# Formalizing Polygonal Knot Origami

Tetsuo Ida

*Professor Emeritus*
*University of Tsukuba, Japan*

Fadoua Ghourabi

*Department of Informatics*
*Kwansei Gakuin University, Japan*

Kazuko Takahashi

*Department of Informatics*
*Kwansei Gakuin University, Japan*

**Abstract**

We present computer-assisted construction of regular polygonal knots by origami. The construction is completed with an automated proof based on algebraic methods. Given a rectangular origami or a finite tape, of an adequate length, we can construct the simplest knot by three folds. The shape of the knot is made to be a regular pentagon if we fasten the knot tightly without distorting the tape. We perform the analysis of the knot fold further formally towards the automated construction and verification. In particular, we show the construction and proof of regular pentagonal and heptagonal knots. We employ a software tool called Eos (e-origami system), which incorporates the extension of Huzita's basic fold operations for construction, and Gröbner basis computation for proving. Our study yields more mathematical rigor and in-depth results about the polygonal knots.

*Key words:* computational origami; knot fold; Gröbner bases; geometrical theorem proving;

⋆ This work was supported by JSPS KAKENHI Grant Number 25330007. The second author of this paper is supported by the postdoctoral fellowship at Kwansei Gakuin University.
This is the revised and enhanced version of the earlier papers published in the conference proceedings: (Ghourabi et al., 2013a; Ida et al., 2014).

*Email addresses:* ida@cs.tsukuba.ac.jp (Tetsuo Ida), ghourabi@kwansei.ac.jp (Fadoua Ghourabi), ktaka@kwansei.ac.jp (Kazuko Takahashi).

## 1. Introduction

Given a rectangular origami, i.e. sheet of folding paper of adequate length, we can construct the simplest knot by performing three times of folds. The shape of the knot is made to be a regular pentagon if we fasten the knot tightly without distorting the origami as shown in Fig. 1. The method is so simple that it should be known since the age of early human civilization. However, only from the middle of the 20th century mathematical investigations of the knot fold construction started to appear (e.g. see, (Brunton, 1961; Sakaguchi, 1982; Wells, 1991)).

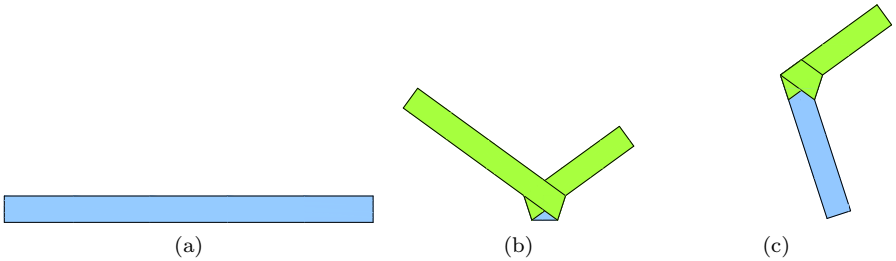
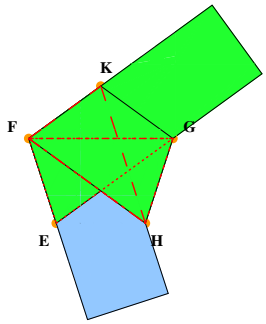



Fig. 1. A regular pentagon obtained by the simplest knot of three folds

In this paper we conduct further a formal analysis of the knot fold for the realization of the computer-assisted construction and verification. The knot fold is decomposed into a sequence of more basic folds, each creating an isosceles triangle as shown in Fig. 2. Three overlapping and congruent isosceles triangles make a pentagonal knot. By superposing the isosceles triangles carefully, we can construct a regular $2n(n \geq 2)+1$-gonal knot in general. If we relax the rigidity of the constructed shape, but yet without breaking the origami, we can make more kinds of regular $n$-gons, some with a hole in the center. We show those properties, focussing on the constructions of regular pentagonal and heptagonal knots.




The regular pentagon knot EHGKF is constructed from the isosceles triangles △GFE, △FHG and △HKF.

Fig. 2. Regular pentagonal knot and overlapping isosceles triangles

Knot fold is interesting because of its familiarity to everyone, of its common use in everyday life and of its simple principle. In everyday life we make knots by various materials such as strings, ropes, cloths and papers, and we can imagine various forms of knots depending on them. To avoid possible misunderstandings arising from the variety of the used substances and to focus on essential properties of knots, mathematical aspects

of knots have been studied deeply since the middle of the 18th century. In this paper we are interested in the concrete shapes created by knotting the paper tape. Hence, we are less concerned with the topological and combinatorial aspects of knots, but are more with the methods of the construction. Furthermore, since a paper tape of a rectangular shape can be constructed from a square sheet of paper (i.e. origami) by repeated folds, we will consider a rectangular origami from the outset. Subsequently, we simply call it a *tape* or an *origami*.

Knot fold is based on physical constraints, i.e. the rigidity and foldability of the paper, unlike Huzita's basic folds that are based on the observable incidence relations over inductively defined points and lines, and on their superpositions (Huzita, 1989). From the origami point of view, the knot fold can be seen as a variant of multi-fold, where we perform folds along multiple fold lines simultaneously (Alperin and Lang, 2009). A general multi-fold is difficult to perform precisely by hand, whereas we will see the knot fold by hand retains a certain degree of precision as far as the constructed shape is concerned.

From algebraic point of view, the knot fold lends itself to a constraint solving problem. If it is formalized algebraically, it results in more precision and rigor, and furthermore a new mathematical view of the subject. Both views are clearly recognized by the use of Eos (e-origami system) under development (Ida et al., 2011, 2006). The knot fold is first specified in the language of Eos, a language of a fragment of many-sorted first-order predicate logic. The specification is transformed into algebraic expressions, which are solved algebraically and numerically for the construction (i.e. for visualization in graphics), and then input to the geometrical prover of Eos that incorporates the methods of Gröbner bases computation.

The rest of the paper is organized as follows. In Sect. 2, we give the notations that we use and present Huzita's basic fold operations that lay the foundation of our study of this subject. In Sect. 3 we discuss the basic geometry underlying the knot fold by treating a pentagonal knot. In Sect. 4, we analyze the knot fold of a regular pentagon, and in Sect. 5 we discuss about its proofs. In Sect. 6, we show the construction and verification of a regular heptagonal knot by Eos. In Sect. 7, we summarize our results and point out directions of further research.

## 2. Preliminaries

In this section we explain the notations that we use in this paper and the basic origami principle.

### *2.1. Notations and programming language*

We deal with plane Euclidean geometry. We deal with only points, segments and lines explicitly during the construction of origamis, and vectors for the verification in addition. Points are denoted by single capital letters (possibly subscripted) of Latin alphabet, e.g. A, B, C, D, X, Y, etc. They are italic in variable and emphasis contexts. Lines are denoted by $m$, $n$, $l$ and $s$. The line passing through points $X$ and $Y$ is denoted by $\overline{XY}$ or simply $XY$. We use the same notation to denote the (closed) segment between points $X$ and $Y$. Although which of the two notations $XY$ refers to should be clear from the context, we may precede the words 'segment' or 'line' to it. The vector from point $X$ to point $Y$

is denoted by $\overrightarrow{XY}$. The distance between two points $X$ and $Y$ is denoted by $|XY|$. The reflection of point $X$ across line $m$ is denote by $X^m$.

A mathematical model of origami is called an *abstract origami*. It is a triple $(\mathsf{O}, \backsim, \succ)$, where $\mathsf{O}$ is a set of polygons called *faces*, $\backsim$ and $\succ$ are binary relations on $\mathsf{O}$, each denoting *adjacent* and *superposition* relations. A fold of the origami changes the triple and creates a new origami. An abstract origami is concretized when points on the faces are given concrete values, i.e. their $x$- and $y$-coordinates. The concretized origami is simply called *origami*. It is also called *configuration* when we focus on the triple $(\mathsf{O}, \backsim, \succ)$ with the points on the faces being assigned concrete values. An origami is denoted by $\mathcal{O}$. We assume a distinguished origami called an *initial origami*, which is a square sheet of paper with four corner points A, B, C and D, enumerated counterclockwise. Later, we relax the condition that the face of the initial origami is a square, and simply assume that the face is a convex rectangle.

We use the Cartesian coordinate system. A point at coordinate $(x, y)$ can be written as $\mathrm{Point}(x, y)$ using the constructor Point. When that point is given a name, say E, it is also written as $\mathrm{E}(x, y)$. Two points $P$ and $Q$ are equal iff their coordinates are equal. For example, the origami obtained from the initial origami ABCD by folding along BD consists of the two congruent triangles, but after the fold, the polygon ABCD is no longer a convex rectangle since the points A and C are equal.

We also use the notion of being to the right or left based on the notion of handedness. It is given precise meaning when we fix the three-dimensional frame of reference to be either right-handed or left-handed. In this paper and the implementation of Eos, we take the $z$-axis to be perpendicular to the Euclidean ($xy$-) plane of our discourse. For $x$-, $y$- and $z$-axes of the reference frame, we adopt the right-handedness. Thus, we can say that a half-plane (of Euclidean plane) divided by line extending the segment $XY$ is to the right (resp. left) of line $XY$ if it is to the right (resp. left) of corresponding (location) vector $\overrightarrow{XY}$. A point is to the right (resp. left) of line $XY$ if it is on the right (resp. left) half-plane of the line $XY$. A face is to the right (resp. left) of line $XY$ if none of the vertices of the face is to the left (resp. right) of the line $XY$.

Abusing the set notation, we use $X \in m$ to mean that point $X$ is incident to line $m$, and $\{X_1, \ldots, X_k\} \subset m$ to mean that all the points $X_1, \ldots, X_k$ are incident to line $m$. A sequence of variables $x_1, x_2, \ldots$ may be denoted by $\underline{x}$. Likewise, a sequence of geometrical objects is denoted by $\underline{o}$.

We use a small programming language to describe the construction and verification of the polygonal knot fold. It is based on the language of Mathematica 9 (Wolfram Research, Inc., 2012) with the following differences. We use round brackets for grouping arguments of functions in order to be compatible with mathematical convention. Thus, function call is written as

$$f(a_1, \ldots, a_n, key_1 \to a_{n+1}, \ldots, key_m \to a_{m-n}), \text{ where } m \geq n \geq 0.$$

Since most of the functions used in Eos have many arguments whose default values are assigned beforehand, we use two kinds of arguments, i.e. position-dependent required arguments and position-independent optional arguments. The former are usual arguments and need no further explanation. The latter are used in the form of "keyword → argument", where the keyword specifies the name of the argument, and are used to reset the default value associated with the name. The functionality of the optional arguments comes from the language of Mathematica. Our language is a language of many-sorted

predicate logic with functions drawn from the rich set of Mathematica functions. The equality (infix) predicate symbol of our language is $=$, and is restricted to the sorts Point, Line and Number. The prefix predicate symbols for them are PointEqual, LineEqual and Equal, respectively.

### *2.2. Huzita's basic fold operations*

We consider the 7 operations proposed by Huzita as the set of basic fold operations (Huzita, 1989). They are implemented as the basic functions in Eos. The set of operations is similar in its role to the five Euclid's postulates in Euclidean Elements.

Huzita observed that the degree of freedom of the paper fold by fold lines can be limited by specifying combination of certain points and lines that are to be superposed. Then, he presented the following operations (O1) $\sim$ (O6), which serve as the basic operations in geometrical constructions of origamis.

(O1) Given two distinct points $P$ and $Q$, fold $\mathcal{O}$ along the unique line that passes through $P$ and $Q$.

(O2) Given two distinct points $P$ and $Q$, fold $\mathcal{O}$ along the unique line to superpose $P$ and $Q$.

(O3) Given two distinct lines $m$ and $n$, fold $\mathcal{O}$ along a line to superpose $m$ and $n$.

(O4) Given a line $m$ and a point $P$, fold $\mathcal{O}$ along the unique line passing through $P$ to superpose $m$ onto itself.

(O5) Given a line $m$, a point $P$ not on $m$ and a point $Q$, fold $\mathcal{O}$ along a line passing through $Q$ to superpose $P$ and $m$.

(O6) Given two 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, fold $\mathcal{O}$ along a line to superpose $P$ and $m$, and $Q$ and $n$.

The notion of superposition may be difficult to grasp when a line $m$ is moved to superpose on a non-moved point $P$. We need to move the half line of $m$ to superpose $m$ and $P$, unless the fold line and the line $m$ are parallel. For the examples, readers are referred to the paper by Ida et al. (2011).

We define the predicate $\mathrm{O5}(P, m, Q, x)$ to state the relationship among the points $P$ and $Q$, and the lines $m$ and $x$, which are involved in Huzita's basic fold (O5). It states that the fold line $x$ passing through point $Q$ superposes point $P$ and line $m$. Note that O5 is a relation, and is not defined as a function $\mathrm{O5}(P, m, Q)$ that returns a fold line $x$, since $x$ may not be unique. There may be 0, 1 and 2 possible fold lines depending on $P$, $m$, and $Q$. For the other basic folds we could define associated relations (or functions for (O1), (O2) and (O4)) in a similar way, but they are not necessary in this paper. Further discussions on the adequacy of operations (O1) $\sim$ (O6) as the set of basic fold operations are shown in (Ghourabi et al., 2013b), for example.

The above statements of Huzita's basic fold operations are about the fold lines, and do not address to the issues of the directions of the fold, i.e. mountain or valley fold, or of the choices of the faces to be moved. When we perform a fold in Eos, we need to specify them unless we rely on the their default values of Eos, i.e. valley fold and the choice of all the faces to the left of the fold line (interpreted as a vector).

## 3. Knot fold

We depicted a knotting operation in Fig. 1. In the figure we made one simple knot. When the height of the tape is infinitesimal and both ends of the tape are connected, the tape can be abstracted as a closed curve, i.e. an object of study in the theory of knots. The knot with 3 crossings is the most basic. It is denoted $3_1$ in the Alexander-Briggs notation (Alexander and Briggs, 1926). When the height is finite, each crossing exhibits a certain polygonal shape. The collection of the crossings, some of which overlap each other, is called a *polygonal knot*. The folds by which we construct the polygonal knot from the tape are called *knot folds*. The pentagonal knot shown in Fig. 2 corresponds to $3_1$.

We are now ready to construct a pentagonal knot. We will first follow the geometrical proof of a regular pentagonal knot of Sakaguchi (1982), but we focus on the formalization of computer-assisted construction and verification of the pentagonal knot.

One of the most fundamental properties of the crossing of the knot fold is expressed in the following lemma.

**Lemma 1** (Isosceles crossing)**.** *For all initial origami ABCD, point $E \in AB$ and point $F \in CD$, for all point $G \in CD$ such that O5(G, EA, E, EF), $\triangle GFE$ is isosceles with $|FG| = |EG|$.*

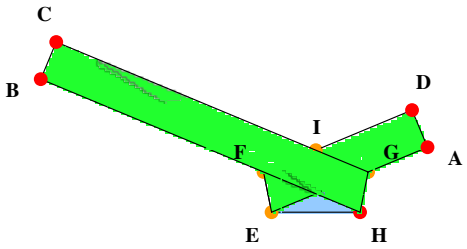
The lemma is written in a somewhat contrived style for reasons explained shortly. The meaning should be clear when it is presented together with the illustrative figure (cf. Fig. 3b). In brief, we try to perform a fold (O5), i.e. to fold the initial origami along fold line EF (passing through point E) to superpose point G and line EA. Point F is the intersection of the fold line and line CD. The polygon connecting points G, F and E is an isosceles triangle. We could argue that this fold is simply the fold (O1) along line EF. For the sake of the symmetry to be observed in Lemma 2, we use the fold (O5) here. With the tapes shown in Figs. 3a and 3b, the proof is straightforward by elementary geometrical reasoning, and hence is omitted.


(a) Initial origami configuration (b) Crossing of isosceles triangle △GFE

Fig. 3. Isosceles crossing

After the fold by which we constructed the isosceles triangle △GFE, we perform another fold to construct another isosceles triangle △FHG in such a way that both isosceles triangles overlap, sharing the segment FG as shown in Fig. 4. Lemma 2 expresses the construction involved and the resulting geometrical property.

**Lemma 2** (Isosceles trapezoid)**.** *For all origami ABCD, point $E \in AB$, point $F \in CD$ and point $G \in CD$ such that O5(G, EA, E, EF), for all point $H \in EB$ and point I such*




The folds along lines EF and HG construct trapezoid FEHG. Point I is the intersection of lines $D^{EF}F$ and $C^{GH}G$. Note that the point labeled as D is the reflection of initial point D across line EF, and that the point labeled as C is the reflection of initial point C across line GH. Triangles $\triangle$GFE and $\triangle$FHG are isosceles and congruent.

Fig. 4. Tape after two folds

*that O5(F, EB, G, GH)* $\wedge$ $I \in D^{EF}F$ $\wedge$ $I \in C^{GH}G$, *we have* $|EF| = |GH|$, *and furthermore* $|IG| = |IF|$.

Note that the expression $D^{EF}F$ is parsed as $(D^{EF})$ F, which denotes the line passing through point F and the reflection of point D across line EF, as explained in Subsect. 2.1.

Before we go into details of the constructions and proofs, it is worthwhile to note, at this point, that the lemmas we treat by the Eos prover have the following structure: For all $o_1, \ldots, o_i$ such that $\mathcal{P}(o_1, \ldots, o_i)$, for all $o_{i+1}, \ldots, o_k$ such that $\mathcal{Q}(o_1, \ldots, o_k)$, some geometrical properties hold among the objects $o_1, \ldots, o_k$, where

(i) "For all $o_1, \ldots, o_i$ such that $\mathcal{P}(o_1, \ldots, o_i)$" describes the configuration before the construction.
(ii) "for all $o_{i+1}, \ldots, o_k$ such that $\mathcal{Q}(o_1, \ldots, o_k)$" describes the step of the construction
(iii) "*some geometrical properties hold among the objects* $o_1, \ldots, o_k$" is what we claim by the construction.

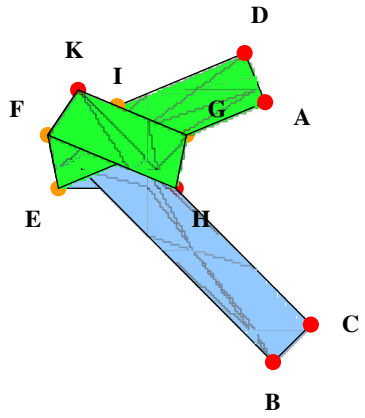
Clause (iii) is called the *goal*, and clauses (i) and (ii) together are called the *premise* of the proposition. With Eos we construct a geometrical object that we are interested in by some objects $o'_1, \ldots, o'_k$ that satisfy $\mathcal{P}(o'_1, \ldots, o'_k)$. After the successful construction, we try to prove a proposition of the above structure. The clause (ii) is true for some objects including $o'_1, \ldots, o'_k$. Therefore, the proposition is not vacuous.

Now let us return to Lemma 2. An intuitive proof of the property that the polygon FEHG is an isosceles trapezoid would be by two applications of Lemma 1. However, this is not completely satisfactory. If point I were not introduced, Lemma 2 would cover one more configuration besides the one shown in Fig. 4. Figures 5a and 5b show the other configuration. We would need to show the goal |EF| = |GH| holds for this case, too. We see that this is impossible by observing Fig. 5a. To eliminate this undesirable configuration, we introduce point I, which is not constructible for the undesirable case. In this case, lines $D^{EF}F$ and $C^{GH}G$ are parallel. Point I will be needed later for Theorem 3 as well.


(a) Before the folds (b) After the folds

Fig. 5. Undesirable case : tapes before and after the folds along EF and GH

We will postpone the description of our proof until Section 5 since the proof scheme is the same in other constructions. We move on to the construction of a pentagonal knot. On the tape shown in Fig. 4, we perform a fold (O5) along a fold line that passes though F, to superpose H and CG. We have two fold lines, say $m_1$ and $m_2$ to make this possible. The fold along $m_1$ creates the shape as shown in Fig. 6. Point K is created at the intersection of $m_1$ and line CG (in Fig. 4). The other case of the fold (along $m_2$) constructs the shape shown in Fig. 7b, where point K is not constructible.




Operation (O5) is applied to fold along the line passing through F and superposing H and CG. Point K is the intersection of line CG and the fold line. This construction shows the case where point K is constructible.

Fig. 6. A loose pentagonal knot EHGKF


(a) The other fold line

(b) Origami after the fold

Fig. 7. The other case of fold by (O5)

We now see how to fold the desired knot. The following shows the Eos program to construct the pentagon EHGKF.

**Program P1** [construction of a pentagonal knot]

1. BeginOrigami("Pentagonal knot",{150,10});
2. NewPoint({E $\rightarrow \{45, 0\}$, F $\rightarrow \{43, 10\}$});
3. HO($\exists_{m,m:\mathrm{Line}} \exists_{n,n:\mathrm{Line}} \exists_{g,g:\mathrm{Point}} \exists_{h,h:\mathrm{Point}} \exists_{i,i:\mathrm{Point}}$(O5($g$, EA, E, $m$) $\wedge$ $m =$ EF $\wedge$ O5(F, EB, $g$, $n$) $\wedge$ $n = g\,h$ $\wedge$ $g \in$ CD $\wedge$ $h \in$ AB $\wedge i \in \overline{\mathrm{D}^m\mathrm{F}}$ $\wedge$ $i \in \overline{\mathrm{C}^n g}$), MarkPointAt $\rightarrow$ {G,H,I}, MarkPointOn $\rightarrow$ {{CG, K}}, Handle $\rightarrow$ {A,B});
4. HO(H, CG, F, Handle $\rightarrow$ B, Direction $\rightarrow$ Mountain, InsertFace $\rightarrow$ Bottom, Case $\rightarrow$ 1);

Each line of the program is a function call, i.e. the calls of BeginOrigami, NewPoint, and two HO (which stands for Huzita Ori). Obeying the program, the construction proceeds as follows:

1. At Step 1 of the program, we start a new session of origami with the session name (i.e. "Pentagonal knot") and with the origami size (i.e $150 \times 10$).
2. At Step 2, we create new points E and F at the locations $(45, 0)$ and $(43, 10)$. The locations can be arbitrary subjected to the constraints that they are on line AB and line CD, respectively, and that they are located to enable us to construct an intended knot at the end.
3. At Step 3, we apply the geometrical construction described in Lemma 2. By solving the constraint, Eos returns two fold lines and three points. We perform two (O5) folds, and mark the three points as G, H and I, which correspond to the values of variables $g$, $h$ and $i$, respectively.
4. Finally at Step 4, we perform a fold (O5) along the fold line passing through F to superpose H and CG (cf. Fig. 6). The intersection of CG and the fold line is marked as K, using keyword parameter MarkPointOn. The operation (O5) is applied on points F, H, C and G of the previous step. To make a knot, we have to perform a mountain fold and insert the moving face of the tape in between the existing non-moving faces of the tape, i.e., in this case, immediately above the bottom face. The

direction of the fold, the handle of the movement and the insertion of the faces are specified as the keyword arguments in the program. By folding an origami $(\mathsf{O}, \backsim, \succ)$, a new origami $(\mathsf{O}', \backsim', \succ')$ is constructed. The valley/mountain fold computes the relations $\backsim'$ and $\succ'$ by the graph-theoretical method (Ida and Takahashi, 2010). When the insertion of a face is involved, we need to modify the relation $\succ'$.

Below, we have further note to the specifics of the Eos program syntax:

- The program is a version obtained from "copy as LaTeX" command of Mathematica 9 and modified for type-setting in Latex, and the actual running program does not look exactly the same. In programming practice, we usually use a restricted set of fonts and alphabet for ease of inputting texts and interpretation by the evaluator. For instance we use double quoted alpha-numeric character sequences to represent points and segments. They are printed without the double-quotes.
- On line 3, we use an existential formula of prenex normal form $\exists_{x_1:\sigma_1 \cdots x_k:\sigma_k} \mathcal{F}$, where $\mathcal{F}$ is a quantifier-free formula. Symbols $\sigma_1, \ldots, \sigma_k$ are the sorts of variables $x_1, \ldots, x_k$, respectively. The variables of sort Line denote fold lines to be used in this HO call. The variables of sort Point are usually the points of the intersection among existing segments and the new fold lines, as the result of the folds by this call. The formula specifies the geometrical relations among the variables.
- The meaning of the keyword arguments is as follows:
  - "Handle $\rightarrow P$" determines the side to be moved (left or right side of the fold line) by specifying one of the faces to be moved. Point $P$ designates that face, when it is on the to-be-moved face.
  - "Case $\rightarrow n$" is the important parameter of this construction. It says that the system choose $n$-th fold line. When this parameter specification is not present and there are more than one possible fold lines, Eos will display possible fold lines and ask the user to choose one among them. "MarkPointOn $\rightarrow \{\{s_1, P_1\}, \ldots, \{s_k, P_k\}\}$" specifies names $P_1, \ldots, P_k$ of the intersections of the segments $s_1, \ldots, s_k$ and the fold line, respectively (when there is only one fold line).
  - "MarkPointAt $\rightarrow \{P_1, \ldots, P_k\}$" specifies names for corresponding existential $k$ variables introduced in the existential formula. By this way, the points bound to the existential variables become available in the following steps of the construction.
  - "Direction $\rightarrow$ Mountain" specifies that the fold is a mountain fold. The default value is Valley. The specified value may be a list of Mountain/Vallley values, since a single call of HO may result in multiple folds, and for each fold the system needs Direction information.
  - "InsertFace $\rightarrow$ *list-of-faces*" is to insert the moving faces above (below in the case of the valley fold) the face in the list. The argument may be a list of faces for the same reason of Direction. Symbol 0 in the list means "no effect".

The pentagon EHGKF is not regular in general as we take E and F arbitrarily on each horizontal edge of the origami. However, we have the following theorem.

**Theorem 3** (Regular pentagon knot)**.** *Referring to Fig. 6, if points I and K coincide, i.e. are equal, the polygon EHGKF is a regular pentagon.*

The proof of the theorem by Eos will be discussed in Section 5.

To satisfy the condition K = I, we have to pull the both ends AD and BC of the origami outwards in Fig. 6, so that the knot becomes tight. This fastening operation is none of Huzita's basic fold operations. A question arises naturally whether we can make the knot using the Huzita's set of basic operations. This is one of the topics of the next section.

## 4. Analysis of the construction

### *4.1. Classical construction*

Let us look at the initial origami with points E and F on it in Fig. 3a again. We may fix the point E on the edge AB. The location of the point F on the edge CD relative to the location of the point E completely determines the subsequent shapes if we obey the same method of the folds discussed in the previous section. Suppose that we slide the point F along the edge CD towards D of the initial origami. This small move propagates in the subsequent three steps of the folds and brings the points K and I closer. The fastening of the knot amounts to those four operations i.e., sliding F and subsequent three folds. It remains to the hands of readers and origamists whether this fastening may be regarded as a single origami step or not.

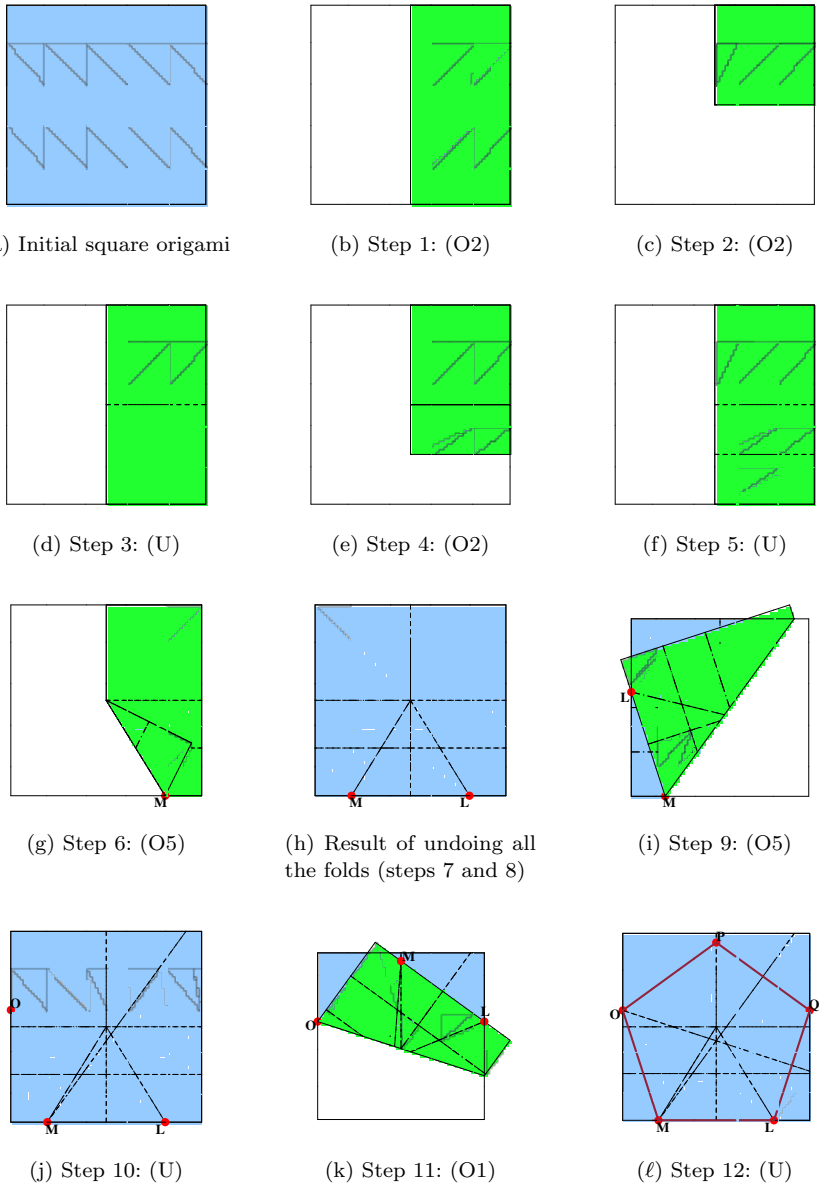
We next turn to the algebraic aspects of the knot fold of this pentagon construction. Since EF becomes one of the edges of the constructed pentagon, the slope of the line EF is $-\tan(2\pi/5) = -\sqrt{5+2\sqrt{5}}$ when the pentagon is regular. The line with this slope is constructible by Huzita's basic fold operations, starting from a square origami in Fig. 8a. The origami shown in Fig. 8$\ell$ is the result of the construction of a regular pentagon from the square origami [1]. It requires 6 steps of Huzita's basic folds and 6 steps of unfolds ((O2), (O2), (U), (O2), (U), (O5), (U), (U), (O5), (U), (O1), (U)) in this order, where (U) stands for unfold). The results of fold and unfold steps are shown in Fig. 8. For clarity, we only mark the points M, L, Q, P and O. The dotted lines are the creases made during the fold/unfold steps. The shape of regular pentagon MLQPO drawn in the red solid lines in Fig. 8$\ell$ is added 'manually', although it can be constructed by subsequent folds. Namely, the edges LQ, QP, PO and OM of the pentagon can be constructed by five (O1) folds. We identify the points O and M in Fig. 8$\ell$ with the points F and E (respectively) of the knot fold construction in Figs. 3, 4 and 6. The first 9 steps will generate the points O and M (see Figs. 8a $\sim$ 8i), and therefore these can be the preliminary steps to the knot folds that we discussed in the previous section.

However, this method obviously suffers from the following drawbacks. It not only requires preparatory steps but also makes the height of the tape shorter and lacks the simplicity of the original knot fold.

### *4.2. Constraint solving fold*

A more elegant construction with Eos is to utilize its constraint solving capability. To do so, we first unfold the knot entirely and examine the locations of the points that have been constructed. When the points K and I coincide, we obtain the tape with the fold lines and the points as shown in Fig. 9.

[1] It is unclear who discovered the fold method of the construction. Since the construction of a pentagon is possible by straightedge-and-compass, it may well be known since the Euclidean day. The method we used is described in the book (Geretschläger, 2002).




(a) Initial square origami (b) Step 1: (O2) (c) Step 2: (O2)

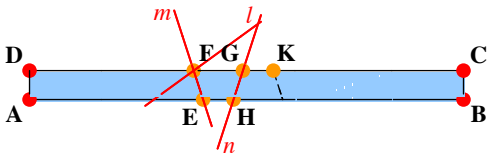
(d) Step 3: (U) (e) Step 4: (O2) (f) Step 5: (U)

(g) Step 6: (O5) (h) Result of undoing all the folds (steps 7 and 8) (i) Step 9: (O5)

(j) Step 10: (U) (k) Step 11: (O1) (ℓ) Step 12: (U)

Fig. 8. Construction of regular pentagon MLQPO by Huzita's basic fold operations

From Fig. 9 we can infer the following. Given an arbitrary but fixed point E, we can find out the locations of the crucial points on the origami relative to point E. Let $f$, $g$, $h$, and $k$ are the variables of sort Point that correspond to F, G, H and K in Fig. 9. They have a geometrical relation that can be expressed in a form of constraint. By solving the constraint for those variables, we can find the locations of points F, G, H and K. Let $m$, $n$ and $l$ be the fold lines used in Steps 3 and 4 of Program P1. Using Lemma 2, we only need the following conditions to find those fold lines:




From the unknotted regular pentagon, we can infer the relations on lines $m$, $n$, $l$ and points F, G, H and K.

Fig. 9. Regular pentagonal knot EHGKF unknotted

1. $h \in \mathrm{AB} \ \wedge \ \{f, g, k\} \subset \mathrm{CD} \ \wedge \ f \in m \ \wedge \ h \in n$ (incidence relations)
2. $\mathrm{O5}(g, \mathrm{EA}, \mathrm{E}, m) \ \wedge \ \mathrm{O5}(f, \mathrm{EB}, g, n) \ \wedge \ \mathrm{O5}\left(h, \overline{\mathrm{C}^n\, g}, f, l\right)$ (folds)
3. $k^n \in \overline{\mathrm{D}^m\, f} \ \wedge \ |\overline{k\, g}| = |\overline{k^n\, f}|$ (properties of $i(= k^n)$ and Lemma 2)
4. $|\overline{k\, g}| = |\overline{h\, \mathrm{E}}|$ (equality of edge length)

Note that some points on the origami are moved by the fold, and that the points given above are located on the initial origami. This is in contrast to the situations in Program P1, where in each call of HO, we need to think of the configuration immediately before and after the calls.

The conditions excluding the clause 4 underspecify the construction of the regular pentagonal knot. In that case we need to give a specific point F. This is what we had in the construction of the loose pentagonal knot.

In Eos language, the construction is stated in the following three function calls.

**Program P2** [construction of a regular pentagonal knot]

BeginOrigami("Regular pentagonal knot",{150,10});
NewPoint({E → {60, 0}});
HO( $\exists_{m,m:\mathrm{Line}} \exists_{n,n:\mathrm{Line}} \exists_{l,l:\mathrm{Line}} \exists_{f,f:\mathrm{Point}} \exists_{g,g:\mathrm{Point}} \exists_{h,h:\mathrm{Point}} \exists_{k,k:\mathrm{Point}}$
$(h \in \mathrm{AB} \ \wedge \ \{f, g, k\} \subset \mathrm{CD} \ \wedge \ f \in m \ \wedge \ h \in n \ \wedge$
$\mathrm{O5}(g, \mathrm{EA}, \mathrm{E}, m) \ \wedge \ \mathrm{O5}(f, \mathrm{EB}, g, n) \ \wedge \ \mathrm{O5}(h, \overline{\mathrm{C}^n\, g}, f, l) \ \wedge$
$k^n \in \overline{\mathrm{D}^m\, f} \ \wedge \mathrm{SquaredDistance}(k, g) = \mathrm{SquaredDistance}(k^n, f) \ \wedge$
$k - g = h - \mathrm{E})$,
Handle → {A,B,B},
MarkPointAt → {F, G, H, K},
InsertFace → {0, 0, Bottom},
Direction → {Valley, Valley, Mountain},
Case→ 7);

We have 8 possibilities of the folds. Only the 7th case makes the regular pentagon.

Note that the −(minus) operator is extended in Eos language. Point$(x_1, y_1)$ - Point$(x_2, y_2)$ is evaluated to be Point$(x_1 - x_2, y_1 - y_2)$.

### *4.3. Fold by algebraic constraint solving*

We know by now that the slope of line EF is critical in determining the locations of the points F, G, H, and K. So, we include the algebraic relations that determine the slope in the constraint. Replacing the HO call of the third line of Program P2 by the following

will do the task. We construct a perpendicular FX to line AB, whose foot X is the on AB. Let us consider △EFX. Let |FX|, |XE| and |EF| be $t$, $t \times x$, and $t \times y$, respectively. Recalling that $1/x = \tan(2\pi/5) = \sqrt{5+2\sqrt{5}}$, we see that $x$ and $y$ satisfy the equations $\{5x^4 - 10x^2 + 1 = 0,\ x^2 + 1 = y^2\}$.

$\mathrm{HO}(\exists_{m,m:\mathrm{Line}}\exists_{n,n:\mathrm{Line}}\exists_{l,l:\mathrm{Line}}\exists_{x,x:\mathrm{Num}}\exists_{y,y:\mathrm{Num}}\exists_{f,f:\mathrm{Point}}\exists_{g,g:\mathrm{Point}}\exists_{h,h:\mathrm{Point}}\exists_{k,k:\mathrm{Point}}$
$(\{\mathrm{E}, f\} \subset m \ \wedge \{g, h\} \subset n \ \wedge \{f, k^n\} \subset l \wedge$
$f - \mathrm{E} = \mathrm{Point}((-x)t, t) \wedge$
$\mathrm{E} - h = \mathrm{Point}((-y)t, 0) \wedge h - g = \mathrm{Point}((-x)t, -t) \ \wedge g - k = \mathrm{Point}((-y)t, 0) \wedge$
$5x^4 - 10x^2 + 1 = 0 \ \wedge x^2 + 1 = y^2)$,
Handle → {A, B, B},
MarkPointAt → {F, G, H, K},
InsertFace → {0, 0, Bottom},
Direction→ {Valley, Valley, Mountain}, Case→ 8);

As noted in Program P2, the cases in selecting the construction of interest are internally decided by the algorithm of the enumeration of the solutions. Only by interactive selection, we can determine the right construction. In this example, Case 8 constructs the regular pentagonal knot.

## 5. Computer assisted correctness proof

In this section we explain the proof method of Eos, and its application to the verifications of the constructions including Lemmas 1, 2 and 3 of the previous sections. The method of earlier versions of Eos with concrete examples was discussed in (Ida et al., 2011) and (Ghourabi et al., 2013a). The propositions that we want to prove are of the following form:

$$\forall \underline{x} \forall \underline{y} (\mathcal{P}(\underline{x}, \underline{y}) \implies \mathcal{Q}(\underline{x}, \underline{y})). \tag{1}$$

To construct a desired object, we choose some specific values for $\underline{x}$, say $\underline{a_0}$, and prove

$$\exists \underline{y} \mathcal{P}(\underline{a_0}, \underline{y}). \tag{2}$$

This is done by transforming the proposition (2) to the following algebraic form

$$\exists \underline{y'} (e_1(\underline{a_0}, \underline{y'}) = 0 \wedge \cdots \wedge e_j(\underline{a_0}, \underline{y'}) = 0), \tag{3}$$

and solve for $\underline{y'}$. Let $\underline{b'}$ be the solutions of $\underline{y'}$. The desired object is a collection from the set $\{\underline{b'}\}$.

To prove the proposition (1), we take arbitrary but fixed $\underline{a}$ for $\underline{x}$, and to prove

$$\forall \underline{y} (\mathcal{P}(\underline{a}, \underline{y}) \implies \mathcal{Q}(\underline{a}, \underline{y})). \tag{4}$$

We prove (4) by contradiction. Assume

$$\exists \underline{y} (\mathcal{P}(\underline{a}, \underline{y}) \wedge \neg \mathcal{Q}(\underline{a}, \underline{y})). \tag{5}$$

We use the same algebraic transformation as the one from (2) to (3), on $\neg \mathcal{Q}(\underline{a}, \underline{y})$ in (5) and obtain

$$\exists \underline{y'} (e_{j+1}(\underline{a}, \underline{y'}) = 0 \wedge \cdots \wedge e_k(\underline{a}, \underline{y'}) = 0). \tag{6}$$

We then compute the Gröbner basis of

$$\{e_1(\underline{a}, \underline{y'}), \ldots, e_j(\underline{a}, \underline{y'}), e_{j+1}(\underline{a}, \underline{y'}), \ldots, e_k(\underline{a}, \underline{y'})\}, \tag{7}$$

in order to decide the solvability of

$$\exists \underline{y'}(e_1(\underline{a},\underline{y'}) = 0 \wedge \cdots \wedge e_j(\underline{a},\underline{y'}) = 0 \wedge e_{j+1}(\underline{a},\underline{y'}) = 0 \wedge \cdots \wedge e_k(\underline{a},\underline{y'}) = 0). \quad (8)$$

If the reduced Gröbner basis of (7) is $\{1\}$ then the proposition (8) is false (unsolvable). Hence, the proposition (1) is proved. If the proposition (8) is true (solvable), we need to check whether none of the solutions will deliver points and lines in $\mathbb{R}^2$-plane. If this is the case, the proposition (1) is proved. Otherwise it is false. Eos checks whether the reduced Gröbner basis is $\{1\}$ or not. When the reduced Gröbner basis is $\{1\}$, Eos answers that the proposition (1) has been proved.

Now we apply the proof scheme to the proof of Lemma 1. The construction is implicit in the structure of the statement of the lemma. The property that we want to verify is $|\mathrm{FG}| = |\mathrm{EG}|$. Since squaring both sides of the above equation yields the equality in polynomial form, we specify the goal by the following call:

$$\mathrm{Goal}(\mathrm{SquaredDistance}(\mathrm{F},\mathrm{G}) = \mathrm{SquaredDistance}(\mathrm{E},\mathrm{G})).$$

Finally, to prove the lemma, we call function Prove:

$$\mathrm{Prove}(\text{“Isosceles crossing”}, \mathrm{Mapping} \rightarrow \{\mathrm{A} \rightarrow \{w1, 0\}, \mathrm{B} \rightarrow \{w2, 0\}, \mathrm{C} \rightarrow \{w2, 1\}, \mathrm{D} \rightarrow \{w1, 1\}, \mathrm{E} \rightarrow \{0, 0\}, \mathrm{F} \rightarrow \{u, 1\}\}) \quad (9)$$

The first parameter of the function call of Prove is the label naming the proposition to be proved, and the second parameter is a list of the initial point mapping. Without loss of generality, we let the height of the initial origami to be 1. The variables $w1$, $w2$ and $u$ in the mapping are arbitrary variables, so-called independent variables. This mapping is used in fixing $\underline{a}$ in (4). Lemma 2 can be proved in the same way.

The call of function Prove triggers the logical and algebraic transformations of the formulas accumulated during the construction and goal specification, and the computation of Gröbner basis. The Gröbner basis computation is carried out using GroebnerBasis function built into Mathematica 9. We specify the monomial order to be DegreeReverseLexicographic, and the coefficient domain to be RationalFunctions if independent variables are involved, otherwise $\mathbb{C}$. Function Prove completes with the message (on the Windows 7 machine with Intel Core i5 CPU M560 2.67 GHz with 4 GB memory) for proving Lemma 2:

```
Groebner basis computation started at 2014/02/02 18:38:18 JST.
Proof by Groebner basis method is successful.
CPU time used for Groebner basis computation is 0.031200 seconds.
```

together with the proof certificate called ProofDoc (Ghourabi et al., 2011).

Next we consider the pentagonal knot construction by the constraint solving fold discussed in SubSect. 4.2. The goal that goes with Program P2 for the construction of the regular pentagon is as follows:

$$\begin{aligned}&\mathrm{Goal}(\forall_{\alpha,\alpha:\mathrm{Complexes}}\\ &\quad(\alpha\ \mathrm{ToZ}(\overrightarrow{\mathrm{EH}}) = \mathrm{ToZ}\ (\overrightarrow{\mathrm{HG}}) \Rightarrow\\ &\quad\alpha\ \mathrm{ToZ}(\overrightarrow{\mathrm{HG}}) = \mathrm{ToZ}(\overrightarrow{\mathrm{GK}}) \wedge \alpha\ \mathrm{ToZ}(\overrightarrow{\mathrm{GK}}) = \mathrm{ToZ}(\overrightarrow{\mathrm{KF}}) \wedge\\ &\quad\alpha\ \mathrm{ToZ}(\overrightarrow{\mathrm{KF}}) = \mathrm{ToZ}(\overrightarrow{\mathrm{FE}}) \wedge \alpha^5 = 1)];\end{aligned}$$

The domain of complex numbers, $\mathbb{C}$-plane, is used in order to show the equalities of the inner angles for each vertex and of the length of the five edges, together. To make a one-to-one correspondence between the points on the $\mathbb{R}^2$-plane and those of the $\mathbb{C}$-plane, we define the function ToZ as follows.

$$\mathrm{ToZ}(X) = x + \imath y, \text{where } X = \mathrm{Point}(x, y)$$

The domain of function ToZ is extended naturally to vectors in the $\mathbb{R}^2$-plane by

$$\mathrm{ToZ}(\overrightarrow{PQ}) = \mathrm{ToZ}(Q) - \mathrm{ToZ}(P).$$

Then, $\alpha\mathrm{ToZ}(\overrightarrow{v})$, where $\alpha = r(\cos\theta + \imath\sin\theta), r \geq 0$ gives the element in the $\mathbb{C}$-plane that corresponds to the vector

$$\overrightarrow{v'} = r \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \overrightarrow{v}$$

The vector $\overrightarrow{v'}$ is the rotated $\overrightarrow{v}$ by angle $\theta$ (with the origin of the coordinate system of the $\mathbb{R}^2$-plane, as the center) and whose length is $r|\overrightarrow{v}|$. Thus, it is clear that the above goal statement completely specifies the required properties of the regular pentagon.

The following call

Prove("Knot pentagon theorem",
Mapping $\rightarrow \{\mathrm{A} \rightarrow \{w1, 0\}, \mathrm{B} \rightarrow \{w2, 0\}, \mathrm{C} \rightarrow \{w2, 1\}, \mathrm{D} \rightarrow \{w1, 1\},$
$\mathrm{E} \rightarrow \{0, 0\}\}$),

leads to the success of the proof. In the call, we take E(0,0) and $w1$ and $w2$ to be arbitrary real-valued variables for the initial point mapping, This shows that we have constructed a regular pentagonal knot. To be precise, we have proved the following Theorem 4. Let $\mathcal{P}(m, n, l, f, g, h, k, \alpha)$ and $\mathcal{Q}(m, n, l, f, g, h, k, \alpha)$ be the predicates defined as follows:

$\mathcal{P}(m, n, l, f, g, h, k, \alpha) \equiv$
$h \in \mathrm{AB} \ \wedge\ \{f, g, k\} \subset \mathrm{CD} \ \wedge\ f \in m \ \wedge\ h \in n \ \wedge$
$\mathrm{O5}(g, \mathrm{EA}, \mathrm{E}, m) \ \wedge\ \mathrm{O5}(f, \mathrm{EB}, g, n) \ \wedge\ \mathrm{O5}(h, \overline{\mathrm{C}^n\, g}, f, l) \ \wedge$
$k^n \in \overline{\mathrm{D}^m\, f} \ \wedge\ \mathrm{SquaredDistance}(k, g) = \mathrm{SquaredDistance}(k^n, f) \ \wedge\ k - g = h - \mathrm{E}$

$\mathcal{Q}(m, n, l, f, g, h, k, \alpha) \equiv$
$\alpha\ \mathrm{ToZ}(\overrightarrow{\mathrm{E}\, h}) = \mathrm{ToZ}(\overrightarrow{h\, g}) \Rightarrow$
$\alpha\ \mathrm{ToZ}(\overrightarrow{h\, g}) = \mathrm{ToZ}(\overrightarrow{g\, k^n}) \wedge \alpha\ \mathrm{ToZ}(\overrightarrow{g\, k^n}) = \mathrm{ToZ}(\overrightarrow{k^n\, f}) \wedge$
$\alpha\ \mathrm{ToZ}(\overrightarrow{k^n\, f}) = \mathrm{ToZ}(\overrightarrow{f\, \mathrm{E}}) \wedge \alpha^5 = 1;$

**Theorem 4.** *For all origami ABCD and all point E on AB, for all* $m, n, l, f, g, h, k, \alpha$ $\mathcal{P}(m, n, l, f, g, h, k, \alpha) \implies \mathcal{Q}(m, n, l, f, g, h, k, \alpha)$.

The same construction and proof scheme is used in the proof of Theorem 3. The premise $\mathcal{P}$ is computed by Eos from the record of the construction from the beginning to the end of the construction. What we have to do is to set up a proper goal. For the proof of the regular pentagonal knot we call:

Goal($\forall_{\alpha,\alpha:\mathrm{Complexes}}$
(PointEqual(I, K) $\Rightarrow \alpha\ \mathrm{ToZ}(\overrightarrow{\mathrm{EH}}) = \mathrm{ToZ}(\overrightarrow{\mathrm{HG}}) \Rightarrow$

$\alpha$ ToZ($\overrightarrow{\mathrm{HG}}$) = ToZ($\overrightarrow{\mathrm{GK}}$) $\wedge$ $\alpha$ ToZ($\overrightarrow{\mathrm{GK}}$)= ToZ($\overrightarrow{\mathrm{KF}}$) $\wedge$
$\alpha$ ToZ($\overrightarrow{\mathrm{KF}}$) = ToZ($\overrightarrow{\mathrm{FE}}$) $\wedge$ $\alpha^5 = 1$));

and then call function Prove (with possibly a theorem name "Regular Pentagonal Knot Theorem") as in (9).

## 6. Regular heptagonal knot

The construction and the verification of a regular heptagonal knot can be done similarly to the case of the pentagonal knot. First, we create arbitrary but fixed new points E and F on the initial origami. We can start the construction of a heptagonal knot by the application of Isosceles Trapezoid Lemma (Lemma 2). Then, we pick the point B to pull the faces on which B is on, and perform a mountain fold (O5), then pick A and perform a mountain fold (O5), and finally pick A and perform a valley fold (O5). The parameters of the folds (O5) are chosen to construct the congruent isosceles trapezoids. We need to pay attention to the keyword arguments Direction (Mountain or Valley) and InsertFace (above Bottom or below Top), to make the knot rigid.

Figure 10 shows the intermediary steps of the fold of heptagonal knot. We finally obtain the heptagon ELHGJKF shown in Fig. 11.

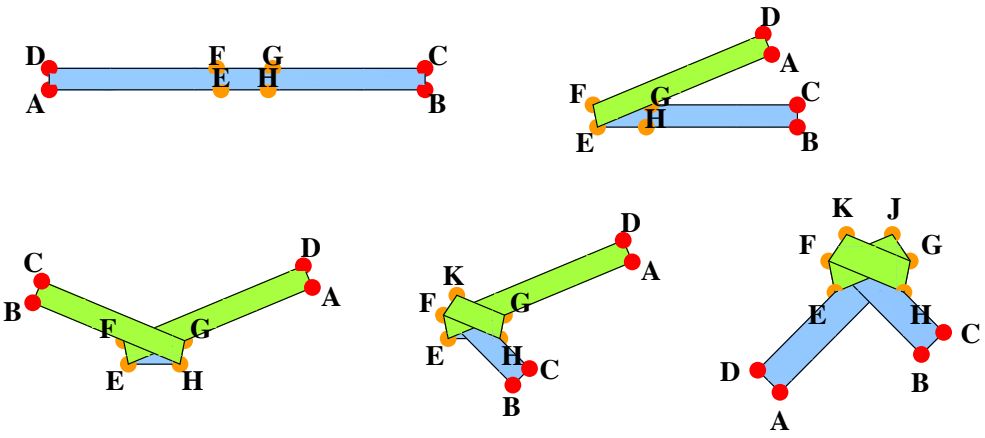



Fig. 10. Intermediary steps of heptagonal knot construction

Similarly to the case of the regular pentagonal knot, to make the polygon ELHGJKF a regular heptagon, we need to add further constraints of the equalities of the edges. We redo the construction after adding the constraints. With hands, we pull the edges AD and BC outwards, as we did with the regular pentagonal knot. This will move point F and perturb the whole shape. The final one is shown in Fig. 12.

As for the proof for the construction shown in Fig. 10, we give to the prover, before specifying the goal, the following assumptions:

$$\begin{aligned}\mathrm{SquaredDistance}(\mathrm{E},\mathrm{F}) &= \mathrm{SquaredDistance}(\mathrm{G},\mathrm{H}) = \mathrm{SquaredDistance}(\mathrm{F},\mathrm{K}) \\ &= \mathrm{SquaredDistance}(\mathrm{E},\mathrm{L}) = \mathrm{SquaredDistance}(\mathrm{J},\mathrm{G}));\end{aligned} \tag{10}$$

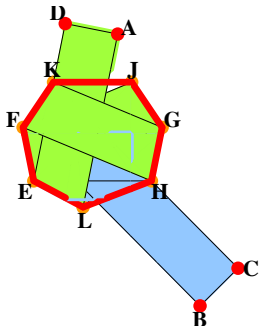



Fig. 11. Heptagon-like knot ELHGJKF

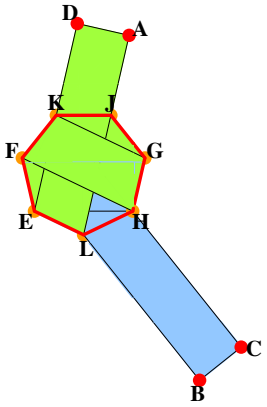



Fig. 12. Regular heptagonal knot

$$\text{SquaredDistance}(\text{E},\text{F}) = \text{SquaredDistance}(\text{K},\text{J}) = \text{SquaredDistance}(\text{L},\text{H}) \qquad (11)$$

The assumption (10) comes from Isosceles Trapezoid Lemma and the assumption (11) is the one ensured by the fastening of the tape. By both assumptions we intend to assume the equalities among the length of the edges involved. The assumption (10) should be unnecessary theoretically, since this is the consequence of Isosceles Trapezoid Lemma. However, in practice, without the assumption (10) the Gröbner basis computation during the proof by our stock hardware does not terminate within a reasonable amount of time. The goal to be proved is similar to the one for the proof of the regular pentagonal knot. We use the same proof scheme as the one for the proof of Theorem 4. As in the case of the regular pentagon, we do not have to check whether the vertices are concyclic, as our algebraic treatment ensures this property. The goal to be established is the statement that each connecting pair of the edges of the heptagon, considered as vectors, are only different by the rotation of angle $2\pi/7$.

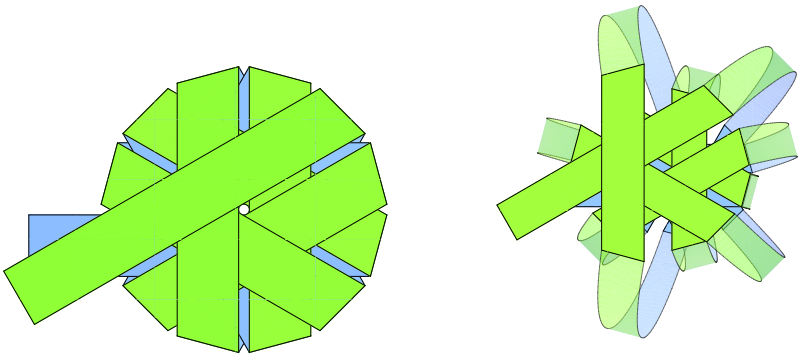
In (Robu et al., 2006), we presented the automated construction and verification of a regular heptagon using the Huzita's basic fold operations using the coordinating system of Theorema (Buchberger et al., 2000) and Eos. What is new, in this regard, in this paper is that we use a knot fold in a new version of Eos, which integrates the construction by logical specification and theorem proving by computer algebra engines.

## 7. Conclusion

We have analyzed the knot fold of polygons using the computer-assisted origami system. We showed that from mathematical and computer science point of view, it is a new kind of fold method. Our analysis relies heavily on algebraic methods, which has the advantage of checking all involved cases of geometrical construction and verification automatically. We mainly discussed the pentagonal knot since it is the most basic knot. As the number of edges of the polygons becomes larger, the time to take for the verification of the regularity becomes longer.

From the computational point of view, the interaction of symbolic computation and numeric computation becomes common in our study. Especially, the construction requires approximate numeric computation, whereas the verification is in symbolic and exact numeric computation mode. A method of Gröbner bases computation of polynomials with approximate numeric (floating point) coefficients such as developed by Sasaki (2012) will be of great help to our research.

We see several directions of the research based on our results and methodologies. We are aware that Brunton (1961) gave several interesting examples without proofs. With the use of Eos it will become a routine work to generate polygonal knots for these examples. Although theoretically not significant, the polygonal knots of even numbered edges (cf. Fig. 13) are interesting to observe. It is generated by the method of the algebraic constraint solving discussed in SubSect. 4.3. Maekawa (2011) gave visionary introductory accounts on knot folds. We believe that to establish origamists' ideas firmly on the ground of computer science and mathematics would be an important step to direct.



(a) Plane view (b) 3D view

Fig. 13. Knot of dodekagon-like shape

## References


Alexander, J. W., Briggs, G. B., 1926. On Types of Knotted Curves. Annals of Mathematics 28 (1/4), 562–586.

Alperin, R. C., Lang, R. J., 2009. One-, Two, and Multi-fold Origami Axioms. In: Origami$^4$, Proceedings of the Fourth International Meeting of Origami Science, Mathematics, and Education (4OSME). pp. 371–393.

Brunton, J. K., 1961. Polygonal Knots. The Mathematical Gazette 45 (354), 299–301.
Buchberger, B., Dupre, C., Jebelean, T., Kriftner, F., Nakagawa, K., Vasaru, D., Windsteiger, W., 2000. The Theorema Project: A Progress Report. In: Symbolic Computation and Automated Reasoning (Calculemus 2000). St Andrews, Scotland, pp. 98–113.
Geretschläger, R., 2002. Mathematics of Origami. Morikita Publishing Co., in Japanese, translation by Hidetoshi Fukagawa.
Ghourabi, F., Ida, T., Kasem, A., 2011. Proof Documents for Automated Origami Theorem Proving. In: Automated Deduction in Geometry. Vol. 6877 of LNCS. Springer, pp. 78–97.
Ghourabi, F., Ida, T., Takahashi, K., 2013a. Logical and Algebraic Views of a Knot Fold of a Regular Heptagon. In: Proceedings of the International Symposium on Symbolic Computation in Software Science (SCSS2013). Vol. 15 of EPiC Series. EasyChair, Hagenberg, Austria, pp. 50–63.
Ghourabi, F., Kasem, A., Kaliszyk, C., 2013b. Algebraic Analysis of Huzita's Origami Operations and their Extensions. In: Automated Deduction in Geometry. Vol. 7993 of LNCS. Springer Berlin Heidelberg, pp. 143–160.
Huzita, H., 1989. Axiomatic Development of Origami Geometry. In: Proceedings of the First International Meeting of Origami Science and Technology. pp. 143–158.
Ida, T., Ghourabi, F., Takahashi, K., 2014. Knot fold of regular polygons: computer-assisted construction and verification. In: The 15th International Symposium on Symbolic and Numerical Algorithms for Scientific Computing (SYNASC 2013). IEEE Computer Society, pp. 12–19.
Ida, T., Kasem, A., Ghourabi, F., Takahashi, H., 2011. Morley's Theorem Revisited: Origami Construction and Automated Proof. Journal of Symbolic Computation 46 (5), 571 – 583.
Ida, T., Takahashi, H., 2010. Origami Fold as Algebraic Graph Rewriting. Journal of Symbolic Computation 45, 393–413.
Ida, T., Takahashi, H., Marin, M., Ghourabi, F., Kasem, A., 2006. Computational Construction of a Maximal Equilateral Triangle Inscribed in an Origami. In: Mathematical Software - ICMS 2006. Vol. 4151 of Lecture Notes in Computer Science. Springer, pp. 361–372.
Maekawa, J., 2011. Introduction of the Study of Knot Tape. In: Origami$^5$, Proceedings of the Fourth International Meeting of Origami Science, Mathematics, and Education (5OSME). CRC Press, pp. 395–403.
Robu, J., Tepeneu, D., Ida, T., Takahashi, H., Buchberger, B., 2006. Computational Origami Construction of a Regular Heptagon with Automated Proof of its Correctness. In: Proceedings of ADG 2004, the 5th International Workshop on Automated Deduction in Geometry. Vol. 3763 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 19–33.
Sakaguchi, K., 1982. On Polygons Made by Knotting Slips of Paper. Technical Report of Research Institute of Education, Nara University of Education 18, 55–58.
Sasaki, T., 2012. A Theory and an Algorithm of Approximate Gröbner Bases. In: The 13th International Symposium on Symbolic and Numerical Algorithms for Scientific Computing (SYNASC 2013). IEEE Computer Society, pp. 23–30.
Wells, D., 1991. The penguin dictionary of curious and interesting geometry. Penguin Books Ltd.
Wolfram Research, Inc., 2012. Mathematica Edition: Version 9.0.0.
URL `http://www.wolfram.com/mathematica/`