1. *If $p \in P$, $l \in L$ and $p \in l$, then $p\,\mathcal{R}_c\,l$.*
2. *If $l^+ \in L$ or $l^- \in L$, $c \in C$ and $l^+ \in c$, then $l^+\,\mathcal{R}_c\,c$.*
3. *If $c \in C$, $a \in A$ and $c \in a$, then $c\,\mathcal{R}_c\,a$.*

Definition 8 (PLCA-connectedness). *It is said that a PLCA expression satisfies* PLCA-connectedness *iff any pair of objects in the expression are PLCA-connected.*

Intuitively, PLCA-connectedness guarantees that no objects are separated, including the *outermost*. Each object is traceable from *outermost*.

Representing PLCA-connectedness has one difficulty: this property holds over the different types of P, L, C and A. Therefore, we have to convert these data types to a common data type *object*, and prepare a new incidence function OIn.

```
Inductive object :=
| o_point   : Point   -> object
| o_line    : Line    -> object
| o_circuit : Circuit -> object
| o_area    : Area    -> object.

Definition OIn (o : object) : Prop :=
 match o with
| o_point p   => In p P
| o_line l    => In l L
| o_circuit c => In c C
| o_area a    => In a A
 end.
```

Then, PLCA-connectedness is implemented as follows.

```
Inductive PLCAconnect : object -> object -> Prop :=
| PLcon : forall(p : Point)(l : Line),
    In p P  -> In l L -> InPL p l ->
      PLCAconnect (o_point p) (o_line l)
| LCcon : forall(l : Line)(c : Circuit),
    In l L  -> In c C  -> LIn l c  ->
      PLCAconnect (o_line l) (o_circuit c)
| CAcon : forall(c : Circuit)(a : Area),
    In c C  -> In a A  -> In c a  ->
      PLCAconnect (o_circuit c) (o_area a)
| SYMME : forall(o1 o2 : object),
      PLCAconnect o1 o2 -> PLCAconnect o2 o1
| TRANS : forall(o1 o2 o3 : object),
      PLCAconnect o1 o2 -> PLCAconnect o2 o3 -> PLCAconnect o1 o3.

Definition PLCAconnected : Prop :=
 forall(o1 o2 : object), OIn o1 -> OIn o2 -> PLCAconnect o1 o2.
```

Definition 9 (PLCA-euler). *Let* $\langle P, L, C, A, o\rangle$ *be a PLCA expression. It is said that the expression satisfies* PLCA-euler *iff* $|P| - |L| - |C| + 2|A| = 0$.

PLCA-euler guarantees that a PLCA expression can be embedded in a two-dimensional plane so that the orientation of each circuit can be correctly defined.

Coq implementation is straightforward.

```
Definition PLCAeuler : Prop :=
  length P + 2 * length A = length L + length C.
```

Definition 10 (planar PLCA). *If a consistent PLCA satisfies PLCA-constraints, PLCA-connectedness, and PLCA-euler, then it is said to be* a planar PLCA*.*

The following theorem shows that inductive PLCA satisfies planarity.

Theorem 1. *Inductive PLCA is a planar PLCA.*

This is proved by proving the following three lemmas.

```
Lemma PLCA_consistency_and_constraints :
forall(P : list Point)(L : list Line)(C : list Circuit)(A : list Area)
      (o : Circuit),
 I P L C A o -> PLCAconsistency P L C A o /\ PLCAconstraints L C A o.

Lemma PLCAconnectedness :
forall(P : list Point)(L : list Line)(C : list Circuit)(A : list Area)
  (o : Circuit),
 I P L C A o ->
   (forall(p : Point)(l : Line),
```

```
     In p P -> In l L -> PLCAconnect P L C A (o_point p) (o_line l))
/\ (forall(l : Line)(c : Circuit),
     In l L -> In c C -> PLCAconnect P L C A (o_line l) (o_circuit c))
/\ (forall(c : Circuit)(a : Area),
     In c C -> In a A -> PLCAconnect P L C A (o_circuit c) (o_area a)).

Lemma PLCAeuler :
forall(P : list Point)(L : list Line)(C : list Circuit)(A : list Area)
      (o : Circuit),
 I P L C A o -> PLCAeuler P L C A.
```

The PLCA-equivalence preserves PLCA planarity.

Theorem 2. *Let e, e' be equivalent PLCA expressions. If e is a planar PLCA then e' is also a planar PLCA.*

We define that an expression of subdivision mentioned in Section 1 as follows. It is defined depending on the case whether a line is included in *outermost* or not. Intuitively, this definition means that for each line, if it is not in *outermost*, then distinct two areas are connected by this line; otherwise, there exists only one area that is connected with the outside of the figure by this line.

Definition 11 (subdivision). *Let $e = \langle P, L, C, A, o\rangle$ be a consistent PLCA expression. For each $l \in L$, if the following condition is satisfied, then e is said to be* a subdivision*: (i) there exist areas $a, a' \in A$ such that $a \neq a'$, $l^+ \in c \in a$, and $l^- \in c' \in a'$, for $l \notin$ outermost, and (ii) there exists an area a and a circuit c such that $l \in c \in a$, $c \neq$ outermost, for $l \in$ outermost.*

Finally, we get the following theorem that means a planar PLCA forms a subdivision.

Theorem 3. *A planar PLCA is a subdivision.*

It is proved by the following theorems by case split depending on whether a line is included by the outermost.

```
Theorem PLCAsubdivision_area :
forall(P : list Point)(L : list Line)(C : list Circuit)(A : list Area)
  (o : Circuit),
 PLCA_planar P L C A o ->
 forall (l : Line),
  In l L
  -> ~LIn l o
  -> exists (c c' : Circuit),
   In c C
   /\ In c' C
   /\ In l c
   /\ In (reverse l) c'
   /\ exists (a a' : Area),
       In a A
```

```
       /\ In a' A
       /\ In c a
       /\ In c' a'
       /\ a <> a'.

Theorem PLCAsubdivision_outermost :
forall(P : list Point)(L : list Line)(C : list Circuit)(A : list Area)
  (o : Circuit),
 PLCA_planar P L C A o ->
 forall (l : Line),
  LIn l o
  -> exists (c : Circuit),
   In c C
   /\ LIn l c
   /\ o <> c
   /\ exists (a : Area),
       In a A
       /\ In c a.
```

5 Related Works

Hypermap is a well-known data structure for handling topological aspects of spatial data; it is an algebraic data structure that consists of a set of darts and two permutations. Hypermap uses a dart as a primitive, and an edge and a vertex are defined as compositions of darts. Circuit and connected components are calculated by permutations. Hypermap is a useful data structure but it is hard to envisage, for example, a figure embedded in a two-dimensional plane, since vertices, edges and faces are not represented directly. On the other hand, PLCA provides a more straightforward expression of a figure.

There are several works that give a formalization based on hypermap and a proof of geographic/topological properties using proof assistants. Brun et al. showed a derivation of a program to compute a convex-hull for a given set of points from their specification [3]. They specified the algorithm and proved its correctness using a structural induction. Gonthier showed the four color theorem based on surface subdivision [9]. However, they did not treat a disconnected component. Dufourd discusses that embedding polyhedra onto a plane is a sufficient condition for planarity [6]. In that work, a hypermap was applied to formalize a discrete version of a Jordan curve and to prove its soundness [7]. Dufourd's most relevant work is a formalization of an Delaunay triangulation [8]. They represented Delaunay triangulation in a form where all edges have two darts and all faces have three vertices. A coordinate was used to represent a proper triangle, and disconnected components were not handled. Our formalization on subdivision includes disconnected components.

Besides a hypermap, Yamamoto et al. studied a graph embedded on a plane using HOL [16]. That model is simpler since it does not handle regions. There are few studies on qualitative spatial representation from the viewpoint of theorem proving. Recently, an interesting work is done by Dapoigny et al. [5]. They

formalized Tarski's mereogeometry using Coq. The representation is based on a part-hole relationship between regions, which is completely different from that of PLCA.

The doubly connected edge list is a popular data structure in computational geometry for representing a surface subdivision allowing disconnected components [1]. It contains records for each face, edge, vertex of the subdivision. Each edge is regarded as two half-edges bounding different faces, and each half-edge has a pointer to its connecting half-edges as well as its twin. Comparing its data structure with PLCA, an edge (or a line) is represented in relating to the two faces that share it in both data structures; however, incidence relations of objects are represented as pointers in the record in doubly connected edge list, whereas an object is explicitly defined using other existing objects in PLCA; the location of each object is determined uniquely by the coordinates in doubly connected edge list, which is different from our work. Moreover, the main goal of computational geometry is to develop an algorithm that can reduce computational complexity, whereas here, we investigate the soundness of an algorithm. The method proposed can handle geometric data on a more abstract level, and can therefore be applied to prove the correctness of algorithms including doubly connected edge lists.

6 Conclusion

We have shown a symbolic representation with PLCA for a surface subdivision of a finite two-dimensional plane that allows holes in a region. All the formalization and proofs are performed in Coq and consist of about 12,800 lines [12].

In future work, following issues are considered:

- To specify other intuitive definition of planarity and prove the equivalence with PLCA-Euler.
- To implement high level computational geometry algorithms such as Delaunay triangulation or graph embedding algorithms in PLCA and prove them.
- To extend the proposed approach to handle more general surfaces such as polyhedra or torus with an arbitrary genus.

References

1. de Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O.: *Computational Geometry*, Springer-Verlag (1997).
2. Bertot, Y. and Castéran, P.: *Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions*, Springer Verlag (1998).
3. Brun, C., Dufourd, J. -F. and Magaud, N,: Designing and Proving Correct a Convex Hull Algorithm with Hypermaps in Coq. *Computational Geometry : Theory and Applications*, 45(8):436-457 (2012).
4. Cohn, A. G. and Renz, J.: Qualitative Spatial Reasoning *Handbook of Knowledge Representation, F. Harmelen, V. Lifschitz and B. Porter(eds.)*, Chapt 13, pp.551-596, Elsevier (2007).

5. Dapoigny, R. and Barlatier, P.: A Coq-Based Axiomatization of Tarski's Mereogeometry, *12th Conference on Spatial Information Theory (COSIT2015)*, LNCS, vol. 9368, pp. 108–129, Springer-Verlag (2015).
6. Dufourd, J. -F.: Polyhedra Genus Theorem and Euler Formula: A hypermap-formalized intuitionistic proof, *Theoretical Computer Science*, 403(2-3):133-159, Elsevier (2008).
7. Dufourd, J. -F.: An Intuitionistic Proof of a Discrete Form of the Jordan Curve Theorem Formalized in Coq with Combinatorial Hypermaps. *Journal of Automated Reasoning*, 43(1):19-51 (2009).
8. Dufourd, J. -F. and Bertot, Y.: Formal Study of Plane Delaunay Triangulation. *1st International Conference on Interactive Theorem Proving (ITP2010)*, LNCS, vol. 6172, pp. 211–226, Springer-Verlag (2010).
9. Gonthier, G.: Formal Proof - The Four Color Theorem. *Notices of the AMS*, 55(11):1382-1393 (2008).
10. Kosniowski, C.: *A First Course in Algebraic Topology*, Cambridge University Press (1980).
11. Moriguchi, S., Goto, M., Takahashi, K.: Towards Verified Construction for Planar Class of a Qualitative Spatial Representation. *7th International Symposium on Symbolic Computation in Software Science (SCSS2016)*, EPiC Series in Computing, vol.39, pp.117-129 (2016).
12. Moriguchi, S., Goto, M., Takahashi, K.: Formalization of IPLCA. `http://ist.ksc.kwansei.ac.jp/~ktaka/IPLCA2016Apr`
13. Stock, O. (Ed.): *Spatial and Temporal Reasoning*. Kluwer Academic Publishers (1997).
14. Takahashi, K., Sumitomo, T. and Takeuti, I.: On Embedding a Qualitative Representation in a Two-Dimensional Plane. *Spatial Cognition and Computation*, 8(1-2):4-26 (2008).
15. Takahashi, K., Goto, M. and Miwa, H.: Construction of a Planar PLCA Expression: A Qualitative Treatment of Spatial Data. *Agents and Artificial Intelligence*, LNCS, vol. 9494, pp. 298–315, Springer-Verlag (2015).
16. Yamamoto, M., Nishizaki, S., Hagiya, M. and Toda, Y.: Formalization of Planar Graphs. *Higher Order Logic Theorem Proving and Its Applications*, LNCS, vol.971, pp. 369–384, Springer-Verlag (2005).

Appendix

Coq specification of each constructor.

```
Inductive I : list Point -> list Line -> list Circuit -> list Area
            -> Circuit -> Prop :=
| single_loop : forall(pl : list Point)(ll : list Line)(x y : Point),
   simplepath x y pl ll
-> x <> y
-> 2 <= length ll
-> I pl ((y,x)::ll) (((y,x)::ll)::(reverse_linelist ((y,x)::ll))::nil)
     ((((y,x)::ll)::nil)::nil) ((reverse_linelist ((y,x)::ll)))

| add_inline : forall(P : list Point)(L : list Line)
             (C : list Circuit)(A : list Area)(o : Circuit)
             (a : Area)(x y z : Point)(pxy pyz pzx : list Point)
             (lxy lyz lzx : list Line),
   I P L C A o
-> In a A
-> In (lxy ++ lyz ++ lzx) a
-> trail x y pxy lxy
-> trail y z pyz lyz
-> trail z x pzx lzx
-> ~LIn (y, z) L
-> y <> z
-> 2 <= length lyz
-> 2 <= length (lxy ++ lzx)
-> I P
     ((y,z)::L)
     (((z,y)::lyz)::(lxy++(y,z)::lzx)
                  ::(set_remove eq_listline_dec (lxy ++ lyz ++ lzx) C))
     ((((z,y)::lyz)::nil)
      ::((lxy++(y,z)::lzx)::(set_remove eq_listline_dec
                                          (lxy ++ lyz ++ lzx) a))
      ::(set_remove eq_listcircuit_dec a A))
     o

| add_inpath : forall(P : list Point)(L : list Line)
             (C : list Circuit)(A : list Area)(o : Circuit)
             (a : Area)(x y z s e : Point)(ap pxy pyz pzx : list Point)
             (al lxy lyz lzx : list Line),
   I P L C A o
-> In a A
-> In (lxy ++ lyz ++ lzx) a
-> trail x y pxy lxy
-> trail y z pyz lyz
-> trail z x pzx lzx
-> simplepath s e ap al
-> 1 <= length (al ++ lyz)
-> 1 <= length (lxy ++ lzx)
```

```
-> y <> z \/ s <> e
-> (forall(p : Point), In p ap -> ~In p P)
-> I (ap++P)
     ((y,s)::(e,z)::al++L)
     ((((s,y)::lyz)++((z,e)::(reverse_linelist al)))
      ::(lxy++((y,s)::al)++((e,z)::lzx))
      ::(set_remove eq_listline_dec (lxy ++ lyz ++ lzx) C))
     (((((s,y)::lyz)++((z,e)::(reverse_linelist al)))::nil)
      ::((lxy++((y,s)::al)++((e,z)::lzx))
      ::(set_remove eq_listline_dec (lxy ++ lyz ++ lzx) a))
      ::(set_remove eq_listcircuit_dec a A))
     o

| add_loop : forall(P : list Point)(L : list Line)
             (C : list Circuit)(A : list Area)(o : Circuit)
             (a : Area)(x y : Point)(ap : list Point)(al : list Line),
   I P L C A o
-> In a A
-> simplepath x y ap al
-> x <> y
-> 2 <= length al
-> (forall(p : Point), In p ap -> ~In p P)
-> I (ap++P)
     ((y,x)::al++L)
     (((y,x)::al)::(reverse_linelist ((y,x)::al))::C)
     ((((y,x)::al)::nil)
     ::((reverse_linelist ((y,x)::al))::a)
           ::(set_remove eq_listcircuit_dec a A))
     o.
```

Author Index

ADG 2016

This volume contains the 14 papers and 3 invited talks presented at ADG 2016: Eleventh International Workshop on Automated Deduction in Geometry held on June 26-28, 2016 in Strasbourg, France.

ADG is a forum facilitating the exchange of ideas, presentation of new research results and demonstrations of software tools lying at the intersection of geometry and automated deduction. The selected papers, reviewed by an international Program Committee, cover diverse topics ranging from polynomial algebra, invariant and coordinate-free methods, synthetic and logic approaches, techniques for automated geometric reasoning from discrete mathematics, symbolic and numeric methods for geometric computation, geometric algorithms, geometric constraint solving, experimental studies with automated theorem provers, applications to mechanics, origami and geometric modeling.

The previous ten workshops were held in Coimbra 2014, Edinburgh 2012, Munich 2010, Shanghaï 2008, Pontevedra 2006, Gainesville 2004, Linz 2002, Zurich 2000, Beijing 1998, and Toulouse 1996.