

Graph Rewriting in Computational Origami

Tetsuo Ida and Hidekazu Takahashi*

Department of Computer Science, University of Tsukuba
Tsukuba 305-8573, Japan
{ida, hidekazu}@score.cs.tsukuba.ac.jp

Abstract. We formalize paper fold (origami) by graph rewriting. Origami construction is abstractly described by a rewrite system $(\mathcal{O}, \rightsquigarrow)$, where \mathcal{O} is the set of abstract origami's and \rightsquigarrow is a binary relation on \mathcal{O} , called *fold*. An abstract origami is a triplet (Π, \smile, \succ) , where Π is a set of faces constituting an origami, and \smile and \succ are binary relations on Π , each representing adjacency and superposition relations of the faces. Origami construction is modeled as a rewrite sequence of abstract origami's. We then address the problems of representation and transformation of abstract origami's and of reasoning about the construction for computational purposes. We present a hypergraph of origami and define origami fold as algebraic graph transformation. The algebraic graph theoretic formalism enables us to reason origami in two separate domains of discourse, i.e. pure combinatoric domain and geometric domain $\mathcal{R} \times \mathcal{R}$, and thus helps us to further tackle challenging problems in origami research.

1 Introduction

Origami provides the methodology of constructing a geometrical object with a piece of paper only by means of folding by hands. Computational origami studies the mathematical aspects of origami. By the assistance of a computer we will be able to formalize origami with rigor and capability that are beyond the methods performed by hands.

In this paper we give graph theoretic formalization of origami. Our main motivation of this study is to give more abstract view of origami fold. Although paper fold appears to be a simple operation to humans, an anatomy of origami reveals that it is not an easy operation. There are two distinct operations in paper fold, i.e. division and reflection of origami faces. These operations lend themselves to distinct modes of computations; algebraic and numeric computation on geometric objects, e.g. finding intersection of lines and checking the overlap of two faces, on one hand, and purely combinatoric computation on discrete objects, e.g. computing transitive closure of the adjacency relation on faces, on the other.

These computations tend to be mixed when origami is analyzed mathematically in our earlier work [3]. Indeed our current implementation of Eos [4] relies

* This research is supported by the JSPS Grants-in-Aid for Exploratory Research No. 19650001 and by Japan-Austria Research Cooperative Program of JSPS and FWF.

very much on algorithms which resort to mixtures of algebraic, numeric and symbolic computing. Sometimes algorithms are not easy to describe mathematically because of this intricacy. There should be clearer separation of computations of discrete and continuous objects in origami. By doing so, we not only clarify the algorithms developed for the implementation of Eos, but also to extend the capability of Eos to allow for more complex origami construction such as of 3D and modular origami.

The rest of the paper is organized as follows. In Section 2 we will formalize basic origami operations. In Section 3 we explain the bases for graph theoretic modeling of origami. In Section 4, we show how basic operations of origami are formalized in the algebraic and graph theoretic framework. In Section 5, we summarize the results and point out the direction of our research.

2 Formalizing origami

2.1 First glimpse of origami

We start an origami construction with a single piece of paper, and repeats folding of the paper until it becomes a desired shape. Here, we see that an origami can be modeled as a set of faces. During the construction, some of the faces get divided by a fold line, get rotated along the fold line and become above or below the others. The faces form layers. The layers of faces exhibit a remarkable shape, which may be regarded as a piece of art such as illustrated in Fig. 1.

The left origami in Fig. 1 is the top view of the constructed object. We see the faces in two different colors in the figure. This is because the initial origami has two sides, each colored differently. During the construction, some faces become up and the others become down, resulting in the two colored object. We can imagine that this origami models a cicada. The right is a 3D view of the same origami after stretching it vertically and making overlapping faces slightly far apart. From the shapes in Fig. 1, we will be able to observe that an origami can be formalized as a set of faces together with the relations that express relative positions, horizontally and vertically, among the faces.

2.2 Abstract origami

An origami can be modeled at several abstraction levels. A most abstract view is to take an origami as an algebra $\langle A, R \rangle$, where A is a set and R is a binary relation on A , where we identify A to be a set of faces that constitute an origami, and R to be a geometrical relation on the faces. The origami construction is then a transformation of the algebras viewed as an abstract rewrite system. We begin with this view of abstraction and gradually make our modeling concrete.

We consider a finite set Π of faces to be the object of our study, and introduce two binary relations on Π , expressing horizontal and vertical arrangements of faces rather than a single binary relation R mentioned before. Then, we have the following definition of an origami.

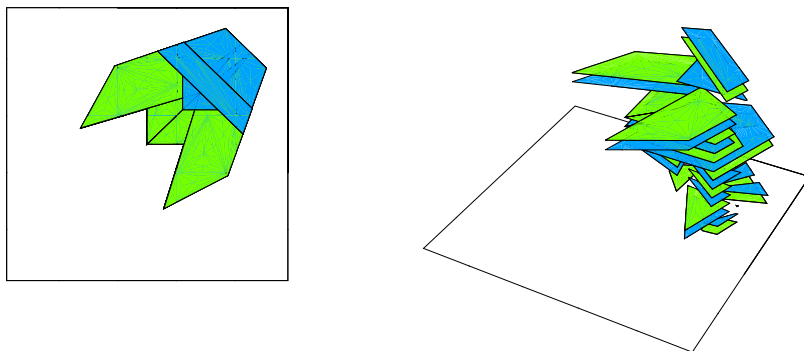


Fig. 1. Origami cicada: art piece (left) and face layers (right)

Definition 1 (Abstract origami). An abstract origami is a structure

$$\langle \Pi, \sim, \succ \rangle$$

where

- Π is the finite set of faces, and
- \sim and \succ are binary relations on Π , called adjacency and superposition relations, respectively.

From our observations so far it is clear that our modeling needs the relations of adjacency and superposition.

Definition 2 (Abstract origami system). An abstract origami system is an abstract rewrite system $(\mathcal{O}, \rightsquigarrow)$ where

- \mathcal{O} is the set of abstract origami's.
- \rightsquigarrow is the binary relation on \mathcal{O} called abstract fold, and is denoted by $O \rightsquigarrow O'$ for $O, O' \in \mathcal{O}$.

Origami construction proceeds stepwise. Namely, we start with an initial origami ($i = 0$) and perform folds along fold lines repeatedly until we obtain a desired shape. Suppose that we are at the beginning of step i of the construction, having an origami $O_{i-1} = (\Pi_{i-1}, \sim_{i-1}, \succ_{i-1})$. We make the next fold and obtain the next origami $O_i = (\Pi_i, \sim_i, \succ_i)$. Thus we have the following:

Definition 3 (Abstract origami construction). An abstract origami construction is a finite sequence of abstract folds:

$$O_0 \rightsquigarrow O_1 \rightsquigarrow \cdots \rightsquigarrow O_n, \quad \text{where } O_0, O_1, \dots, O_n \in \mathcal{O}$$

Although some properties of origami can be studied with necessary rigor at this level, more geometric information is needed to understand many of the properties of origami. We are lead to the definition of face in Def. 4.

Before we proceed, we note the following definition of an n -gon. An $n(n \geq 3)$ -gon is a polygon consisting of n edges none of which intersect each other. We also use the notion of overlapping. Let an expression p° denote the interior of an n -gon p . We identify the interior of an n -gon with the set of all the points in the interior. N -gons p and q are called *overlapping* if $p^\circ \cap q^\circ \neq \emptyset$.

Definition 4 (Face). *A face is a convex n -gon.*

Then we can define the adjacency relation as follows:

Definition 5 (Face adjacency). *Two faces are adjacent if they share an edge.*

We can determine whether a face is adjacent to the other face.

Concerning the superposition relation, we assume a decision procedure of determining above or below relation among the faces. Namely, given two faces f and g , we can determine whether:

- (1) f is above g or
- (2) g is above f or
- (3) f and g are not related.

We also say that g is below f if f is above g . Now we have the following definition of the superposition relation.

Definition 6 (Face superposition). *Face f superposes face g if f and g are overlapping, f is above g and no faces that are above g is below f .*

2.3 Formalization of fold

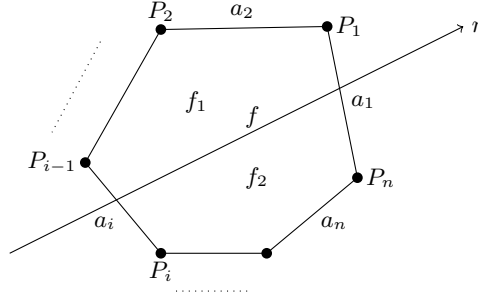
Fold of an origami is a complex operation consisting of the following sub-operations.

- (1) Specify the set \mathcal{F} of the faces of concern and decide a basic fold operation.
We may use one of Huzita's basic folds [2] or classical fold methods such as mountain and valley folds.
- (2) Compute a fold line l and make it a direct line called a ray r .
- (3) For each face f in \mathcal{F} , do the following until $\mathcal{F} = \emptyset$.
 - (a) Divide f by r into two faces f_1 and f_2 , where f_2 is to the right of r . See below for more details.
 - (b) Update \mathcal{F} by removing f from \mathcal{F} and adding faces that are affected by this division using the superposition and adjacency relations.
- (4) Compute new superposition and adjacency relations induced by the division.
- (5) Rotate the faces to the right of r along r .
- (6) Compute new superposition relation induced by the rotation.

As for the division of a face, we distinguish four cases of geometrical configurations:

1. The line l intersects with the edges of f at two distinct points. The face f is divided into two faces f_1 and f_2 , where f_2 is to be rotated.
2. The line l overlaps with some edge of f .
3. The line l passes through f at only one vertex.
4. The line l intersects with none of the edges of f .

Let us now make our modeling more concrete. We represent an n -gon as a sequence of points $\langle P_1, \dots, P_n \rangle$, where points P_1, \dots, P_n are vertices of the n -gon. A face is thus represented as a sequence of points. When points P_1, \dots, P_n are arranged counterclockwise, we say that the face is up, and when clockwise, it is down. The division of an up face is illustrated in Fig. 2. This case is further investigated for the graph transformation in the next section.



The face f is divided by the ray r into $\langle f_1, f_2 \rangle$. The face f_2 is to be rotated.

Fig. 2. Face division

3 Graph formalization

3.1 Hypergraph and graph term

To make origami amenable to computation, we further concretize the abstract origami by graph theoretic formalism. We use a labeled hypergraph for this purpose. Since we do not need algebraic graph theories in full generality such as discussed in [1], we work with hypergraphs defined as follows.

Definition 7 (Hypergraph). A hypergraph is a quadruple $\langle V, E, s, t \rangle$, where

- V is the set of nodes¹,
- E is the set of edges, and
- $s, t : E \rightarrow V^*$ are source and target functions.

¹ We use of the word *node* here to avoid the clashes with *vertices* of a polygon. We cannot avoid the clashes of the word *edge* of a polygon and of a graph, however.

Definition 8 (Hypergraph labeling). A hypergraph labeling consists of a pair $\langle \mathcal{L}_E, \mathcal{L}_V \rangle$ of label alphabets together with functions $\tau_s, \tau_t : \mathcal{L}_E \rightarrow \mathcal{L}_V^*$.

Definition 9 (Labeled hypergraph). Given a pair $\mathcal{L} = \langle \mathcal{L}_V, \mathcal{L}_E \rangle$ of label alphabets, an \mathcal{L} -labeled hypergraph is a 6-tuple $\langle V, E, s, t, l_V, l_E \rangle$, where

- $\langle V, E, s, t \rangle$ is a hypergraph and
- $l_V : V \rightarrow \mathcal{L}_V$ and $l_E : E \rightarrow \mathcal{L}_E$ are functions satisfying $\tau_s \cdot l_E = l_V^* \cdot s$ and $\tau_t \cdot l_E = l_V^* \cdot t$.

Hereafter, we only consider hypergraphs. Therefore we call hypergraph and hyperedges without prefix “hyper” unless we want to emphasize “hyper”.

Definition 10 (Graph term). Given an \mathcal{L} -labeled graph $G = \langle V, E, s, t, l_V, l_E \rangle$, graph term representation \mathcal{G} of a graph G is defined as

$$\{l_E(e)[s(e), t(e)] \mid e \in E\} \uplus \{l_V(v)[v] \mid v \in V\} \quad (3.1)$$

The expression $l_E(e)[s(e), t(e)]$ is of the form $F[v_1, \dots, v_n]$, which is actually the term representation in our language for graph transformation as we will see shortly. Note that $s(e), t(e)$ is a sequence of nodes.

In Eq.(3.1), $\{l_E(e)[s(e), t(e)] \mid e \in E\}$ is a multi-set and \uplus is the union operation on multi-sets. The element of $\{l_E(e)[s(e), t(e)] \mid e \in E\}$ is called *edge term* of the graph and the element of $\{l_V(v)[v] \mid v \in V\}$ is called *node term*. Both terms are called *graph terms*, *g-term* in short. This representation allows us to reason with tree structures even when we are dealing with graphs. Note that terms are representation of trees. A hypergraph is now represented as a set of g-terms. In most algebraic graph transformations to follow, our focus is more on the manipulation on edges than of nodes. The rewrite rules for the graph transformation are designed on the basis of the manipulation of edge terms.

Example 1. Let \mathcal{L}_V and \mathcal{L}_E be $\{F\}$ and $\{A, R, L\}$, respectively, and let G be an \mathcal{L} -labeled graph $\langle V, E, s, t, l_V, l_E \rangle$, where

- $V = \{f, f_1, f_2\}$
- $E = \{e_1, e_2, e_3, e_4\}$
- $s = \{e_1 \mapsto f, e_2 \mapsto f, e_3 \mapsto \langle f_1, a_1, \dots, a_i, f_2 \rangle, e_4 \mapsto \langle f_2, a_i, \dots, a_n, a_1, f_1 \rangle\}$
- $t = \{e_1 \mapsto f_1, e_2 \mapsto f_2, e_3 \mapsto \langle f_1 \rangle, e_4 \mapsto \langle f_2 \rangle\}$
- $l_E = \{e_1 \mapsto L, e_2 \mapsto R, e_3 \mapsto A, e_4 \mapsto A\}$
- $l_V = \{f_1 \mapsto F, f_2 \mapsto F, f \mapsto F\}$.

The hyperedge e_3 forms a loop visiting the nodes $f_1, a_1, \dots, a_i, f_2$ in this order. The hyperedge e_4 also forms a loop. A looped hyperedge visiting nodes v_1, \dots, v_n in this order may be described as a hyperedge visiting nodes $v_i, \dots, v_n, v_1, \dots, v_{i-1}$ for any $i \in \{1, \dots, n\}$, as well. We, however, use a unique representation of the hyperedge by imposing the condition that the first node is always a newly created node(face), as seen in this example.

The g-term representation \mathcal{G} of G is:

$$\mathcal{G} = \{ L[f, f_1], R[f, f_2], \\ A[f_1, a_1, \dots, a_i, f_2 \cdot f_1], A[f_2, a_i, \dots, a_n, a_1, f_1, f_2], \\ F[f], F[f_1], F[f_2] \}$$

The graph is shown in Fig. 3. In the graph the node v_i with label L is represented as a circled $v_i: L$. The edge e_i with label L is represented as a boxed $e_i: L$. This graph represents the face division given in Fig. 2.

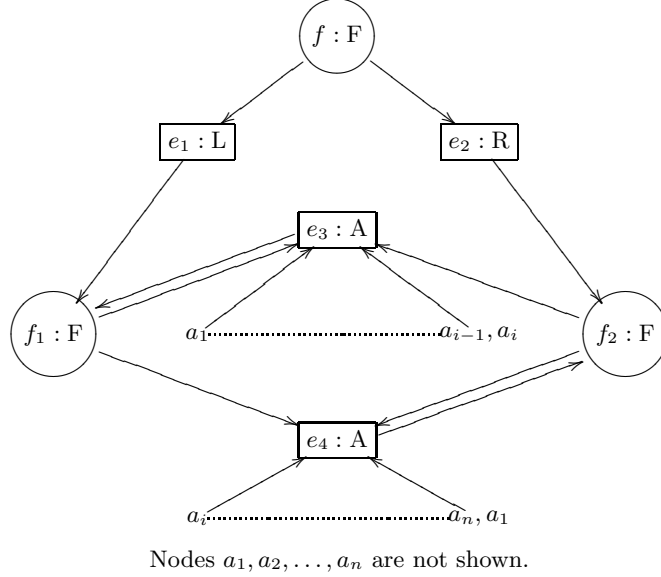


Fig. 3. Graph of an origami created by face division

3.2 Graph rewriting

In this section we give a language of graph rewriting. G-term t is defined by the following grammar:

$$t := x \mid \underline{x} \mid f[t_1, \dots, t_n]$$

Here, x denotes a variable, and the underlined one is a sequence variable. The sequence variable is indispensable since a function symbol f in g-term $f[t_1, \dots, t_n]$ representing a hyperedge has an unfixed arity. A g-pattern is a g-term possibly with a condition c . A g-pattern p is defined by the following grammar:

$$p := t \mid t/c$$

The syntax of condition c is not given here, as it is defined by an external language. Readers can assume that a condition c has the same syntax as a g-term, or more concretely has the syntax of Mathematica, which we are using

as our implementation language. The expression of the form t/c is called a conditional g-term. The intended use of the conditional g-term is as a left-hand side of a graph rewrite rule, which is defined below. It is used for conditional pattern matching. During pattern matching using a substitution θ for selecting a subgraph, if $c\theta$ evaluates (by an external evaluator) to True, $(t/c)\theta$ represents $t\theta$, and otherwise it represents \perp . We use u to denote either a g-term or a g-pattern.

As a meta notation we use $\langle u_1, \dots, u_n \rangle$ to denote $\text{List}[u_1, \dots, u_n]$.

Definition 11 (Graph rewrite rule). *A graph rewrite rule (rewrite rule for short) is a pair of g-terms*

$$g_l \rightarrow g_r$$

where $g_l := \langle u_1, \dots, u_m \rangle$ and $g_r := \langle t_1, \dots, t_n \rangle$.

Definition 12 (Graph rewriting).

A graph \mathcal{G} is rewritten to \mathcal{G}' by a rewrite rule $r := g_l \rightarrow g_r$, denoted by

$$\mathcal{G} \Rightarrow_r \mathcal{G}'$$

if there exist terms s_1, \dots, s_m , and a substitution θ such that $\{s_1, \dots, s_m\} \subseteq \mathcal{G}$, $g_l\theta$ after the evaluation of the conditions, if any, is $\langle s_1, \dots, s_m \rangle$ and $\mathcal{G}' = \mathcal{G} \setminus \{s_1, \dots, s_m\} \cup \{t_1, \dots, t_n\}$, where $g_r\theta = \langle t_1, \dots, t_n \rangle$.

4 Fold as a graph rewriting

We are now ready to describe the fold explained in Subsection 2.3 in graph rewriting framework. We recall that the fold consists of the following operations:

- (1) division of faces,
- (2) update of the adjacency relation,
- (3) update of superposition relation induced by face division, and
- (4) update of superposition relation induced by face rotation.

These operations are preformed in sequence, and we describe them in graph rewriting.

Face division We consider the division of a face f into f_1 and f_2 by a ray r as shown in Figs. 2 and 3. Figure 3 is the subgraph of the entire graph of the origami that we are working on.

The graph was transformed in the following steps from the graph of the previous step:

- (1) Construct nodes f_1 and f_2 .
- (2) Construct the edge e_1 that links f with f_1 and e_2 that links f with f_2 . Depending on whether the divided faces are to the left or right of the ray r , attach the labels R (R for Right) or L (L for Left) to the edges. In this case, the label of e_1 is L since face f_1 is to the left of the ray, and the label of e_2 is R.

- (3) Construct the edges e_3 and e_4 issuing from f_1 and from f_2 , respectively. We have $s(e_3) = \langle f_1, a_1, \dots, a_i, f_2 \rangle$, $t(e_3) = f_1$, $s(e_4) = \langle f_2, a_i, \dots, a_n, a_1, f_1 \rangle$ and $t(e_4) = f_2$. We label those edges by A (A for Adjacency) since the constructed edges represent the adjacency relation.

Face update The graph constructed in the face division step has to be updated since some of other faces are also divided, but the edges still link to those nodes of previous (undivided) faces. This step does this face update. This update is straightforward by the following rewrite rules:

$$\{L[f, f_1], A[f_1, \underline{x}], L[g, g_1] / ; g \neq g_1 \wedge g \in \{x\} \} \\ \rightarrow \{L[f, f_1], A[f_1, \underline{x}\{g \rightarrow g_1\}], L[g, g_1]\}$$

$$\{R[f, f_1], A[f_1, \underline{x}], R[g, g_1] / ; g \neq g_1 \wedge g \in \{x\} \} \\ \rightarrow \{R[f, f_1], A[f_1, \underline{x}\{g \rightarrow g_1\}], R[g, g_1]\}$$

Update of superposition relation induced by division Suppose that faces f and g such that $f \succ g$ are divided into $\langle f_1, f_2 \rangle$ and $\langle g_1, g_2 \rangle$, respectively. In the graph of Fig. 4, the edge e_5 is labeled S (S for Superposition) since $f \succ g$. We have $f_1 \succ g_1$ if $f_1^\circ \cap g_1^\circ \neq \emptyset$, and $f_2 \succ g_2$ if $f_2^\circ \cap g_2^\circ \neq \emptyset$. In Fig. 4, we assume that $f_1^\circ \cap g_1^\circ \neq \emptyset$ and $f_2^\circ \cap g_2^\circ \neq \emptyset$. Therefore, the edges e_6 and e_7 are added in this step. The A labeled edges are omitted for simplicity.

This transformation is realized by the following rewrite rule:

$$\{S[f, g], L[f, f_1], R[f, f_2], L[g, g_1] / ; f_1^\circ \cap g_1^\circ \neq \emptyset, R[g, g_2] / ; f_2^\circ \cap g_2^\circ \neq \emptyset \} \\ \rightarrow \{S[f_1, g_1], S[f_2, g_2], S[f, g], L[f, f_1], R[f, f_2], L[g, g_1], R[g, g_2]\}$$

The g-terms $S[f_1, g_1]$, $S[f_2, g_2]$, $S[f, g]$, $L[f, f_1]$, $R[f, f_2]$, $L[g, g_1]$ and $R[g, g_2]$ match with edges e_6 , e_7 , e_5 , e_1 , e_2 , e_3 and e_4 , respectively.

Depending on whether the faces are divided and non-divided, and on whether they are to the right or left of the ray in the case of non-divided, we have other four cases. Defining the rewrite rules for these cases is straightforward.

Update of superposition relation induced by rotation For any pair of faces f and g , we have to check if they are related by the superposition. We distinguish the following three cases for each pair.

- (1) We rotate the face f that is to the right of r after the division. It may overlap with the face g that is to the left of r . In this case, $f \succ g$ if no other faces above g is below f .
- (2) We rotate the faces f and g that are $f \succ g$ and are to the right of r . We delete the relation $f \succ g$ and newly add $g \succ f$.
- (3) Otherwise no superposition is added.

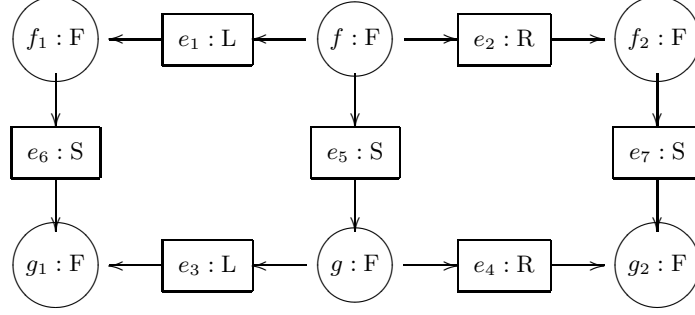


Fig. 4. Addition of superposition relation by division

Example 2. Figure 5 shows the origami at the intermediary step during the fold. We are about to make a fold along the fold line indicated by the dotted line. The ray r corresponding to the fold line runs from lower right to upper left. The faces that form the base layer of the origami has been divided by r into faces a and f . At this step, the origami consists of faces a, b, c, d, f, g and h . We see that the faces a, b, c and d are to the right of r , and that the faces f, g and h are to the left of r .

Figure 6 shows the result of the rotation along r , where the faces to the right are moved.

Figure 7 shows the graph representation of the intermediary origami of Fig. 5. Faces g and h superpose face f . Faces b and c superpose a . Face d superposes b . For simplicity we omit the A, L and R labeled edges.

Figure 8 shows the graph representation of the origami of Fig. 6. The rotation effects the graph transformation from the graph of Fig. 7 to that of Fig. 8. The newly added S labeled edges are e_6, e_7, e_8, e_9 and e_{10} . This is the result of the following computation: Let $RF = \{a, b, c, d\}$ and $LF = \{f, g, h\}$.

- (1) We take d from RF and g from LF .
- (2) Since $d^\circ \cap g^\circ = \emptyset$, we have no superposition relation between d and g .
- (3) We take h from LF , and check the overlap of d and h .
- (4) We have $d^\circ \cap h^\circ \neq \emptyset$, we have $d \succ h$. This is shown by the edge e_6 .
- (5) Likewise, we add the edge e_7 .
- (6) We add the edge e_8 since we had $d \succ b$, the edge e_9 since we had $b \succ a$ and edge e_{10} since we had $c \succ a$.

5 Conclusion

We have presented an abstract model of origami. The abstraction lead to graph theoretic modelling and transformation of origami. More concretely, we formalized an origami as a hypergraph and define the fold as algebraic graph transformations. The algebraic graph theoretic formalism enables us to reason origami in

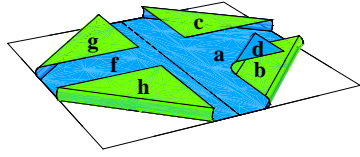


Fig. 5. Origami at the intermediary step during the fold

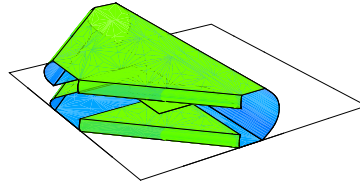


Fig. 6. Origami after the fold

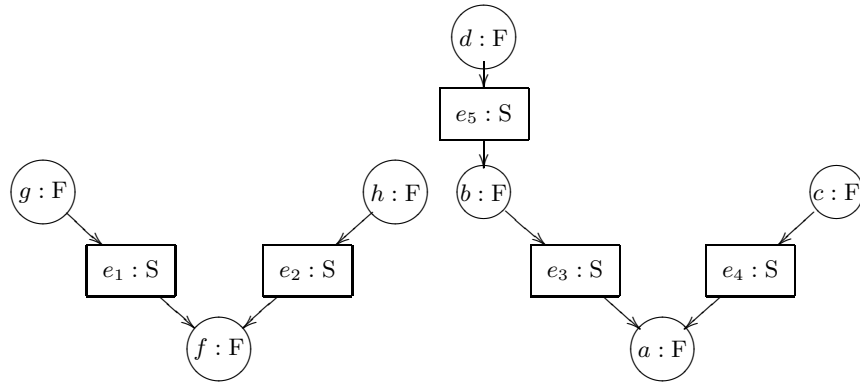


Fig. 7. Graph representation of the origami of Fig. 5

two separate domains of discourse, i.e. pure combinatoric domain, and geometric domain $\mathcal{R} \times \mathcal{R}$, and thus helps us to further tackle challenging problems such as of discovering a new construction given an origami shape, and of discovering a new origami that has certain geometric properties.

Our formalism follows closely that of algebraic and categorical graph theory, and we anticipate the rich theory in this area will be applicable to our computational origami research.

References

1. H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Springer-Verlag, 2006.

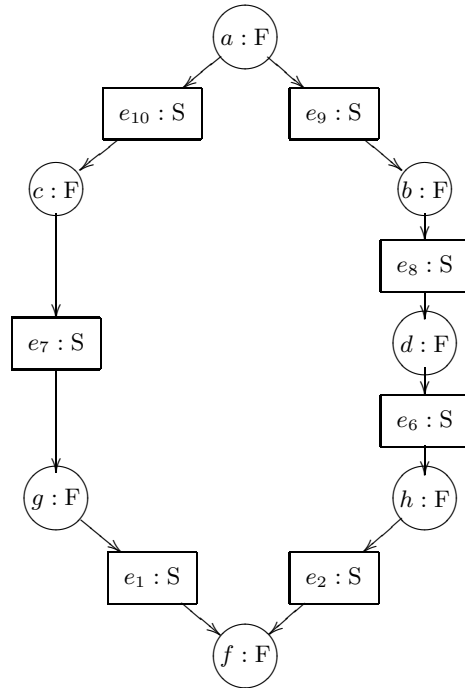


Fig. 8. Graph representation of the origami of Fig. 6

2. 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.
3. 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.
4. T. Ida, H. Takahashi, M. Marin, F. Ghourabi, and A. Kasem. Computational Construction of a Maximal Equilateral Triangle Inscribed in an Origami. In *Mathematical Software - ICMS 2006*, volume 4151 of *Lecture Notes in Computer Science*, pages 361–372. Springer, 2006.