




Computational Origami Construction as Constraint Solving and Rewriting

tion and similar papers at core.ac.uk

brought to you by  CORE

provided by Elsevier - Publisher Connector

Hidekazu Takahashi^{1,2} and Fadoua Ghoura^{1,2}

Department of Computer Science, University of Tsukuba, Tsukuba 305-8357, Japan

Abstract

Computational origami is the computer assisted study of mathematical and computational aspects of origami. An origami is constructed by a finite sequence of fold steps, each consisting in folding along a fold line. We base the fold methods on Huzita's axiomatization, and show how folding an origami can be formulated by a conditional rewrite system. A rewriting sequence of origami structures is viewed as an abstraction of origami construction. We also explain how the basic concepts of constraint and functional and logic programming are related to this computational construction. Our approach is not only useful for computational construction of an origami, but it leads us to automated theorem proving of the correctness of the origami construction.

Keywords: computational origami, constraint solving, rewriting, functional logic programming, automated theorem proving

1 Introduction

We are interested in programming support for computational origami, the computer assisted study of origami. An origami is constructed by a finite sequence of fold steps, each consisting in folding along a fold line. We use several admitted fold operations to find a fold line and then we fold an origami along the fold line.

As Euclidean postulates are the basis of Euclidean geometry, origami can be based on a formal system. Huzita in 1989 proposed an axiomatization of origami [5]. Since then, his axiomatization has been studied extensively [1,3,4]. We base our study of origami on Huzita's axiomatization. We will formulate Huzita's axiomatization first in the language of first-order predicate logic and then as a conditional rewrite system. A rewriting sequence of origami structures is seen as an abstraction

¹ Emails: {ida, mmarin, hidekazu, ghourabi}@score.cs.tsukuba.ac.jp

² This research is supported by the JSPS Grants-in-Aid for Scientific Research (B) No. 17300004 and for Exploratory Research No. 19650001.

of the origami construction. In so doing, we will be able to turn the declarative statements about the origami foldability to computing statements, i.e. program.

We use an origami construction of Morley’s triangle as an illustrative example of our study. We also explain how the basic concepts of constraint and rewrite-based programming are related to this computational construction. Our approach is not only useful for computational construction of origami, but it leads to automated theorem proving of correctness of the construction. The rest of the paper is organized as follows. Section 2 gives the basic notions and notations used in our modeling of origami. In Sect. 3, we formalize the basic fold operations that Huzita proposed. Then in Sect. 4, we formalize the fold. In Sect. 5, we give the construction of Morley’s triangle. In Sect. 6, we briefly describe the current programming and computational capabilities of our computational origami environment *Eos*, and discuss desirable new features. In Sect. 7, we summarize our work and indicate a direction of further research.

2 Basics of origami modeling

We give the notions about the geometric objects of our study and notations used in this paper. Thorough descriptions of geometric notions in plane geometry are treated in standard textbooks [11,13].

A point is the basic object we use without definition. We denote points by P_1, \dots, P_n . By $\underline{P_{i,j}}$, where $i \leq j$, we denote the sequence P_i, \dots, P_j of points. When $i = 1$, we omit i and write $\underline{P_j}$.

By $\langle \underline{P_n} \rangle$, we denote several geometric objects that are the ingredients of origami. A ray, i.e. a directed line segment, is represented by a structure $\langle \underline{P_1, P_2} \rangle$, where $\underline{P_1}$ and $\underline{P_2}$ are the points that the line passes through. A ray is also denoted by $\overrightarrow{P_1 P_2}$.

An $n(n \geq 3)$ -gon, i.e. a simple n -edge polygon (polygon consisting of n edges none of which intersect), is represented by a structure $\langle \underline{P_1, \dots, P_n} \rangle$ made of a sequence of vertices P_1, \dots, P_n , with the following property: $\langle \underline{P_1, \dots, P_n} \rangle$ and its cyclic permutation $\langle \underline{P_{1+k}, \dots, P_{n+k}} \rangle$ for arbitrary $k \geq 1$ are the same³.

The degenerate cases of $P_1 = P_2$ of the ray, and of $P_i = P_{i+1}$ for some i of the n -gon are not specifically treated as this kind of rigor is not what we pursue in this paper. It is assumed that the duplication of the same points in the those structures are automatically removed when they are used.

Definition 2.1 (Face) *A face is a convex n -gon.*

A face $f = \langle \underline{P_n} \rangle$ is up if the vertices P_1, \dots, P_n are arranged counter-clockwise, otherwise down. Because of the convexity of the n -gon, we can easily determine whether the vertices $\underline{P_n}$ are arranged clockwise or counter-clockwise.

Definition 2.2 (Face division) *Let f be $\langle \underline{P_n} \rangle$. Suppose that points X and Y lie on the ray $\overrightarrow{P_i P_{i+1}}$ and $\overrightarrow{P_j P_{j+1}}$, respectively, where $i < j$. The division of the face f by the ray \overrightarrow{XY} , written as $\delta_{\overrightarrow{XY}}(f)$, is defined as follows:*

³ The index addition is performed in modulo the number of elements of the sequence.

Case 1: \overrightarrow{XY} does not overlap with one of the edges of f

$$\delta_{\overrightarrow{XY}}(f) = \langle f_1, f_2 \rangle$$

where $f_1 = \langle Y, X, \underline{P_{i+1,j}} \rangle$ and $f_2 = \langle X, Y, \underline{P_{j+1,n+i}} \rangle$.

Note that $\langle Y, X, \underline{P_{i+1,j}} \rangle$ and/or $\langle X, Y, \underline{P_{j+1,n+i}} \rangle$ would become degenerate cases if the points X and/or Y are the vertices of the n -gon to be divided.

Case 2: \overrightarrow{XY} overlaps with one of the edges of f

$$\delta_{\overrightarrow{XY}}(f) = \langle f, \langle X, Y \rangle \rangle \text{ or } \langle \langle Y, X \rangle, f \rangle.$$

The division of the front side of the face is illustrated in Fig. 1.

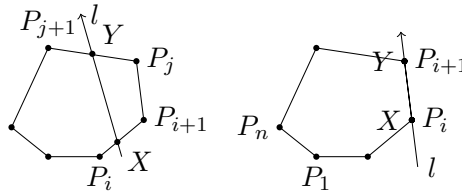


Fig. 1. Face division.

The application of $\delta_{\overrightarrow{XY}}$ to up face f is a pair consisting of geometric objects f_1 and f_2 , and f_1 is to the right of \overrightarrow{XY} and f_2 to the left of \overrightarrow{XY} .

Definition 2.3 (Face rotation) For a point P , $\rho_r^\theta(P)$ is the rotation of P by an angle θ along a ray r . ρ_r^θ is extended to a face. Namely, if $f = \langle P_1, \dots, P_n \rangle$ then $\rho_r^\theta(f) = \langle \rho_r^\theta(P_1), \dots, \rho_r^\theta(P_n) \rangle$.

In origami folds, faces are divided before rotation. Namely, each face f subjected to fold is first divided into f_1 and f_2 , i.e. $\delta_r(f) = \langle f_1, f_2 \rangle$, and then f_1 is rotated along r by applying ρ_r^θ to f_1 .

Definition 2.4 (Abstract Origami) An abstract origami is a structure (Π, \succ, \sim) , where Π is the set of faces, \succ is an overlay relation on Π and \sim is an adjacency relation on Π . We hereafter call the abstract origami simply an origami.

Regarding the overlay and adjacency relations, we do not details them as we discussed elsewhere [7]. We use \mathcal{O} to denote a set of origami's. When we make a fold, we specify which faces we will fold. The set of faces that we are interested in folding is called the set of the faces of concern.

Definition 2.5 (Single-step fold) Let $O_1, O_2 \in \mathcal{O}$ where $O_1 = (\Pi_1, \succ_1, \sim_1)$. An origami single-step fold from O_1 to O_2 is defined as a relation:

$$O_1 \rightsquigarrow_{\mathcal{F}, r, \theta} O_2$$

where $\mathcal{F} \subseteq \Pi_1$ is the set of faces of concern, r is a fold ray, and θ is the angle of rotation.

$\mathcal{V}(O)$ denotes the set of points in O . When the number of divided faces is k , and r is \overrightarrow{XY} , we have:

$$\mathcal{V}(O_2) = \mathcal{V}(O_1) \cup \{\underline{X}_k, \underline{Y}_k\},$$

where the points $\underline{X}_k, \underline{Y}_k$ are created in this fold⁴.

Definition 2.6 (Origami construction) Let $O_1, \dots, O_n \in \mathcal{O}$. A fold sequence

$$O_1 \rightsquigarrow_{\mathcal{F}_1, r_1, \theta_1} O_2 \rightsquigarrow_{\mathcal{F}_2, r_2, \theta_2} \cdots \rightsquigarrow_{\mathcal{F}_{n-1}, r_{n-1}, \theta_{n-1}} O_n$$

is called an origami construction from O_1 to O_n . We may also write $O_1 \rightsquigarrow^* O_n$.

3 Huzita's axiomatization of origami construction

3.1 Huzita's basic folds in brief

Huzita proposed the following operations as the basic folds:

- (O1) We can make a fold along a line that passes through given points P and Q .
- (O2) We can make a fold along the line that superposes given points P and Q .
- (O3) We can make a fold along a line that superposes two given lines m and n .
- (O4) We can make a fold along a line that is perpendicular to a given line m , and passes through a given point P .
- (O5) We can make a fold along a line that passes through a given point Q and superposes a given point P on a given line m .
- (O6) We can make a fold along a line that superposes a given point P on a given line m and a given point Q on a given line n .

Remark 3.1

- (i) When we say that we are given n points, those n points are distinct and they are all on the origami. Likewise, when we are given n lines, those n lines are distinct and they pass through two distinct points on the origami.
- (ii) In the case of (O3), there exists one or two fold lines.
- (iii) In the cases of (O5) and (O6), there may be multiple fold lines or none. It is decidable whether a fold line satisfying the properties exists or not.

The statements (O1)~(O6) are often called Huzita's axioms in the literature. The statements (O5) and (O6) are not axioms in mathematical sense since we have situations where a fold is impossible to satisfy the properties. To be rigorous, we have to understand the statements (O5) and (O6) together with the above remark (iii) in order to call them axioms.

⁴ We let $X_1 = X$ and $Y_1 = Y$.

3.2 Huzita's axioms in first-order language

Let P and Q denote points, and l , m and n denote lines. Corresponding to the statements (O1) ~ (O6) in the previous subsection, we have the following formulas.

$$(A1) \quad \forall P, Q \exists l \text{ OnLine}(P, l) \wedge \text{OnLine}(Q, l).$$

$$(A2) \quad \forall P, Q \exists l \text{ reflection}(P, l) = Q.$$

$$(A3) \quad \forall m, n \exists l \forall P \text{ OnLine}(P, m) \Rightarrow \text{OnLine}(\text{reflection}(P, l), n).$$

$$(A4) \quad \forall P, m \exists l \forall Q \text{ OnLine}(P, l) \wedge (\text{OnLine}(Q, m) \Rightarrow \\ \text{OnLine}(\text{reflection}(Q, l), m)).$$

$$(A5) \quad \forall P, Q, m \exists l \text{ OnLine}(Q, l) \wedge \text{OnLine}(\text{reflection}(P, l), m).$$

$$(A6) \quad \forall P, Q, m, n \exists l \text{ OnLine}(\text{reflection}(P, l), m) \wedge \text{OnLine}(\text{reflection}(Q, l), n).$$

All the predicates and functions have to be given precise meaning in the proper semantic domain. A rigorous treatment of the semantics of Huzita's axioms is given in [4]. We give informal meaning to those symbols based on the geometrical intuition, as we go along.

We first fix the domain of interpretation to be the domain of algebraic numbers. We will see shortly that if we represent a line as a term $\text{line}(a, b, c)$ that passes through two given points defining the line, we are always able to find the values of a , b and c in formulas (A1) ~ (A6). However, the values may be complex numbers. In such cases, we can not give geometric meaning in the Euclidean plane. This observation supplements Remark (iii) in Subsection 3.1.

Formula (A1) states that for any points P and Q , there exists a line l such that P is on l and Q is on l . In formula (A2), the predicate $\text{reflection}(P, l) = Q$ states that the reflection of a point P in a line l is the point Q .

3.3 Huzita's axioms in rewrite rules

To reason about the computation implicit in the Huzita's axioms, the description in first-order predicate logic is not ideal. When we prove the correctness of an existentially quantified formula, often we are not only interested in the satisfiability of the formula, but in the substitution made to the existential variables. Obtaining the values instantiated in the existential variables is the intended purpose of the proving, often called solving rather than proving, however.

A rewrite rule clearly serves better for this intention, if we are more interested in computation rather than declarative statements. Let us consider basic fold operation (O1) to begin with. The basic idea in transcribing a first-order formula to a rewrite rule is to define a new relation $\mathcal{R}(P, Q, l)$, which is **true** if $\text{OnLine}(P, l) \wedge \text{OnLine}(Q, l)$ holds. Namely, we state

$$\forall P, Q, l (\mathcal{R}(P, Q, l) \Leftarrow \text{OnLine}(P, l) \wedge \text{OnLine}(Q, l)).$$

This means that for any combinations of values of P , Q and l , if $\text{OnLine}(P, l) \wedge$

$\text{OnLine}(Q, l)$ holds, then $\mathcal{R}(P, Q, l)$ holds. In particular if for any combinations of values of P , Q and l for which formula (A1) holds, then $\mathcal{R}(P, Q, l)$ holds. This can be written as a 3-CTRS (type 3 Conditional Term Rewrite System)[10].

$$\text{foldTh}(P, Q) \rightarrow l \Leftarrow \text{OnLine}(P, l) \wedge \text{OnLine}(Q, l).$$

Here, we have introduced a new function symbol foldTh , and replaced the relation $\mathcal{R}(P, Q, l)$ by the rewrite relation $\text{foldTh}(P, Q) \rightarrow l$.

We then want to define OnLine as a rewrite rule. At this point we need to commit ourselves to a certain representation of points and lines. In this paper, we use the Cartesian coordinate system to represent points and a linear equation to represent a line. We use constructor symbols point and line to represent them in term structures.

$$\text{OnLine}(\text{point}(x, y), \text{line}(a, b, c)) \rightarrow \mathbf{true} \Leftarrow ax + by + c = 0$$

To make the equation $ax + by + c = 0$ always represent a line, we need to impose a constraint on the coefficients a , b , and c . This constraint is stated in the predicate $\text{Coeff}(\text{line}(a, b, c))$. One possible definition of $\text{Coeff}(\text{line}(a, b, c))$ is the following:

$$(-1 + b)b = 0 \wedge (-1 + a)(-1 + b) = 0.$$

A slightly complicated re-reasoning of the above rewrite rule leads to the following rewrite rule.

$$\text{foldTh}(P, Q) \rightarrow l \Leftarrow \text{OnLine}(P, l) \wedge \text{OnLine}(Q, l) \wedge \text{Coeff}(l).$$

(O2) is easily transcribed to the following rewrite rule:

$$(1) \quad \text{foldBr}(P, Q) \rightarrow l \Leftarrow \text{reflection}(P, l) = Q \wedge \text{Coeff}(l),$$

where $\text{reflection}(P, l)$ is written as a rewrite rule:

$$\begin{aligned} & \text{reflection}(\text{point}(x, y), \text{line}(a, b, c)) \\ & \rightarrow \text{point} \left(\frac{-a^2x + b^2x - 2a(c + by)}{a^2 + b^2}, \frac{-2b(c + ax) + a^2y - b^2y}{a^2 + b^2} \right). \end{aligned}$$

Since (O3) is already a formula of implication, transcribing (O3) needs a bit of algebraic manipulation. Let P , l , m , n be $\text{point}(x, y)$, $\text{line}(a, b, c)$, $\text{line}(a_1, b_1, c_1)$, $\text{line}(a_2, b_2, c_2)$, respectively. Then the algebraic interpretation of the right-hand side (of \Rightarrow) in formula (A3) is

$$c_2 + \frac{b_2(-2b(c + ax) + a^2y - b^2y)}{a^2 + b^2} + \frac{a_2(-a^2x + b^2x - 2a(c + by))}{a^2 + b^2} = 0,$$

which is equivalent to the formula $a'x + b'y + c' = 0$, where

$$\begin{aligned} a' &= -a^2a_2 + a_2b^2 - 2abb_2 \\ b' &= -2aa_2b + a^2b_2 - b^2b_2 \\ c' &= -2aa_2c - 2bb_2c + a^2c_2 + b^2c_2 \end{aligned}$$

Let l' be the line $\text{line}(a', b', c')$. Then it is easy to see that

$$(a', b', c') = -k \cdot (a_1, b_1, c_1) \text{ for some } k.$$

From the above formula, we can obtain the following rewrite rules:

$$\begin{aligned} \text{foldBrLine}(\text{line}(a_1, b_1, c_1), \text{line}(a_2, b_2, c_2)) &\rightarrow l \Leftarrow l = \text{line}(a, b, c) \\ \wedge -a^2a_2 + a_2b^2 - 2abb_2 + ka_1 = 0 &\wedge -2aa_2b + a^2b_2 - b^2b_2 + kb_1 = 0 \\ \wedge -2aa_2c - 2bb_2c + a^2c_2 + b^2c_2 + kc_1 = 0 &\wedge \text{Coeff}(l). \end{aligned}$$

Note that the algebraic expressions in the above formula are square-root free. In general, solving the conditional part of this rewrite rule yields two solutions for a , b , c and k , which shows that there are two fold lines that superpose lines m and n .

Similarly, we can obtain the following rewrite rules that correspond to the axioms (O4) \sim (O6), respectively.

$$\begin{aligned} \text{foldPerTh}(P, m) &\rightarrow l \Leftarrow \\ &\text{OnLine}(P, l) \wedge (\text{OnLine}(Q, m) \Rightarrow \text{OnLine}(\text{reflection}(Q, l), m)) \wedge \text{Coeff}(l). \\ \text{foldThBr}(P, Q, m) &\rightarrow l \Leftarrow \text{OnLine}(Q, l) \wedge \text{OnLine}(\text{reflection}(P, l), m) \wedge \text{Coeff}(l). \\ \text{foldBrBr}(P, Q, m, n) &\rightarrow l \Leftarrow \\ &\text{OnLine}(\text{reflection}(P, l), m) \wedge \text{OnLine}(\text{reflection}(Q, l), n) \wedge \text{Coeff}(l). \end{aligned}$$

The conditional part of the rewrite rules will become the conjunction of equations when the rules are applied. In order to perform rewriting of the left-hand side to the righthand side of the rewrite rules, we need to solve the system of equations. This can be done either statically or dynamically. By *statically*, we mean that we can transform the rewrite rule to a new rewrite rule that does not require constraint solving at run time. This will be detailed in the next subsection.

The conditional rewrite system that describes Huzita's axioms can rewrite the same ground term in more than one way. This behavior indicates that some single-step folds can be performed in more than one way. It can be shown that every ground term can be transformed in at most 3 ways. When the rewriting is non-deterministic, it is desirable to allow the user to view the alternatives and to choose one to proceed to the next fold step of the construction.

3.4 Transformation of rewrite rules

In this subsection we will show an example of static transformation of rewrite rules. Rewriting of the term $\text{foldBr}(P, Q)$ by the rewrite rule (1) for (O2) requires con-

straint solving of the set of equations

$$\left\{ \begin{aligned} (-1+a)(-1+b) &= 0, (-1+b)b = 0, \\ \frac{-2b(c+ax_1) + a^2y_1 - b^2y_1}{a^2 + b^2} &= y_2, \\ \frac{-a^2x_1 + b^2x_1 - 2a(c+by_1)}{a^2 + b^2} &= x_2 \end{aligned} \right\}$$

for a , b , and c . The solution of the above system is

$$\left\{ a \rightarrow 1, b \rightarrow 0, c \rightarrow \frac{-x_1 - x_2}{2} \right\}$$

if $y_1 = y_2$, and

$$\left\{ a \rightarrow \frac{x_1 - x_2}{y_1 - y_2}, b \rightarrow 1, c \rightarrow \frac{-x_1^2 + x_2^2 - y_1^2 + y_2^2}{2(y_1 - y_2)} \right\}$$

if $y_1 \neq y_2$. Therefore, the rewrite rule (1) is reduced to

$$\text{foldBr}(\text{point}(x_1, y), \text{point}(x_2, y)) \rightarrow \text{line}(1, 0, \frac{-x_1 - x_2}{2})$$

$$\text{foldBr}(\text{point}(x_1, y_1), \text{point}(x_2, y_2)) \rightarrow \text{line}\left(\frac{x_1 - x_2}{y_1 - y_2}, 1, \frac{-x_1^2 + x_2^2 - y_1^2 + y_2^2}{2(y_1 - y_2)}\right)$$

If rewriting is performed in a symbolic computation environment such as of *Mathematica* [14], the above transformation can be done semi-automatically. So on one hand it is possible to reduce the run time computing cost by static transformation. We need not solve the same constraints repeatedly.

On the other hand, for theorem proving, we need to maintain the set of equations symbolically. Origami construction is interactive. As our experiences with *Eos* [8] show, constraint solving is not prohibitively expensive, even when we use usual laptop computers. Therefore solving constraints dynamically is also feasible.

4 Formalizing Fold

Now let us consider formalization of the essential part of origami, i.e. fold. Each fold requires the following parameters: the set \mathcal{F} of faces of concern, the rotation angle θ (either π or $-\pi$ in this paper) and the basic fold operation to apply together with appropriate parameters of lines and points given to the operation.

A fold operation can be decomposed into seven computational steps:

- (F-1) Choose a basic fold operation from among (O1)~(O6).
- (F-2) Find a fold line l according to the fold method.
- (F-3) Specify the set \mathcal{F} of the faces of concern and the ray r obtained from l .
- (F-4) Compute the set \mathcal{G} of all the faces that are affected by the fold.
- (F-5) Divide the faces by l and classify all the obtained faces into “to be moved” and “to be non-moved”.

- (F-6) Compute the overlay and adjacency relations.
- (F-7) Rotate the “to be moved” faces along r by the angle θ .

In this paper we consider an origami construction where each fold operation is followed by an unfold, so that we always make a fold on a square piece of origami paper. This is the case of the construction of Morley’s triangle. The modeling of the folds in general cases requires the analysis of overlay and adjacency relations on the faces at step (F-4). Then the faces are divided and classified at step (F-5). At step (F-6) the overlay and adjacency relations are computed using the relations computed in the previous origami construction step. Finally, at step (F-7), the “to be moved” faces are rotated. The resulting origami forms layers of faces and exhibits an artistic shape.

5 Construction of Morley’s triangle

In this section we will show how to formalize the origami construction of Morley’s triangle [2]. Given an arbitrary triangle, Morley’s triangle is the triangle inside the given triangle, formed by the three intersections of neighboring trisectors of the angles of the given triangle. The Morley’s triangle is always equilateral. This is an interesting example of a construction that can be realized with only a piece of origami paper by following simple constructive steps [6]. Since angle trisection cannot be realized by means of a compass and a ruler, it is a surprising result, which shows that an origami construction exceeds the capabilities of the Euclidean construction by the compass and ruler.

5.1 Overview of the construction

Figure 2 illustrates the construction of Morley’s triangle. We need 28 fold steps to obtain Morley’s triangle. The first four steps are preparation of the construction. Morley’s triangle is clearly marked in thick solid line (red if colored) in O_{29} . Suppose we are given an initial origami ABCD with a point E. The thin solid line segments (light brown if colored) EA and EB in Fig. 2 are shown for the convenience of readers, and are not part of the construction. Let this origami be O_5 . The construction proceeds by trisecting the internal angles of $\triangle ABE$. The points B1, C1 and S are the intersections of the neighboring trisectors. Morley’s theorem states that $\triangle B1C1S$ is equilateral.

The construction is described as follows:

$$\begin{aligned}
 O_5 \rightsquigarrow_{\{ABCD\}, \overrightarrow{FG}} O_6 \rightsquigarrow_{\{GFAB\}, \overrightarrow{GF}} O_7 \rightsquigarrow_{\{GFAB\}, \overrightarrow{HI}} O_8 \rightsquigarrow_{\{IHDA\}, \overrightarrow{IH}} \\
 O_9 \rightsquigarrow_{\{FGAB\}, \overrightarrow{AL}} O_{10} \rightsquigarrow_{\{ABL\}, \overrightarrow{LA}} \\
 O_{11} \rightsquigarrow_{\{ABL\}, \overrightarrow{AM}} O_{12} \rightsquigarrow_{\{ABM\}, \overrightarrow{MA}} O_{13} \rightsquigarrow \dots \rightsquigarrow O_{29}
 \end{aligned}$$

Here we omit θ since θ is always π . \mathcal{F} is specified by the set of the names of the

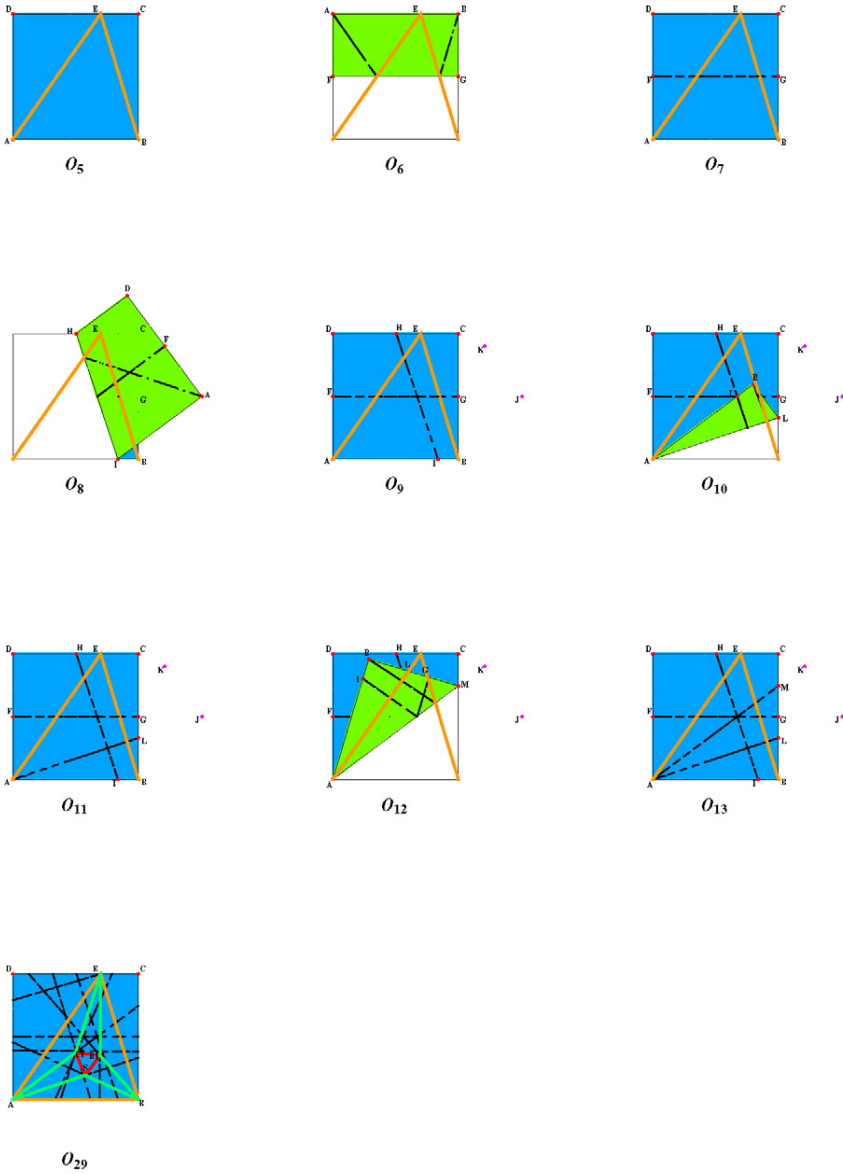


Fig. 2. Origami construction of Morley's triangle

faces of concern. Here, the name of $\langle P_i, \dots, P_j \rangle$ is $P_i \cdots P_j$.

5.2 Trisection of an angle

$$\begin{aligned} &\{-1 + \eta_3, (-1 + b_8)b_8, (-1 + a_8)(-1 + b_8), (-1 + b_9)b_9, \\ &(-1 + a_9)(-1 + b_9), c_9 + a_9x_1 + b_9y_1, \\ &2a_9c_9 + (a_9^2 + b_9^2)x_6, c_9 + a_9x_2 + b_9y_2, y_2, \\ &c_8 + a_8x_3 + b_8y_3, b_9^2(x_3 - x_8) - a_9(2c_9 + a_9(x_3 + x_8) + 2b_9y_3), c_8 + a_8x_4 + b_8y_4, \\ &-1 + x_4, 2b_9c_9(x_3 - x_4) - 2a_9c_9(y_3 - y_4) + (a_9^2 + b_9^2)(-x_4y_3 + x_3y_4), \\ &-1 + x_5, a_9^2y_6 - b_9(2c_9 + 2a_9x_6 + b_9y_6), 2b_9c_9 + (a_9^2 + b_9^2)y_6, \\ &b_9^2x_6 - a_9(2c_9 + a_9x_6 + 2b_9y_6), -1 + x_7, x_7y_6 - x_6y_7, \\ &a_9^2(y_3 - y_8) - b_9(2c_9 + 2a_9x_3 + b_9(y_3 + y_8)), x_8y_5 - x_5y_8, \\ &a_8, -1 + y_1, x_3, b_8 + 2c_8, -b_9(b_9 + 2c_9)x_{13} + a_9(a_9x_{13} + 2(b_9 + c_9)y_{13})\} \end{aligned}$$

We have A(0, 0), B(1, 0), C(1, 1), D(0,1), E(x_{13} , y_{13}), M(x_5 , y_5), L(x_7 , y_7), G(x_4 , y_4), F(x_8 , y_8), H(x_1 , y_1), I(x_2 , y_2). Lines GF, IH are represented by line(a_8 , b_8 , c_8), line(a_9 , b_9 , c_9), respectively.

Fig. 3. Premise equations for the proof of the trisection of $\angle EAB$

The steps from O_5 to O_{13} correspond to the construction of the trisector of $\angle EAB$. The rest is for constructing the other two trisectors. Let us see in more detail how the computation proceeds. The rewrite rules applied in the steps from O_5 to O_{13} is foldBr, unfold, foldBrBr, unfold, foldTh, unfold, foldTh and unfold. During the construction, the system solved the constraints numerically, and at the same time it saved the constraints in symbolic expression. Therefore, at the step where O_{13} is obtained, we can generate the set \mathcal{C} of the equations, and can prove that we indeed have constructed the trisector. The set \mathcal{C} is given in Fig. 3. We abuse the notation of a set of polynomials and identify it with the system of equations.

The conclusion of the proposition that we want to verify must then be formulated. For the proof based on the Gröbner bases method, we need the statement about the trisection in terms of polynomials. We use the square of sin of concerned angles that is defined by Spread in rational trigonometry [13]. Then the conclusion is formulated as follows:

$$\begin{aligned} \text{Spread}(\angle LAB) &= \frac{y_7^2}{x_7^2 + y_7^2}, \\ \text{Spread}(\angle MAL) &= \frac{(x_7y_5 - x_5y_7)^2}{(x_5^2 + y_5^2)(x_7^2 + y_7^2)}, \\ \text{Spread}(\angle EAM) &= \frac{(x_5y_{13} - x_{13}y_5)^2}{(x_{13}^2 + y_{13}^2)(x_5^2 + y_5^2)}, \text{ and} \\ \text{Spread}(\angle LAB) &= \text{Spread}(\angle MAL) = \text{Spread}(\angle EAM) \end{aligned}$$

From the above, we obtain \mathcal{D} given in Fig. 4.

We want to prove

$$(2) \quad \mathcal{C} \Rightarrow \mathcal{D}.$$

Let $\mathcal{C} = \{c_1, \dots, c_n\}$ and $\mathcal{D} = \{d_1, \dots, d_m\}$. The proof of proposition (2) is by contradiction using the theory of Gröbner bases. We compute the Gröbner basis of the set of polynomials

$$(3) \quad \{c_1, \dots, c_n, (\kappa_1 d_1 - 1) \cdots (\kappa_m d_m - 1)\}$$

where $\kappa_1, \dots, \kappa_m$ are newly introduced slack variables.

$$\begin{aligned} & \{-y_5(x_7^2 y_5 - 2x_5 x_7 y_7 - y_5 y_7^2), \\ & \quad - (x_7 y_{13} - x_{13} y_7) \\ & \quad (x_5^2(x_7 y_{13} + x_{13} y_7) - y_5^2(x_7 y_{13} + x_{13} y_7) + x_5(-2x_{13} x_7 y_5 + 2y_{13} y_5 y_7))\} \end{aligned}$$

Fig. 4. Conclusion equations for the proof of the trisection of $\angle EAB$

We consider the set of the polynomials (3) as the system of equations. The non-existence of its solution implies the correctness of the proposition. Namely, if the reduced Gröbner basis of (3) is $\{1\}$, the proposition is proved.

Eos has an interface to assist computational origamists to perform the proofs. To compute the reduced Gröbner basis of (3), it took 0.33 seconds for the proof of the trisector of $\angle EAB$ on a Pentium 1.2 GHz laptop computer equipped with 1 GB memory running under Windows XP.

After trisecting the angles and proving the correctness of the construction of the trisectors, we can go on to prove that the triangle $\triangle B1C1S$ is equilateral. The triangles formed by the three intersections of neighboring trisectors of the angles are not necessary inside the given triangle. Out of the 27 possible triangles, 18 of them are equilateral. Origami experiments are reported in [6] and thorough geometrical investigations are given in [15,12]. For this paper we only give the timing of the proof for particular origami construction when E lies at some point inside the origami. It took 84.49 seconds of computation by the same laptop computer mentioned above.

6 Programming language supports for origami

We have seen the model of computation for computational origami. The model comes with a formal language to reason about the model. However, the formal language is not sufficient for end-users, i.e. origamists, to explore the possibilities of computational origami. The computational origami environment *Eos* developed by us assists the user to perform origami construction and reason about geometric properties. The programming and computational capabilities described in this paper are an abstraction of what have been implemented so far. We need more programming supports for computational origami.

In this section we briefly discuss what we already have and the desiderata for further development. *Eos* provides primitives for the following operations:

- Simulation of the origami construction: The realization of the single-step fold $O_1 \mapsto_{\mathcal{F},r,\theta} O_2$ is achieved by one of the Huzita's axioms, together with appropriate parameters and the additional information needed in the formalization of the fold. For example, the fold using (O1) is realized by a call $\text{HFold}[A, \text{Along} \rightarrow \{P, Q\}]$ and the fold using (O2) is realized by $\text{HFold}[P, Q]$. In the latter case, point P is brought to point Q. In this way, the origamists do not have to know the faces of concern. In the former case the origamists must give point A as the first parameter, such that the system can identify the face of concern. The system processes the input specification, and computes \mathcal{G} and r , and constructs the data structure O_2 for the origami produced by the single-step fold.
- Visualization of the origami O_2 produced by single-step folds $O_1 \mapsto_{\mathcal{F},r,\theta} O_2$.
- Maintaining an algebraic representation of the origami construction, that can be used for proving geometric theorems about origami.
- Providing a set of functions that enable origamists to compute and reason about the geometric objects used in origami.

The implementation of these capabilities requires the support for conditional term rewriting, capabilities of symbolic computation such as Gröbner basis computation and manipulation of polynomials, and graphics processing. *Eos* is implemented in *Mathematica*, since *Mathematica* is based on higher-order term rewriting with functionalities that we described above.

As our experiences grow with many examples of origami's, more functionalities have become needed. From programming point of view, typing and object orientation are highly desirable. However, typing of geometrical objects is non-trivial since the type may change as the shape of an object gets degenerated; for instance a polygon becomes a line when the number of vertices becomes 2. Features proposed by Liang and Wang [9] would have to be taken into account. For computational origami the desired functionalities are also for producing high-quality art pieces of origami. We expect to have better human friendly interface which can be obtained by integrating off-the-shelf software components.

7 Conclusion

We have shown how the origami construction can be modeled by a sequence of fold steps. At each fold step, an origami structure is transformed to a new structure. The fold line along which the fold is made is computed by a 3-CTRS. The rewrite rules of 3-CTRS are derived from Huzita's axioms. The application of the rewrite rules requires constraints solving in order to satisfy the conditions of the rewrite rules.

The numerical solutions are used to simulate the construction of origami, and to visualize the origami shapes at each step. The accumulated constraints are used for proving the geometric properties of the constructed origami.

We argued for the usefulness of the modeling of the construction as a rewrite sequence, and discussed the language features that are essential in the geometrical modeling of origami.

The formalization of origami is an important step towards the study of reverse problems of origami (e.g., finding a shape from the fully unfolded origami with creases), exploration of new construction methods of origami pieces and the study of foldability beyond the power of Huzita’s axiomatization.

References

- [1] R. C. Alperin and R. J. Lang. One-, two-, and multi-fold origami axioms. In *Proceedings of 4th International Conference on Origami, Science, Mathematics and Education (4OSME)*, 2006.
- [2] A. Bogomolny. Morley’s Miracle. 1996. <http://www.cut-the-knot.org/triangle/Morley>, ©1996-2005.
- [3] R. Geretschläger. *Geometric Constructions in Origami*. Morikita Publishing Co., 2002. In Japanese, translation by Hidetoshi Fukagawa.
- [4] F. Ghourabi, T. Ida, H. Takahashi, M. Marin, and A. Kasem. Logical and Algebraic View of Huzita’s Origami Axioms with Applications to Computational Origami. In *Proceedings of the 22nd ACM Symposium on Applied Computing*, pages 767–772. ACM Press, March 2007.
- [5] 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.
- [6] T. Ida, M. Marin, and H. Takahashi. Computational Origami of a Morley’s Triangle. In M. Kohlhasse, editor, *Proceedings of 4th International Conference on Mathematical Knowledge Management (MKM 2005)*, volume 3863 of *LNCS*, page 267. Springer.
- [7] T. Ida, H. Takahashi, M. Marin, and F. Ghourabi. Modelling origami for computational construction and beyond. In O. Gervasi and M. Gavrilova, editors, *International Conference on Computational Science and Its Applications 2007 (ICCSA 2007)*, volume 4151 of *Lecture Notes in Computer Sciences*, pages 653 – 665. Springer-Verlag Berlin Heidelberg.
- [8] T. Ida, H. Takahashi, M. Marin, A. Kasem, and F. Ghourabi. Computational Origami System Eos. In *Proceedings of 4th International Conference on Origami, Science, Mathematics and Education (4OSME)*, page 69, 2006.
- [9] T. Liang and D. Wang. Towards a Geometric-Object-Oriented Language. In *Automated Deduction in Geometry*, volume 3763 of *Lecture Notes in Computer Science*, pages 130–155, 2006.
- [10] A. Middeldorp and E. Hamoen. Completeness results for basic narrowing. *Applicable Algebra in Engineering, Communication and Computing*, 5:213 – 253, 1994.
- [11] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Texts and Monographs in Computer Science. Springer-Verlag, 1985.
- [12] D. Wang. *Elimination Practice: Software Tools and Applications*. Imperial College Press London, 2004.
- [13] N. J. Wildberger. *Divine Proportions*. Wild Egg Pty Ltd, 2005.
- [14] S. Wolfram. *The Mathematica Book*. Wolfram Media, 5th edition, 2003.
- [15] W.T. Wu. Basic principles of mechanical theorem proving in elementary geometry. *Journal of Automated Reasoning*, 2:221–252, 1986.