Proof assistant decision procedures for formalizing origami

Cezary Kaliszyk and Tetsuo Ida

{kaliszyk,ida}@cs.tsukuba.ac.jp
Symbolic Computation Research Group
University of Tsukuba

Abstract. Origami constructions have interesting properties that are not covered by standard euclidean geometry. Such properties have been shown with the help of computer algebra systems. Proofs performed with computer algebra systems can be accompanied by proof documents, still they lack complete mathematical rigorousity, like the one provided by proof assistant checked proofs. Transforming such proofs to machine checkable proof scripts poses a number of challenges.
In this paper we describe issues that arise when proving properties of origami constructions using proof assistant decision procedures. We examine the strength of Gröbner Bases implementations comparing proof assistants with each other and with the implementations provided in computer algebra systems. We show ad-hoc decision procedures that can be used to optimize the proofs. We show how maximum equilateral triangle inscribed in a square construction can be formalized. We show how a equation system solving mechanism can be embedded in a CAS decision procedure of a proof assistant.

1 Introduction

1.1 Computational Origami

Origami is the traditional Japanese art of paper folding. In the recent years it is becoming more popular because of its applications in science, and education. It can also be a tool for geometrical constructions; namely it describes representing objects using paper folds. Instead of ruler and compass in Euclidean geometry one can use paper folding as a basis to create new points and lines on a surface. Starting with a square area represented by four corners of an initial origami, one can specify folds to superpose existing points or lines. Such superpositions give rise to new lines and the intersections of the new lines with existing ones create new points on the origami.

The traditional meaning of folds has been expressed in a rigorous way by Huzita and Justin [7, 8, 11]. Their work has classified all the possible folds, creating a system of operations for origami geometry (called *axioms* in the literature). They also show that the system is complete; this means that for the case of folding along one line no other operations that superpose points and lines are possible. The origami constructable points are a superset of the points constructable with Euclidean geometry; which also means that properties of the constructions performed with origami can be more complicated than those performed with ruler and compass.

The term *computational origami* has been used to refer to the branch of computer science that studies various aspects of origami. It includes representation of origami, design of origami algorithms, visualization, and geometrical theorem proving of computational and mathematical properties of origami. Computational origami most often refers to analyzing origami with a computer; while origami refers to an infinite virtual paper. In this paper the most important notion from computational origami used is the correspondence between geometry and algebra that allows to express the origami axiom system logically.

Recently a system for computational origami Eos has been created [15]. This system is capable of visualizing origami constructions based on Huzita's axioms, analysing the origami folds algebraically, and showing properties of the constructions. During the computational origami construction, geometrical constraints are accumulated. The symbolic representation is then transformed into algebraic forms by constraint solving using the generalized equations which describe the axioms. The algebraic representation is a set of polynomials, which describes the geometrical construction in a general way. The polynomials are then used to prove properties of the construction using the Gröbner bases method and CAD (Cylindrical Algebraic Decomposition). The proofs performed by Eos are automated; after specifying the goal, assumptions and a coordinate system, they use Mathematica's symbolic expression transformations and the implementations of the above two proof methods.

A number of constructions have been performed with Eos and certain properties of those constructions have been proved with its help:

- Trisection of an angle
- Maximum equilateral triangle
- Regular Heptagon
- Morley's Triangle
- Crane layers and sides

In this paper we use Eos as a source of computational origami problems; together with its mechanism for gathering the properties of origami constructions.

1.2 Computer Algebra functionality in Proof Assistants

Mainstream Computer Algebra Systems, for instance Mathematica and Maple, are weakly founded. This means that the expressions processed there do not have precisely defined semantics. The way expressions are processed in those systems is supposed to resemble mathematics as done on paper. But the fact that semantics are not well defined can be a reason for errors. There are various reasons for the mistakes found in mainstream CAS systems: assumptions can be lost, types of expressions can be forgotten [2], the system might get confused between branches of 'multi-valued' functions, and of course the algorithms of the system themselves may contain implementation errors [21]. Simple mistakes have been found and fixed over the years; however mistakes made when performing more complicated computations are still found.

```
 In1 := vector [2; 2] - vector [1; 0] + vec 1
Out1 := vector [2; 3]
 In2 := diff (diff (λx. 3 * sin (2 * x) + 7 + exp (exp x)))
Out2 := λx. exp x pow 2 * exp (exp x) + exp x * exp (exp x) +
        -- 12 * sin (2 * x)
 In3 := N (exp (1)) 10
Out3 := #2.7182818284 + ...
 In4 := x + 1 - x / 1 + 7 * (y + x) pow 2
Out4 := 7 * x pow 2 + 14 * x * y + 7 * y pow 2 + 1
 In5 := sum (0,5) (λx. x * x)
Out5 := 30
 In6 := sqrt (x * x) assuming x > 1
Out6 := x
```

Fig. 1. Example session that shows interaction with the prototype computer algebra input-response loop. When the user inputs expressions to be processed in the `In` fields, the system produces output in `Out` lines together with (not displayed) HOL Light theorems that state the equality between the inputs and the outputs. Because of the way numbers are defined in HOL Light, working with them requires coercions. To work with multiple types in HOL Light coercions are needed and the `&` symbol is necessary for some numbers; here prioritization has been assumed and the coercions were skipped. Similarly the rest of a numerical approximation (marked as `...`) is hidden in the output.

We have built a prototype computer algebra like input-response-loop inside HOL Light [13], with the user interface designed close to the interfaces of popular computer algebra systems. In Figure

This architecture also has drawbacks, the two main drawbacks are the complexity of the implementation and the efficiency. Every simplification that one needs to perform has to be a proof producing simplification and every equation needs to be accompanied with a proof. Similarly all the simplifications as seen in Figure

The formulas that show up when proving properties of origami systems can be processed in a commercial computer algebra system; however already there it can take a substantial amount of time. The Eos proof document describing the proof for Morley's theorem by Abe's method [5] shows that the call to Mathematica's Gröbner bases algorithm takes 1668.12 seconds on a modern machine. Taking into account that all operations of the prototype CAS system are much slower than their Mathematica counterparts, and the fact that the implementations of algorithms chosen to be certified are less efficient algorithms it is naturally not possible to obtain the same proof performed automatically in the prototype system.

In this paper we show tactics that allow simplifying the goals that arise in Eos in such a way that they can be fully formally verified with the help of decision procedures. The tactics solve (or simplify) systems of equations. We compare the efficiency of the implementations of computer algebra algorithms in various proof assistants and show how the goals that arise in certain origami constructions can be proved with the help of these tools.

1.3 Contents

The rest of this paper is organized as follows. In Section

2 Formalizing a Proof Document

When an origami construction is performed with the help of Eos and its properties are checked, optionally the system can create a *proof document*. It is an output that includes all the operations performed within an origami construction along with all the steps of the proof. Every proof document starts with a header and a copy of the user given program for constructing the origami. The program always starts with a new origami and includes the performed folds and unfolds and the new points and lines. For every new point and line the way it is obtained is stored.

In the first stage of the proof document the points and lines are represented in a geometrical way. This means that geometric relations between points and lines are given, but no coordinates are assigned to the points.

In the second stage the points and lines are given variable coordinates and with the help of the origami axiom system the geometrical properties (properties that talk about points, lines and folds) are translated into algebraic properties. The operations of origami constructions are expressed in terms of equations (assumed as axioms) and this step is a substitution in these equations. During this stage some precision is lost. The equations that talk about equal distances would be expressed as equality between square roots. To avoid square roots that are not handled by real arithmetic decision procedures, those equations are squared.

There are two kinds of optimizations performed in the second stage. First, same points and lines are assigned the same coordinates. Second, predicates that are not relevant for the goal are removed. These two steps have been introduced since Mathematica's algorithm is not able to eliminate them automatically and otherwise its complexity increases with a bigger number of variables and equations. In a formalization irrelevant assumptions and equal variables can be easily removed (for example in Isabelle the `clarify` tactic removes both).

The last part of the proof document is the crucial one from our point of view. The decision procedures built into Mathematica are used to verify the large formula prepared in the previous steps. The computation performed in this step is done behind the scenes and as such cannot be verified independently. This is the reason for a formalization of this step, which we describe in the following section.

3 Equilateral triangle construction

In this section we will focus on the proof of the maximal equilateral triangle construction performed with origami, and the formalization of the computational part of the equilateral proof. The construction as shown in Figure

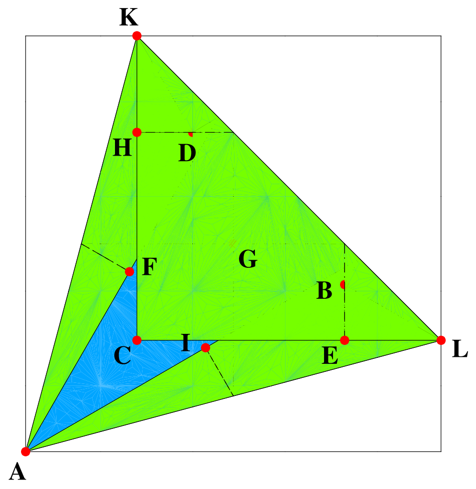

Fig. 2. Equilateral triangle construction as performed by Eos. Only the final origami state is shown. For the folds where more than one resulting line is possible, the system interactively asks the user which of the lines is the intended one.

The computer-algebra decision procedure as presented in [13] is not able to do any simplifications on a formula this big. Still parts of it can be simplifed and this can be made the first part of the proof. We performed this first part in three proof assistants to compare the available mechanisms for reasoning on goals arizing from Eos; we also compare it with Mathematica's simplification. For the experiment we used HOL Light [6], Isabelle [19] and Coq [3].

The Gröbner bases algorithms present in the 3 proof assistants are too weak to solve the goal without any preprocessing. In case of HOL Light and Isabelle the mechanisms do not terminate in reasonable time (we tried one week on a powerful server), in case of Coq after a few minutes the procedure exits with the goal transformed in ring equations but not solved.

To perform the complete proof in HOL Light we first need to simplify the equations. To do this manually we first solve some of the sub-equations using the ring and field decision procedures. The `COMPLEX_FIELD` conversion will produce

$$\begin{aligned}&a_1 = 0 \land (-1+a_1)*(-1+b_1) = 0 \land (-1+b_1)*b_1 = 0 \land (-1+a_2)*(-1+b_2) = 0 \land (-1+b_2)*b_2 = \\ &0 \land (-1+a_3)*(-1+b_3) = 0 \land (-1+b_3)*b_3 = 0 \land (-1+a_4)*(-1+b_4) = 0 \land (-1+b_4)*b_4 = \\ &0 \land (-1+b_4)*(b_1+c_1+2a_1) = 0 \land (b_1+2*c_1)/2 = 0 \land c_1 - b_1 + 2*a_1 + a_4^2*b_1 + a_4^2*c_1 + (-2)*a_1* \\ &b_4 + 2*a_1*a_4^2 + 2*a_4*b_1 + 2*b_1*b_4 = 0 \land (-1+b_3)*(a_2+c_2) = 0 \land c_2 - a_2 + a_2*a_3^2 + a_3^2*c_2 + 2*a_2* \\ &a_3 + 2*a_2*b_3 + 2*b_2*b_3 = 0 \land a_3+b_3+c_3 = 0 \land a_4+b_4+c_4 = 0 \land x_3 = 0 \land x_8 = 0 \land b_1*x_1 - a_1*y_1 = \\ &0 \land c_1 + (a_1*x_1)/2 + (b_1*y_1)/2 = 0 \land b_1*(-1+x_2) - a_1*y_2 = 0 \land a_2*y_2 + b_2*x_1 - a_2*y_1 - b_2*x_2 = \\ &0 \land c_1 + a_1*(1+x_2)/2 + (b_1*y_2)/2 = 0 \land (2*c_2 + a_2*x_1 + a_2*x_2 + b_2*y_1 + b_2*y_2)/2 = \\ &0 \land c_1 + a_1*x_3 + b_1*y_3 = 0 \land c_1 + a_1*x_4 + b_1*y_4 = 0 \land c_2 + a_2*x_4 + b_2*y_4 = 0 \land -1 + y_5 = \\ &0 \land c_2 + a_2*x_5 + b_2*y_5 = 0 \land a_1*(y_5-y_6) + b_1*(x_6-x_5) = 0 \land c_1 + a_1*(x_5+x_6)/2 + b_1(y_5+y_6)/2 = 0 \land y_7 = \\ &0 \land c_3 + a_3*x_7 + b_3*y_7 = 0 \land c_4 + a_4*x_8 + b_4*y_8 = 0 \land -1 + (x_8 - y_8 + x_7(-1+y_8) + (1-x_8)*y_7)*\xi_1 = 0 \Rightarrow \\ &-((1-x_7)^2) + (x_7-x_8)^2 - (1-y_7)^2 + (y_7-y_8)^2 = 0 \land (x_7-x_8)^2 + (y_7-y_8)^2 - (1-x_8)^2 - (1-y_8)^2 = 0\end{aligned}$$

Fig. 3. Algebraic formula to be proved. The equations arising from all the steps of the construction have been gathered by EOS. We show it here, as an example problem to be solved; the equation systems for other origami problems are similar, just contain of more equations with more variables.

the rewrite rules for solving linear equations and `FIX_X_ASSUM` can be used to replace the original equation assumption with the solved one. The HOL Light goal-state is internally a sequence of implications, where the goal is implied by all the assumptions. To rewrite all variable occurrences in the other assumptions, a specific solved assumption is the only one kept in the assumptions list while all other ones are moved to the goal.

Solving linear equations and performing substitutions allows reducing the goal to a system of 5 equations. Moving the assumptions back to the goal and generalizing over the five variables creates two subgoals, which both can be solved by HOL Light's built in real quantifier elimination procedure. The procedure takes 18 seconds to prove the two goals on a modern computer. The HOL Light proof script is 98 lines long.

The same proof can be performed in a similar way in Isabelle; with two important differences. The `clarify` tactic together with the simplifier is able to substitute the value of a computed assumption in other ones and remove unnecessary assumptions automatically. Similarly the `algebra` decision procedure is able to operate on a goal with variables and assumptions introduced in the context as well as with a goal being a conjunction of two equations. This means that the manual proof can be cleaner than the HOL Light one; still the choice of equations to solve and the rules to apply is left to the user. The final call to the Gröbner bases decision procedure takes 1.4 seconds to solve the simplified goal. The Isabelle proof script is 151 lines long.

In our experiments the `nsatz` tactic in Coq was too weak to solve even the goal with five equations. This result was quite surprising, as we expected the procedure that makes use of reflection to be stronger than the ones present in HOL Light or Isabelle. The procedure is able to solve goals involving quadratic equations, but for more complicated goals it transforms the goal into equations mentioning explicitly the real ring and exits.

4 Equations solving tactic

In this section we describe a tactic that solves or simplifies systems of equations motivated by the manual formalization. With the help of this tactic the original set of equations can be simplified to a much simpler formula, namely for the problem presented in the previous section it can be simplified automatically to the one presented in Figure

$$a_4^2 + 4 * a_4 + 1 = 0 \wedge a_3^2 + 4 * a_3 + 1 = 0 \wedge 1 + a_3 - a_3 * x_7 = 0 \wedge 1 - y_8 + a_4 = 0 \wedge -1 - xi_1 * y_8 - \xi_1 * x_7 + \xi_1 * x_7 * y_8 = 0 \Rightarrow -2 + y_8^2 + x_7 * 2 = 0 \wedge -2 + y_8 * 2 + x_7^2 = 0$$

Fig. 4. Simplified, with solved variables removed for clarity

The tactic is similar to extended Gaussian elimination; it allows the original system to contain equations of higher degrees as well as inequalities. The simplifications are performed in an order that takes into account the complexity of symbolic transformations. Given a system of equations the tactic counts the numbers of occurrences of variables in each equation. As the tactic is running a threshold for a maximum number of variables is increasing. For the equations that have less variables than the current threshold, all the occurrences are moved to one side of the equation. Next the left side of the equation can be normalized, and in case if the equation is a linear equation it can be solved.

In the following we will distinguish the assumptions and goal part of the HOL Light goal-state, as described in Section

The tactic optionally is able to remove the variables that have been solved. Since the equations are processed together with the goal, the variables that have been solved have also been replaced in the goal. This means that removing equations involving those variables does not weaken the goal.

Repeating the tactic performs a simplification of the original system of equations. For a system of linear equations the procedure is equivalent to Gaussian elimination optimized to perform fewer operations. It also accepts higher order equations and inequalities, performing normalization of higher degree equations.

4.1 Evaluation

We tested the tactic on four proofs. The equilateral triangle construction gives rise to two obligations: the proof that the triangle is indeed equilateral and the proof that it is a maximal triangle inscribed in the original square of the origami. We also looked Eos regular heptagon construction and the Morley's triangle construction proofs.

We believe these are the most relevant and interesting proofs from origami theorem proving that can be analyzed with algebra only. There are two other important origami proofs performed with Eos that we cannot deal with our automatic approach, namely the proof about the sides and layers of the crane origami construction and the proofs about angle trisection constructions. The

1. Start with threshold t equal 1.
2. If t is greater than maximum threshold exit.
3. Find assumptions with number of free variables less than t. Move those assumptions to the assumptions part of the goal-state using `DISCH_TAC` and keep the other assumptions in the goal part using `UNDISCH_TAC`.
4. If there are no assumptions, increase t by 1 and go to step 2.
5. If there are no variables in the assumptions, simplify all the assumptions with `COMPLEX_POLY_CONV`, increase t and go to step 2.
6. For each equation select a variable to normalize
   (a) Choose a variable that has only one linear factor equal 1
   (b) If not possible choose a variable with only linear factors
   (c) If not possible choose a variable with the lowest degree
7. In every equation bring all occurrences of the variable to normalize to one side and the other factors on the other and call `COMPLEX_POLY_CONV` to simplify both sides.
8. If there are equations where only a linear factor remains, divide both sides by the linear factor and simplify the goal and the other equations with it. Set t equal 1 and go to step 3.
9. Increase t by 1 and go to step 2.

Fig. 5. The simplification algorithm performed by the tactic

reason for this is that the first kind includes three dimensional properties that cannot be easily encoded with the same kind of algebraic formulas and the second requires trigonometry.

In all the proofs that we have tried, the tactic did reduce the number of equations and inequalities and the number of quantified variables. In case of the equilateral proof it was able to reduce the 35 equations to 5 equations. For the maximal proof it reduced 37 (in)equations to 9 (in)equations. For the regular heptagon proof it reduced 63 equations to 12 equations. In case of the Morley's triangle construction the tactic would only eliminate 8 simple linear equations but would not be able to progress further because of a bit amount of inequalities. This construction has 27 cases and only for 9 of them the triangle is indeed a Morley's triangle. This means that there are 6 inequalities that limit the effectiveness of the tactic.

For all the above examples the tactic run-time is less than one second on a modern computer. Only in the equilateral triangle proofs the Gröbner bases decision procedure is able to finish the proof automatically. This takes 18 seconds.

In case of the regular heptagon construction, preprocessing reduces an original system of 63 equations to 12 equations. At this stage the Gröbner decision procedures present in the proof assistants are still too weak to finish the goal and additional manual solving of higher-level equations is needed. Finishing the proof manually is complicated, since processing higher order equations manually is cumbersome. When processing the equations in a computer algebra system, irrational numbers and their linear factors are automatically moved from denominators to numerators by multiplying the equations by appropriate factors.

4.2 Inequalities

During the construction of an origami, some of the fold operations have a number of solutions. When the user interactively requests such a fold in an EOS construction, the system will in turn ask which of the two of three fold lines is the intended one. This is then turned in a geometric constraint that a point is included in a certain segment.

Similarly inequalities may come from assumptions about points being on the origami (some approaches to computational origami assume that the original paper is infinite; however after performing folds the area on which points can be considered becomes bounded) or even from the property to be proved (for example the case of a constructed triangle being the maximal one). Such constraints are naturally easier provided as inequalities and therefore cannot be directly used in conjunction with the Gröbner bases algorithm.

McLaughlin and Harrison [16] have implemented a proof producing CAD algorithm that follows the Hörmander's CAD algorithm. We tested the efficiency of the algorithm for practical problems and unfortunately it is too slow to be used; in fact we were not able to solve problems of size 3. However the procedure comes useful when solving second order equations. Still to be useful in a manual formalization, additional tactics for normalizing equations with irrational fractions would be needed.

5 Related Work

As a part of the ForMath project, a proof producing CAD in Coq has been implemented [4]. Its efficiency is however not enough to perform proofs as shown in this paper. We also tried the `nsatz` tactic present in Coq [20]. It has the advantage of using reflection to obtain very efficient proofs. Still many of the origami goals require complex numbers and the tactic does not work with those; also for the proofs that can be performed with real numbers (for example the simplified system of 5 equations for the equilateral triangle proof) the tactic is not able to solve the goal.

A Common-Lisp implementation of the Buchberger algorithm has been verified in ACL2 [17]. There are numerous bridges between proof assistants and computer algebra systems (for example [1]). Such architectures have different degree of trust; the proof of the computer algebra is either used as an oracle or as a hint and recomputed. In this research we aim at a complete formalization done in a proof assistant.

There are many programs for proving geometrical properties. Certain programs are able to export goals to proof assistants to be able to formally certify the constructions. Examples of those are GeoProof [18] or GCL [10]. We have not known of any origami proofs performed in this way.

6 Outlook

There are many decision procedures for real and complex arithmetic present in proof assistant libraries and extensions. They are certainly useful, but they still require special goal shapes and are very slow in comparison with the algorithms present in computer algebra.

We have shown how the goals that arise in origami proofs can be automatically simplified and some of those can be immediately solved by proof assistant decision procedures possibly with minor manual interaction. We compared the efficiency of implementations of Gröbner bases algorithm in HOL Light, Isabelle and Mathematica.

The developed equation system simplifying tactic is available as a part of the CAS-conversion [12]. As such it is both available in the prototype certified computer algebra system inside HOL Light, and as a part of the general CAS simplification procedure.

```
lemma equilateral_triangle:
fixes
      A B C D E F G H J K :: point
  and k l m n :: line
assumes
      a1: "A = (0, 0)"
  and a2: "B = (1, 0)"
  and a3: "C = (1, 1)"
  and a4: "D = (0, 1)"
  and a5: "k = fold2 A D"
  and a6: "E = intersect k A D"
  and a7: "F = intersect k B C"
  and a8: "l = fold2 A B"
  and a9: "H = intersect l D C"
  and aa: "G = intersect l E F"
  and ab: "m ∈ fold5 D l A"
  and ac: "J = intersect m C D"
  and ad: "n ∈ fold5 B k A"
  and ae: "K = intersect n B C"
shows
      "dist A J = dist J K"
  and "dist J K = dist A K"
```

Fig. 6. With a formalization of the origami axiom system, a sequence of unique folds and intersections can be described in a goalstate.

6.1 Future Work

In this paper we assumed the correctness of the part of the Eos system that gathers the geometric conditions and translates them to an algebraic formula. Origami operations are usually expressed in terms of numbers of possible fold lines that satisfy certain constraints. For a proof assistant development, for every operation equivalent algebraic constraints could be proved. The original constraints of the operations are expressed using geometry, and it is also possible to transform those into geometrical constraints.

The principles of folding could be formalized in a proof assistant together with the basic properties of the fold lines that are implied by particular folds. Those properties have been proved geometrically only on paper (for example in [9]). Creating a development like this would be a basis for a formalized origami theory. For operations that give rise to a single possible fold line, this is straightforward; however certain origami fold operations are known to be equivalent to solving higher order equations. In the Equilateral triangle construction we use two of the Huzita's fold principles: `fold2` and `fold5`. The first of those superposes one point onto a different point creating only one fold line. However the latter one superposes one point onto a line and making the crease pass through another point (tangent from a point to a parabola) giving two possible fold lines in the general case. When the user performs such a fold interactively, Eos presents the user with a choice visually. In a proof assistant, one needs a formal statement that will be known to limit the equation to a single solution.

In case of the equilateral triangle construction we can limit the result of a fold operation by adding intersection constraints. With such a mechanism we can imagine the formal statement in a proof assistant to look like on Figure

References

[1] A. Adams, M. Dunstan, H. Gottliebsen, T. Kelsey, U. Martin, and S. Owre. Computer algebra meets automated theorem proving: Integrating Maple and PVS. In R. J. Boulton and P. B. Jackson, editors, *TPHOLs*, volume 2152 of *Lecture Notes in Computer Science*, pages 27–42. Springer, 2001.

[2] H. Aslaksen. Multiple-valued complex functions and computer algebra. *SIGSAM Bulletin (ACM Special Interest Group on Symbolic and Algebraic Manipulation)*, 30(2):12–20, June 1996.

[3] B. Barras, S. Boutin, C. Cornes, J. Courant, Y. Coscoy, D. Delahaye, D. de Rauglaudre, J.C. Filliâtre, E. Giménez, H. Herbelin, et al. *The Coq Proof Assistant Reference Manual Version 8.3.* LogiCal project, 2010. `http://coq.inria.fr/refman/`.

[4] C. Cohen and A. Mahboubi. A formal quantifier elimination for algebraically closed fields. In S. Autexier, J. Calmet, D. Delahaye, P. D. F. Ion, L. Rideau, R. Rioboo, and A. P. Sexton, editors, *AISC/MKM/Calculemus*, volume 6167 of *Lecture Notes in Computer Science*, pages 189–203. Springer, 2010.

[5] F. Ghourabi, T. Ida, and A. Kasem. Proof documents for automated origami theorem proving. Submitted to ADG 2011 post-proceedings.

[6] J. Harrison. HOL Light: A tutorial introduction. In M. Srivas and A. Camilleri, editors, *Proceedings of the First International Conference on Formal Methods in Computer-Aided Design (FMCAD'96)*, volume 1166 of *LNCS*, pages 265–269. Springer-Verlag, 1996.

[7] H. Huzita. Axiomatic development of origami geometry. In H. Huzita, editor, *Proceedings of the First International Meeting of Origami Science and Technology*, pages 143–158, 1989.

[8] H. Huzita. The trisection of a given angle solved by the geometry of origami. In H. Huzita, editor, *Proceedings of the First International Meeting of Origami Science and Technology*, pages 195–214, 1989.

[9] T. Ida, F. Ghourabi, A. Kasem, and C. Kaliszyk. Towards theory of origami fold. Submitted to ISSAC 2011.

[10] P. Janicic. Geometry constructions language. *J. Autom. Reasoning*, 44(1-2):3–24, 2010.

[11] J. Justin. Résolution par le pliage de l'équation du troisième degré et applications géométriques. In H. Huzita, editor, *Proceedings of the First International Meeting of Origami Science and Technology*, pages 251–261, 1989.

[12] C. Kaliszyk. Prototype computer algebra system in HOL Light. `http://score.cs.tsukuba.ac.jp/~kaliszyk/holcas.php`.

[13] C. Kaliszyk and F. Wiedijk. Certified computer algebra on top of an interactive theorem prover. In M. Kauers, M. Kerber, R. Miner, and W. Windsteiger, editors, *Calculemus/MKM*, volume 4573 of *Lecture Notes in Computer Science*, pages 94–105. Springer, 2007.

[14] A. Kasem and T. Ida. Computational origami environment on the Web. *Frontiers of Computer Science in China*, 2(1):39–54, 2008.

[15] A. Kasem, T. Ida, H. Takahashi, M. Marin, and F. Ghourabi. E-origami system Eos. In *Proceedings of the Annual Symposium of Japan Society for Software Science and Technology*, Tokyo, Japan, September 2006. JSSST.

[16] S. McLaughlin and J. Harrison. A proof-producing decision procedure for real arithmetic. In R. Nieuwenhuis, editor, *CADE*, volume 3632 of *Lecture Notes in Computer Science*, pages 295–314. Springer, 2005.

[17] I. Medina-Bulo, F. Palomo-Lozano, and J.-L. Ruiz-Reina. A verified Common Lisp implementation of Buchberger's algorithm in ACL2. *J. Symb. Comput.*, 45(1):96–123, 2010.

[18] J. Narboux. A graphical user interface for formal proofs in geometry. *J. Autom. Reasoning*, 39(2):161–180, 2007.

[19] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.

[20] L. Pottier. Connecting Gröbner bases programs with Coq to do proofs in algebra, geometry and arithmetics. In P. Rudnicki, G. Sutcliffe, B. Konev, R. A. Schmidt, and S. Schulz, editors, *LPAR Workshops*, volume 418 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[21] M. Wester, editor. *Contents of Computer Algebra Systems: A Practical Guide*, chapter A Critique of the Mathematical Abilities of CA Systems. John Wiley & Sons, Chichester, United Kingdom, 1999.