Proving and Constraint Solving in Computational Origami

Tetsuo Ida, Dorin Tepeneu

Department of Computer Science, University of Tsukuba, Japan

Bruno Buchberger

Research Institute for Symbolic Computation, Johannes Kepler University, Austria

Judi Robu

Department of Computer Science, University of Cluj, Rumania

Origami

- Origami is a Japanese traditional art of paper folding.
- The word origami is the combined word coming from ori (fold) and kami (paper).
- For several centuries, origami has been popular among Japanese common people as an art, as playing toys, and teaching material for children.

- *It is a sophisticated and powerful technique of constructing geometrical objects.*
- *What interests us is the computational aspects of Origami, as well as of artistic and pedagogical ones.*

Computational Origami

- *We are proposing computational origami, as part of a discipline of origami science [Haga 1999].*
- *We believe that the rigor of paper folding and the beauty of origami artworks enhance greatly when the paper folding is supported by a computer.*
- *In our earlier work [APLAS 2003], computational origami performs paper folding by solving both symbolically and numerically certain geometrical constraints, followed by the visualization of origami by computer graphics tools.*
- *We reported [AISC 2004] the extention of the system for proving the correctness of the origami constructions.*

Origami construction by hand

Origami is easy to practice

Repeat

find a crease on the origami

fold the origami along the crease

until you are satisfied

■ Origami construction by computer

- Origami construction is made even easier with a computer.
- We invoke a sequence of origami folding functions on the Mathematica Notebook or by the interaction with our system, running in the computing server, using a standard web browser.
- The constructed final result, as well as the results of the intermediate steps, can be visualized and manipulated.
- Moreover, in cases where a proof is needed, the system will produce the proof of the correctness of the construction.

Origami Construction [Huzita]

(01) Given two points, we can make a fold along the crease passing through them.

(02) Given two points, we can make a fold to bring one of the points onto the other.

(03) Given two lines, we can make a fold to superpose the two lines.

(04) Given a point P and a line m, we can make a fold along the crease that is perpendicular to m and passing through P.

(05) Given two points P and Q and a line m, either we can make a fold along the crease that passes through Q, such that the fold superposes P and m, or we can determine that the fold is impossible.

(06) Given two points P and Q and two lines m and n, either we can make a fold along the crease, such that the fold superposes P and m, and Q and n, simultaneously, or we can determine that the fold is impossible.

Implementation of Origami Folds

- Huzita's origami construction tells us the possibility of finding crease(s).
- Finding the creases is reduced to solving geometrical constraints (maximum 3rd degree polynomial system).
- We define OFold (Origami Fold) functions.

OFold (Origami Fold Function)

Need to specify points to compute the crease, points to determine the face to be moved, and the direction of the fold (mountain or valley)

(01) OFold[X, Along→PQ]

All the faces contaning the values of X are moved

(02) OFold[P, Q]

(03) OFold[AB, CD]

(04) OFold[X, AlongPerpendicular→{P, AB}]

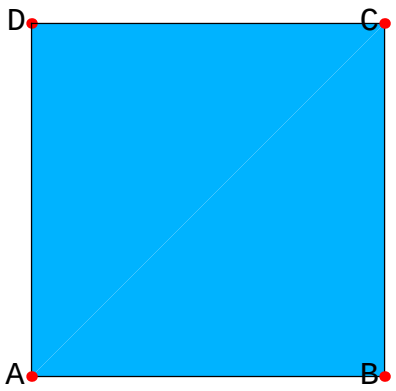
(05) OFold[P, AB, Though→Q]

(06) OFold[P, AB, Q, CD]

9 of 22

Trisecting an angle

```
In[4]:= NewOrigami[Square[10, MarkPoints → {"A", "B", "C", "D"}]];
        ShowOrigami::id :   1
```



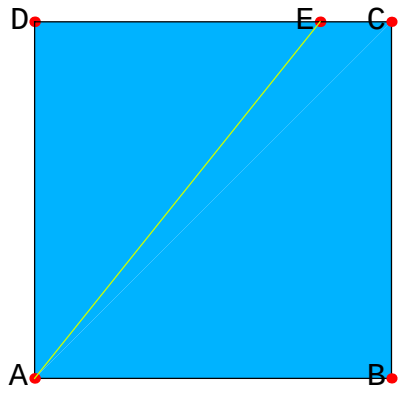
10 of 22

Let E be an arbitrary point on the origami

We will trisect the angle ∠EAB.

```
In[5]:= PutPoint[{"E", Point[8, 10]}]; ShowFolded[
          Show → {ImageSize → 200, More -> Graphics3D[{Hue[0.2], GraphicsLine[E, A]}]}];
```

```
ShowOrigami::id :    1
```

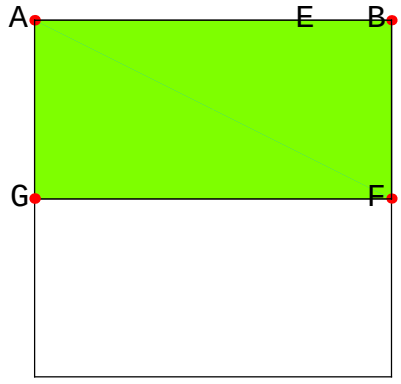


11 of 22

Fold to make creases

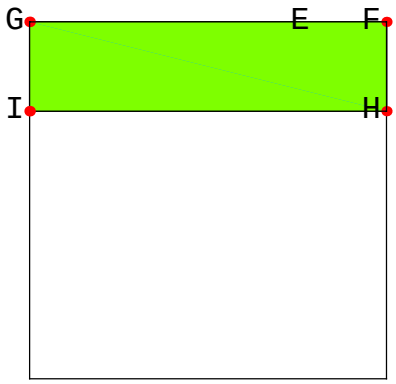
```
In[6]:= OFold[A, D];
```

```
ShowOrigami::id :    2
```



In[7]:= **OFold[G, A];**

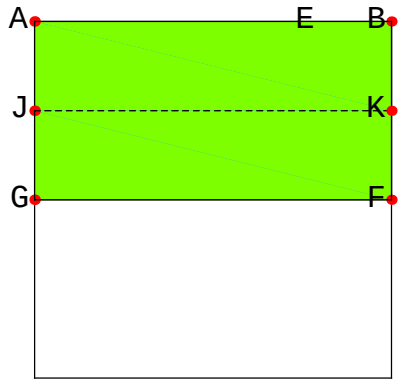
```
ShowOrigami::id :    3
```



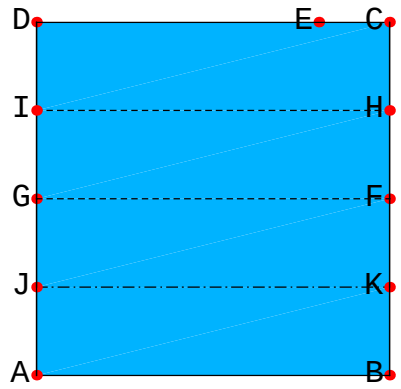
Unfold All

In[9]:= **UnfoldAll[];**

```
ShowOrigami::id :    4
```




```
ShowOrigami::id :    5
```



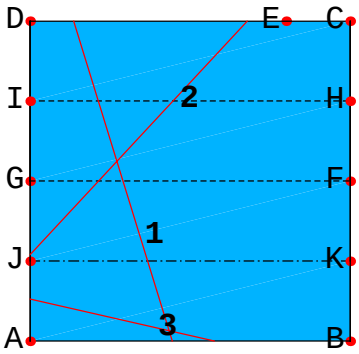
13 of 22

Make a fold

Bring G to AE and A to JK.

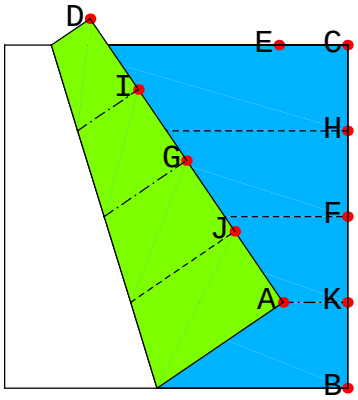
```
In[10]:= OFold[G, AE, A, JK];
          Which line(1,2,3)?
        ShowOrigami::id :   5
```

Specify the line number.



Choose crease 1

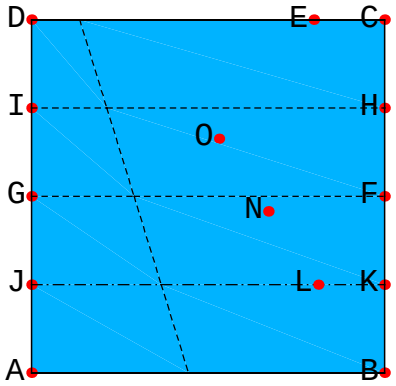
```
In[11]:= OFold[Along → 1, Move → A, MarkCrease → False];
        ShowOrigami::id :   6
```



Let L, N and O be the points to which A, J and G are moved by the fold, and then unfold.

```
In[12]:= CopyPoint[{A, J, G}]; Unfold[];
```

```
ShowOrigami::id :    7
```

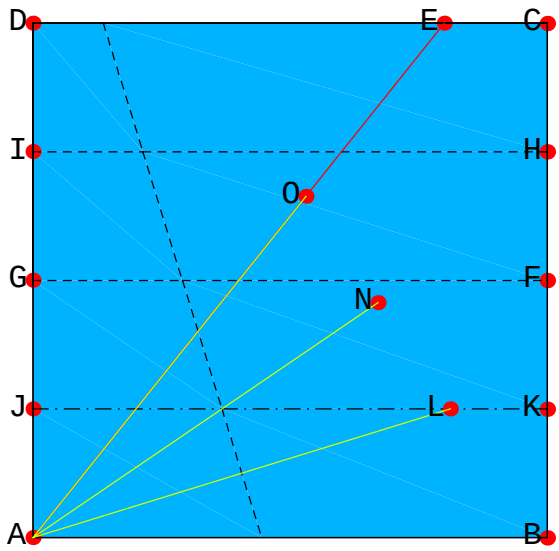


16 of 22

We conjecture that both segments AL and AN trisects ∠EAB.

```
In[13]:= ShowOrigami[More -> Graphics3D[{{Hue[0.2], GraphicsLine[A, L],
           GraphicsLine[A, N], GraphicsLine[A, O]}, {Hue[0], GraphicsLine[A, E]}}]];
```

```
ShowOrigami::id :    7
```



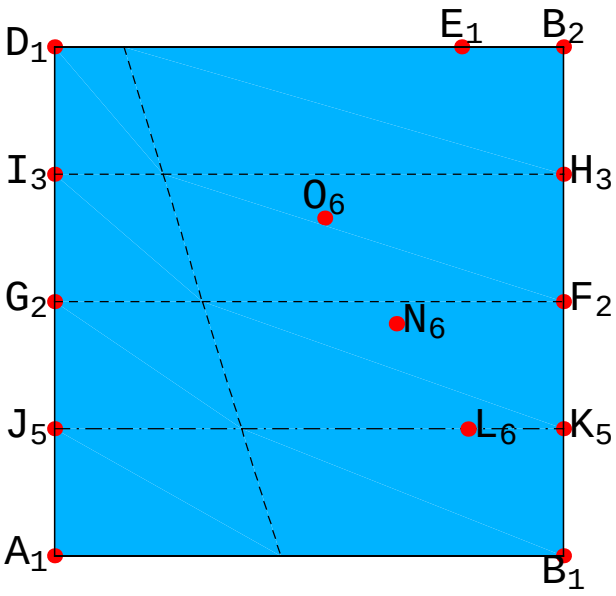
17 of 22

Finding a proof

Proposition: $\angle EAB = 3\,\angle LAB \wedge \angle NAB = 2\,\angle LAB$

```
In[14]:= BeginProof[];
```

In[15]:= **ShowProofSupport[];**



18 of 22

Preparing for proof

Gather necessary geometric constraints

In[16]:= **props = GatherProperty[]**

```
Out[16]= {BrBrP[line7, {G, 2}, line8, {A, 1}, line9],
           BringP[line5, {A, 1}, {D, 1}], BringP[line6, {G, 2}, {D, 1}],
           SymmetricPointP[{A, 1}, {L, 6}, line7], SymmetricPointP[{A, 2}, {G, 2}, line6],
           SymmetricPointP[{B, 1}, {B, 2}, line5], SymmetricPointP[{D, 1}, {D, 6}, line7],
           SymmetricPointP[{F, 2}, {F, 3}, line6], SymmetricPointP[{G, 2}, {O, 6}, line7],
           SymmetricPointP[{H, 3}, {K, 5}, line5], SymmetricPointP[{I, 3}, {I, 6}, line7],
           SymmetricPointP[{I, 3}, {J, 5}, line5], SymmetricPointP[{J, 5}, {N, 6}, line7],
           ThruP[line8, {A, 1}, {E, 1}], ThruP[line9, {J, 5}, {K, 5}],
           XPointP[{F, 2}, {line2, line5}], XPointP[{G, 2}, {line4, line5}],
           XPointP[{H, 3}, {line2, line6}], XPointP[{I, 3}, {line4, line6}]}
```

Cartesian coordinate mapping

In[17]:= **cmap = CoordinateMapping[props,**
InitialShape → SquareP[{Point[0, 0], r}, {"A", "B", "C", "D"}]];

Premise generation

```
In[18]:= premise = ToAlgebraic[props, cmap]
```

$$\text{Out[18]}= \Big\{c8,\ a5\,r,\ c5+\frac{b5\,r}{2},\ -r+x5,\ x7,\ -r+x8,\ x9,\ c9+a9\,x11+b9\,y11,\ c9+a9\,x12+b9\,y12,$$
$$-b7\,x13+a7\,y13,\ c7+\frac{a7\,x13}{2}+\frac{b7\,y13}{2},\ b7\,(x11-x14)+a7\,(-y11+y14),$$
$$c7+\frac{1}{2}a7\,(x11+x14)+\frac{1}{2}b7\,(y11+y14),\ b5\,(r-x2)+a5\,y2,$$
$$c5+\frac{1}{2}a5\,(r+x2)+\frac{b5\,y2}{2},\ -b7\,x3+a7\,(-r+y3),\ c7+\frac{a7\,x3}{2}+\frac{1}{2}b7\,(r+y3),$$
$$c8+a8\,x4+b8\,y4,\ c5+a5\,x5+b5\,y5,\ b6\,(x5-x6)+a6\,(-y5+y6),$$
$$c6+\frac{1}{2}a6\,(x5+x6)+\frac{1}{2}b6\,(y5+y6),\ b6\,x7+a6\,(r-y7),\ b7\,(-x15+x7)+a7\,(y15-y7),$$
$$c5+a5\,x7+b5\,y7,\ c6+\frac{a6\,x7}{2}+\frac{1}{2}b6\,(r+y7),\ b6\,(x1-x7)+a6\,(-y1+y7),$$
$$c6+\frac{1}{2}a6\,(x1+x7)+\frac{1}{2}b6\,(y1+y7),\ c7+\frac{1}{2}a7\,(x15+x7)+\frac{1}{2}b7\,(y15+y7),$$
$$b5\,(-x12+x8)+a5\,(y12-y8),\ c6+a6\,x8+b6\,y8,\ c5+\frac{1}{2}a5\,(x12+x8)+\frac{1}{2}b5\,(y12+y8),$$
$$b7\,(-x10+x9)+a7\,(y10-y9),\ b5\,(-x11+x9)+a5\,(y11-y9),\ c6+a6\,x9+b6\,y9,$$
$$c7+\frac{1}{2}a7\,(x10+x9)+\frac{1}{2}b7\,(y10+y9),\ c5+\frac{1}{2}a5\,(x11+x9)+\frac{1}{2}b5\,(y11+y9),$$
$$-1+r\,\eta1,\ -1+(a5^2+b5^2)\,\eta2,\ -1+(a6^2+b6^2)\,\eta3,\ -1+(a7^2+b7^2)\,\eta4,$$
$$-1+(a8^2+b8^2)\,\eta5,\ -1+(a9^2+b9^2)\,\eta6,\ c8+a8\,\mu1+b8\,\mu2,\ b7\,(x7-\mu1)+a7\,(-y7+\mu2),$$
$$c7+\frac{1}{2}a7\,(x7+\mu1)+\frac{1}{2}b7\,(y7+\mu2),\ -b7\,\mu3+a7\,\mu4,\ c7+\frac{a7\,\mu3}{2}+\frac{b7\,\mu4}{2},\ c9+a9\,\mu3+b9\,\mu4\Big\}$$

Conclusion polynomial generation

Let $\gamma = \angle$EAB, $\beta = \angle$NAB and $\alpha = \angle$LAB.

Our conclusion is $\beta = 2\alpha \wedge \gamma = 3\alpha$

Since function tan restricted to $(0, \pi/2)$ is bijective , we will prove

$\tan[\beta] = \tan[2\alpha] \wedge \tan[\gamma] = \tan[3\alpha]$

```
In[19]:= tanγ = E1 /. Point[x_, y_, ___] :> y / x;
         tanβ = N6 /. Point[x_, y_, ___] :> y / x;
         tanα = L6 /. Point[x_, y_, ___] :> y / x
```

$$\text{Out[21]}= \frac{y13}{x13}$$

Using the well-known elementary trigonometric formulas, we have:

```
In[22]:= tan3α = Simplify[(3 Tan[x] - Tan[x]^3) / (1 - 3 Tan[x]^2) /. Tan[x] → tanα];
         tan2α = Simplify[(2 Tan[x]) / (1 - Tan[x]^2) /. Tan[x] → tanα]
```

$$\text{Out[23]}= \frac{2\,x13\,y13}{x13^2-y13^2}$$

The conclusion polynomial set is {Numerator[Together[tanγ-tan3α]]=0, Numerator[Together[tanβ-tan2α]]=0}, which can be translated to algebraic form as follows. The variables were automatically generated when the coordinate mapping table was constructed.

```
In[24]:= concl =
          ToAlgebraic[{Numerator[Together[tanβ - tan2α]], Numerator[Together[tanγ - tan3α]]}]

Out[24]= {-2 x13 x14 y13 + x13^2 y14 - y13^2 y14, -3 x13^2 x4 y13 + x4 y13^3 + x13^3 y4 - 3 x13 y13^2 y4}
```

19 of 22

Sending all the data to theorem prover Theorema

Establishing link with Theorema

```
In[71]:= (* thma=LinkConnect["50000@192.168.11.2",LinkProtocol→"TCPIP" ] *)

In[25]:= thma = LinkConnect["50000@localhost", LinkProtocol → "TCPIP" ]

Out[25]= LinkObject[50000@ida2, 2, 2]
```

Send data via the link

```
In[27]:= SendTheoremaFormula[thma, premise, concl, "TrisectingAngle"];
```

Read the proof via the link

```
In[28]:= NotebookSave[LinkRead[thma], "TrisectingAngleProof.nb"];
```

Close the link

```
In[29]:= LinkClose[thma];

In[30]:= EndProof[];
```

20 of 22

Summary and future research

- Computational origami
- Reverse origami problem
- Website for computational origami
- Computational origami for art

21 of 22

Outline

Origami (paper folding) has a long tradition in Japan's culture and education. We are developing a computational origami system, based on symbolic computation system Mathematica,for performing and reasoning about origami on the computer.This system is based on the implementation of the six fundamental origami folding steps (origami axioms) formulated by Huzita.In this paper, we show how our system performs origami folds by constraint solving,visualizes each step of origami construction,and automatically proves general theorems on the result of origami construction using algebraic methods.We

illustrate this by a simple example of trisecting an angle by origami.The trisection of an angle is known to be impossible by means of a ruler and a compass.The entire process of computational origami shows nontrivial combination of symbolic constraint solving, theorem proving and graphical processing.

22 of 22

Initialization

```
In[1]:= Needs["OrigamiBasics`"]

        Needs::nocont :
         Context OrigamiBasics` was not created when Needs was evaluated. More…

In[2]:= SetOptions[ShowOrigami, ShowFrame → True] ; $exact = True;

In[3]:= SetOptions[ShowFolded, Show → {ImageSize → 200}] ;

In[5]:= $lexicographic = False ; $exact = True ;
```