Interactive vs. automated proofs in computational origami

Tetsuo Ida

University of Tsukuba, Japan

Motivation

- **The public are more and more concerned with security and safety in our life, e.g. natural calamities, financial risks, radioactivities, and viruses.**
- **In what ways and to what extent , we, i.e. experts in (at least) one professional domain, can strive to ensure the security and safety of the subjects of study, and explain to the public that we are no less concerned with those notions?**
- **The tone of the (mass) media sounds that experts form a closed society of profession and hide what are detrimental to their interests from the eyes of the public.**

Motivation - continued

- In my view, this is not true. Scientists are careful about the statements that may convey to the public that something is 100% sure, or that something bad will happen definitely.
- We recall "We must know, but (by now we are aware that) we shall (not) know everything."
- Then, I will argue what we can do in our profession is to increase the rigor of our knowledge as well as to ensure the process of acquiring high quality new knowledge.

My research interest

- **Object of study: geometric objects, in particular, origami**
- **Methodology: (until mid 80s) computer architecture, hardware construction; physical entity behind computation**
- **Methodology: (from then until now) software design, computation model, verification**

Security and safety is the real issue in classical, modern and computational geometry.

- **image processing**
- **face recognition**
- **visualization**
- **animation**

Loss of rigor in geometry

- **Perception of images by human is inexact.**
- **We appeal to intuition where intuition alone cannot fill the gap of reasoning steps.**
- **Non-degeneracy conditions are not stated.**

Origami (paper fold)

- **Origami is a concrete example of theories of fold, and more.**
- **Nature is abundant with folds; imagine the process of flowering.**
- **We have industrial applications; automobile and space industries.**
- **Origami is a more powerful geometrical construction tool than straightedge and compass.**

How to fold?

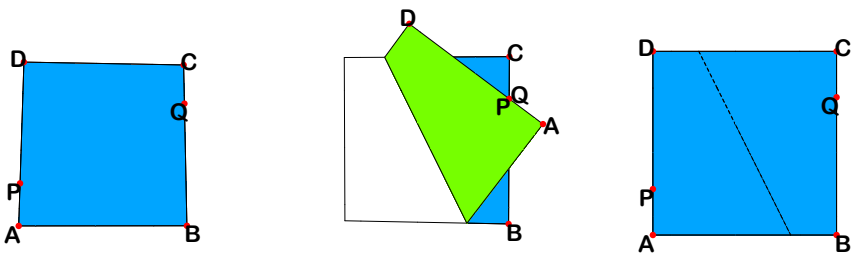
While folding a paper, we can perform the following.

- **Determine the line (called fold line) along which we make a fold.**
- **Then, fold along the fold line.**

How to determine fold line?

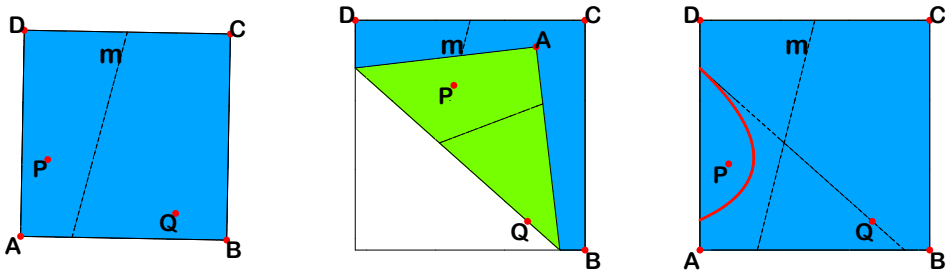
- **The fold line can be determined by superposing constructed points and lines.**

Superposition of two points P and Q



- the perpendicular bisector of the segment formed by the points, if the two points are distinct

Superposition of a point P and a line m



- tangents to the parabola defined by focus P and directrix m

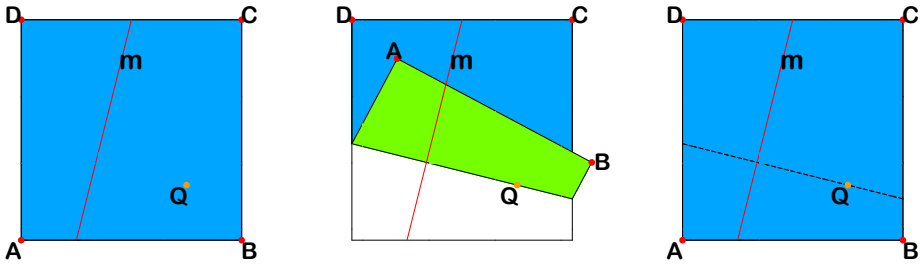
---

Superposition of lines m and n

1. **m == n**
   - **superposition of a line with itself**
2. **m ≠ n**

   Find two distinct points P and Q on the line m.
   - **Perform two point-line superpositions, i.e. P and n, and Q and n.**

Superposition of a line with itself



- perpendiculars to the line

Superposition pair

It is meaningful to talk about a pair of geometrical objects that are to be superposed.

Let s = (α, β) be a superposition pair (s-pair for short), where α and β are geometrical objects (so far, either a point or a line).

Some notations

- **P↕Q : perpendicular bisector of the segment PQ**
- **$\mathcal{B}$ (m) = $\{X \updownarrow Y \mid X, Y \in m, X \neq Y\}$: the set of perpendiculars to m generated by superposition pair (m, m).**
- **$\mathcal{I}$ (P) = $\{\gamma \mid P \in \gamma\}$: the set of lines that pass through P generated by superposition pair (P, P)**
- **Γ (P, m) = $\{X \updownarrow P \mid X \in m\}$: the set of tangents of the parabolas defined by focus P and directrix m generated by superposition pair (P, m)**

Huzita's basic origami operations (1989)

1. Given two distinct points, you can fold making the crease pass through both points (`ruler operation`).
2. Given two distinct points, you can fold superposing one point onto the other point (`perpendicular bisector`).
3. Given two distinct (straight) lines, you can fold superposing one line onto another (`bisector of the angle`).
4. Given one line and one point, you can fold making the crease perpendicular to the line and passing through the point (`perpendicular footing`).
5. Given one line and two distinct points not on this line, you can fold superposing one point onto the line and making the crease pass through the other point (`tangent from a point to a parabola`).
6. Given two distinct points and two distinct lines, you can fold superposing the first point onto the first line and the second point onto the second line at the same time.

Huzita's fold principle H

- **A method of defining fold lines using s-pairs**

  It consists of the rules of the following form:

  - **Fold origami O along fold lines $\gamma_1, .., \gamma_j$ defined by superposition pairs $\sigma_1, .., \sigma_k$.**
  - **In short, fold along $\{\gamma_1, .., \gamma_j\}$.**

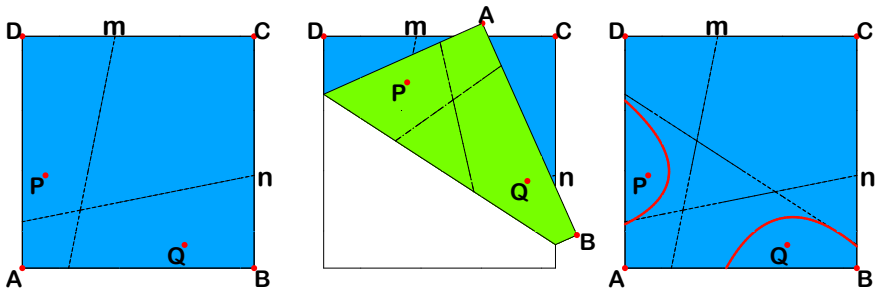
- **In Huzita's principle, j =1 and k $\leqslant$2.**

Method H

1. **Fold along $\mathcal{I}(\mathbf{P}) \bigcap \mathcal{I}(\mathbf{Q})$, where P ≠ Q. (Fold along the line passing through P and Q.)**
2. **Fold along {P ↕ Q} where P ≠Q.**
3.
4. **Fold along $\mathcal{B}(\mathbf{m}) \bigcap \mathcal{I}(\mathbf{Q})$.**
5. **Fold along $\Gamma(\mathbf{P}, \mathbf{m}) \bigcap \mathcal{I}(\mathbf{Q})$, where P ∉ m.**
6. **Fold along $\Gamma(\mathbf{P}, \mathbf{m}) \bigcap \Gamma(\mathbf{Q}, \mathbf{n})$, where P ∉ m ∧ Q ∉ n ∧ (P, m) ≠ (Q, n).**
7. **Fold along $\Gamma(\mathbf{P}, \mathbf{m}) \bigcap \mathcal{B}(\mathbf{n})$, where P ∉ m .**

Operation 6

Fold along a line to superpose P and m, and Q and n simultaneously.



Comparison with straightedge and compass (Euclidean geometry)

- **Huzita's principle is more powerful than straightedge and compass**
  - **By Huzita's fold principle, we can construct a trisector of an angle, while by straightedge and compass we cannot (P. L. Wanzel 1837)**

Origami constructible points

Theorem 10.3.4 (Galois Theory book by Cox 2004)
Let A = (0, 0), B = (1, 0), and D = (0, 1), and Identity the points on $\mathbb{R} \times \mathbb{R}$ with them on $\mathbb{C}$.
The (hpoint) set O, the set of origami constructible numbers, is a sub-field of $\mathbb{C}$.

1. Let $\alpha = a + i\,b$ where $a, b \in \mathbb{R}$. Then $\alpha \in O \Leftrightarrow a, b \in O$
2. $\alpha \in O \Rightarrow \sqrt{\alpha}, \sqrt[3]{\alpha} \in O$
3. $\alpha \in O$ $\Leftrightarrow$ **there are subfields**
   $Q = F_0 \subset F_1 \subset \ldots \subset F_{n-1} \subset F_n \subset \mathbb{C}$
   such that $\alpha \in F_n$ and $[F_i : F_{i-1}] = 2$ or 3 for $1 \le i \le n$.

Modeling point

```
datatype point = Point real real
```

```
abbreviation "pA ≡ Point 0 0"
abbreviation "pB ≡ Point 1 0"
abbreviation "pC ≡ Point 1 1"
abbreviation "pD ≡ Point 0 1"
```

Modeling line

```
datatype line = Line real real real
```

A line is represented as an equation a x + b y + c = 0
The following line constraint (on slope) makes the line representation unique :

```
fun slope::"real  ⇒ real  ⇒ bool"
where
"slope a b = ((b = 1) ∨ ((b = 0) ∧ (a = 1)))"
```

Origami constructible points and lines

inductive_set

```
hpoint::"point set" and
hline::"line set"
where
ha[intro!]:" pA ∈ hpoint" |
hb[intro!]:" pB ∈ hpoint" |
hc[intro!]:" pC ∈ hpoint" |
hd[intro!]:" pD ∈ hpoint" |
hx[intro!]:"⟦ m ∈ hline; n ∈ hline; ¬  m ∥ n ⟧ ⟹ Ox m n ∈
hrefl[intro!]: "⟦p ∈ hpoint; m ∈ hline ⟧ ⟹  p ⊣ m ∈ hpoin
```

and **hpoint**::point set

hline::line set

where

inductive_set (continued)

```
h1l[intro!]: "⟦ p ∈ hpoint; q ∈ hpoint; p ≠ q⟧ ⟹ O1 p q ∈
h2l[intro!]: "⟦ p ∈ hpoint; q ∈ hpoint; p ≠ q⟧ ⟹ O2 p q ∈
h4l[intro!]: "⟦ p ∈ hpoint; m ∈ hline⟧ ⟹ O4 m p ∈ hline"|
h5l[intro!]: "⟦ p ∈ hpoint; q ∈ hpoint; m ∈ hline; ¬ p ε m
                l ∈ O5_preset p m q⟧
             ⟹  l  ∈ hline" |
h6l[intro!]: "⟦ p ∈ hpoint; q ∈ hpoint; m  ∈ hline; n  ∈ h
                ¬(p = q ∧  m = n );
                (¬ p ε m ∧ ¬ q ε n); l ∈ O6_preset p m q n
              ⟹  l  ∈ hline" |
h7l[intro!]: "⟦ p ∈ hpoint; m  ∈ hline; n  ∈ hline; m ≠ n;
               ¬ p ε m;  l ∈ O7_preset p m n ⟧
               ⟹  l  ∈ hline"
```

```
definition
O5_preset::"point ⇒ line ⇒  point ⇒ line set" where
"O5_preset p m q ≡
{l . linecc l ∧ p ⊣ l  ε m ∧  q ε l}"

fun O5::"point ⇒ line  ⇒  point  ⇒ line" where
"O5 p m q  = (SOME n .n ∈ O5_preset p m q)"

definition
O6_preset::"point ⇒ line => point ⇒ line ⇒ line set"
 where
"O6_preset p m q n  ≡
{ l . linecc l  ∧   p ⊣ l ε  m ∧ q ⊣ l ε  n}"

fun O6::"point ⇒ line  ⇒  point  ⇒ line   ⇒ line" where
"O6 p m q n =  (SOME l. l ∈ O6_preset p m q n)"

definition
O7_preset::"point ⇒ line => line  ⇒ line set" where
"O7_preset p m n   ≡ {l
  . linecc l ∧ p ⊣ l ε m ∧ l ⊥ n}"
```

Some theorems

```
theorem O5_collapses_to_O1:
"∧p q m .⟦ p ∈ hpoint; q ∈ hpoint;  m ∈ hline; p ≠ q; p ε r
⟹ O1 p q ∈ O5_preset p m q"
theorem O6_collapses_to_O5:
"∧x  p q m n . ⟦linecc x; linecc m; linecc n; p ε m ⟧
 ⟹ x ∈  O5_preset q n p ⟹ x ∈  O6_preset p m q n"
```

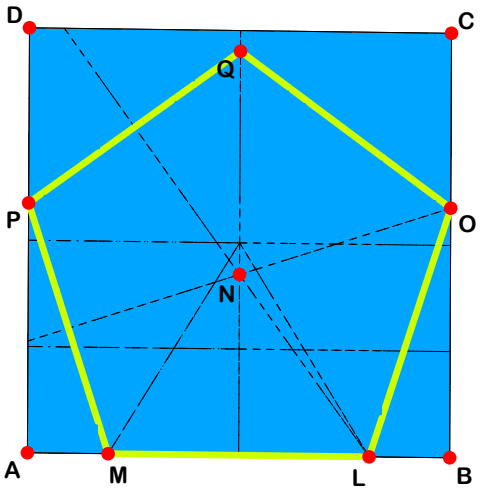
Origami theorems

- We proved hundreds of lemmas, which we would be useful for reasoning about elementary origami geometry.
- These are in some sense “obvious” from geometrical point of view.
- However, what intrigued me while stating (and developing the proofs) are degeneracy conditions, that are tacitly unstated in human proofs, but are absolutely necessary for computer provers to complete the proof.
- In geometrical theorem proving, mostly we need to translate geometrical statements to algebraic expressions, and then those algebraic expressions are manipulated. The transformations are not always straightforward, and moreover algebraic treatment does not clearly correspond to geometrical counterpart.
- Overall, the task so far is time-consuming and hard - a different kind of hardship from the one that we would encounter when we are developing a large software system over years.

Interlude - Logicomix

Photos from Logicomics, showing 1 + 1 = 2 took 362 page (deleted for the consideration of copy right)

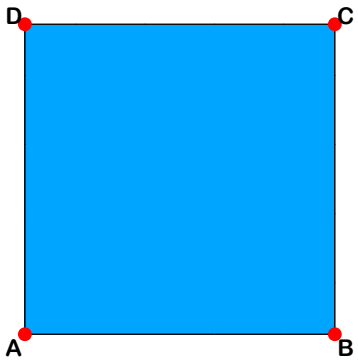
Regular pentagon by origami



"Classical" construction

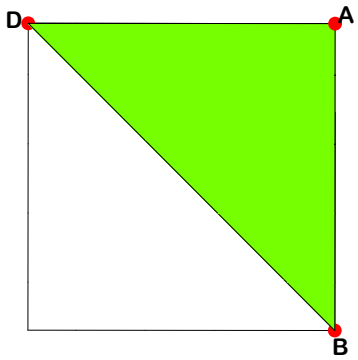
```
BeginOrigami[];
```

Step 1



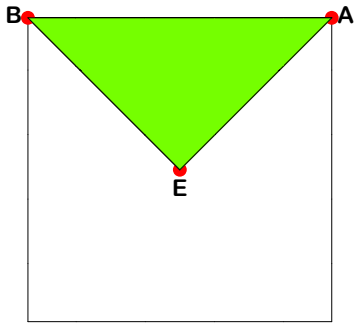

```
HO["A", "C"];
```

Step 2



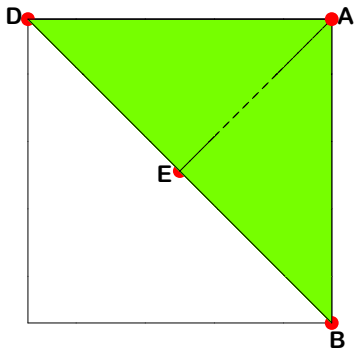

```
HO["B", "D"];
```

Step 3



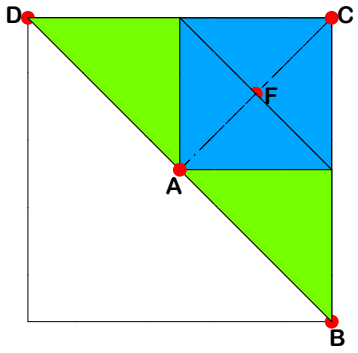

```
Unfold[];
```

Step 4



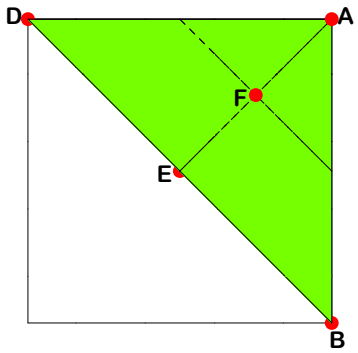

```
HO["A", "E", MarkPointOn → "AE"];
```

Step 5



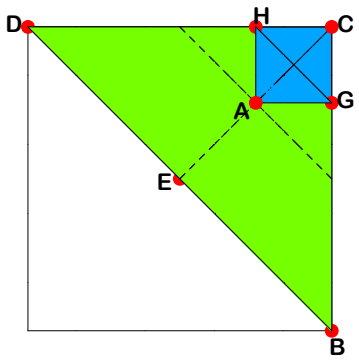

```
Unfold[];
```

Step 6



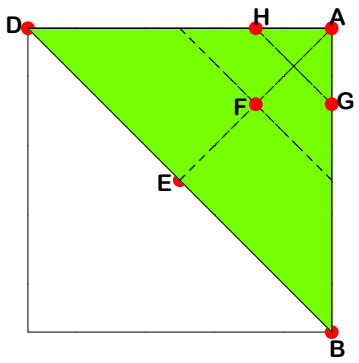

```
HO["A", "F", MarkPointOn → {"AB", "AD"}];
```

Step 7



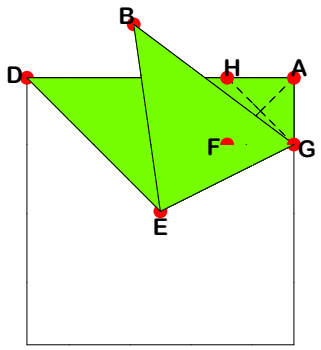

```
Unfold[];
```

Step 8



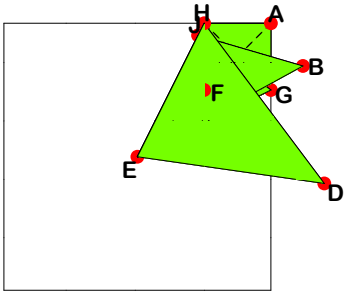

```
HO["B", Through → {"E", "G"}, MarkPointOn → False];
```

Step 9




```
HO["D", Through → {"E", "H"}, MarkPointOn → "BG"];
```

Step 10

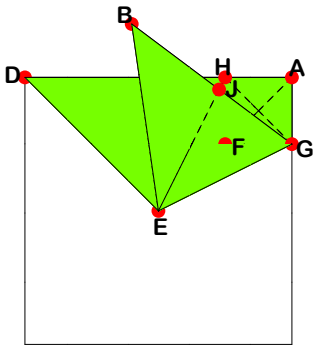



```
DupPoint["J"]
```

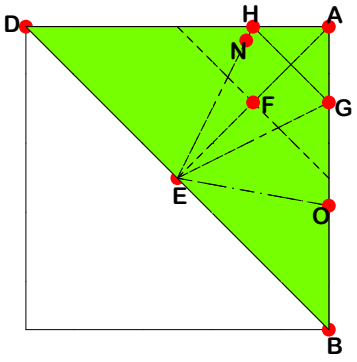
```
{{{M, Origami40}, {N, Origami40}, {O, Origami40}}}
```

```
UnfoldAll[];
```

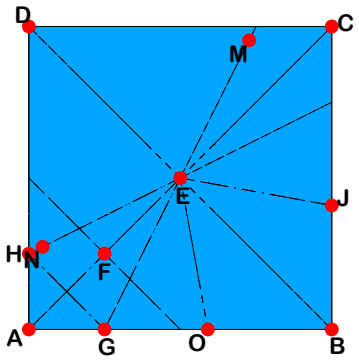
Step 11



Step 12

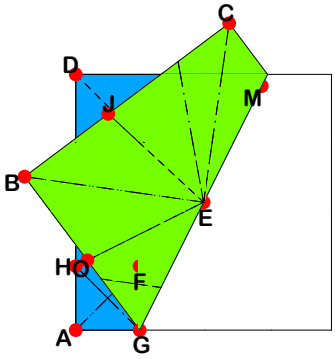


Step 13




```
HO["J", Through -> {"M", "E"}, MarkPointOn -> False];
```

Step 14

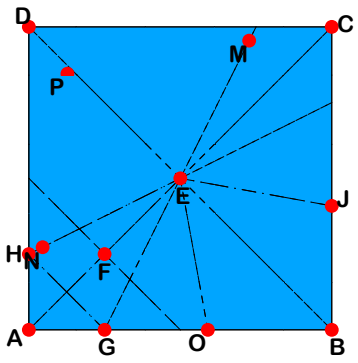



```
DupPoint["J"]
```

```
{{{P, Origami48}}}
```

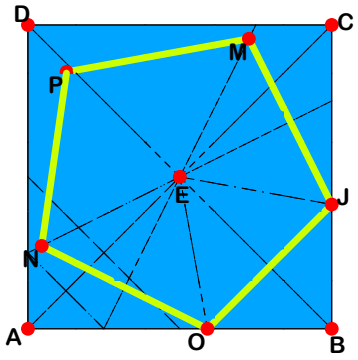
```
Unfold[];
```

Step 15




```
ShowFolded[ShowMarkPoints →
    {"A", "B", "C", "D", "E", "M", "P", "N", "O", "J"},
  More → {Thickness[0.01], Hue[0.2],
    GraphicsLine[{M, P, N, O, J}]}];
```

Step 15



Automated theorem proving by Groebner bases computation

Let P be $p_1 = 0 \wedge \ldots \wedge p_n = 0$. P is a premise.

Consider $\forall\ (P \Rightarrow C)$, **(1)**

where C is the conclusion that we want to obtain.

C is any logical combination of equalities.

C holds after the construction.

Negate (1)

$\exists \neg\ (P \Rightarrow C) \Leftrightarrow \exists\ (P \wedge \neg\ C) \Leftrightarrow \exists\ Q$,

where Q is a conjunctive normal form of $P \wedge \neg\ C$

Q is False $\Leftrightarrow$ $1 \in$ Ideal (Q) $\Leftrightarrow$ $1 \in$ GB (Ideal (Q)),

where Ideal (Q) is $\{q_1, \ldots, q_m\}$ and Q is $q_1 = 0 \wedge \ldots \wedge q_m = 0$

To reason about the construction

- **Proof of " correctness"**

To reason about the construction

- **Proof of " correctness"**

In[109]:=
```
Goal[∀α,α∈Num (VectorToComplex[α "EM"⃗ - "EP"⃗] == 0 ⇒
    (VectorToComplex[α^2 "EM"⃗ - "EN"⃗] == 0 ⋀
      VectorToComplex[α^3 "EM"⃗ - "EO"⃗] == 0 ⋀
      VectorToComplex[α^4 "EM"⃗ - "EJ"⃗] == 0 ⋀
      α^5 == 1))];
```

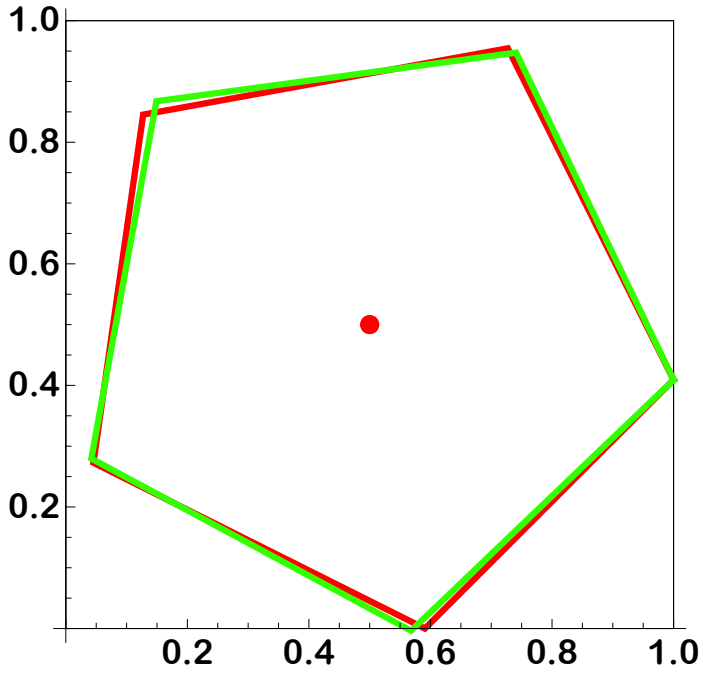
In[110]:=
```
Prove["Looks_like_pentagon Th."];
```

```
Groebner basis computation
  started at 2012/09/24 18:23:39 BST.
Proof by Groebner basis method failed.

   See the proofDoc for details.


CPU time used for Groebner basis computation is
 4.13299 seconds.
```

Analysis




```
Solve[Tan[θ/4] == 1/3, θ] // System`N
```

```
{{θ → 1.287}}
```

```
(2π/5) // System`N
```

```
1.25664
```

```
1.25664/1.287 // System`N
```

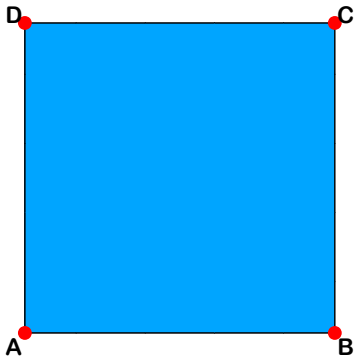
```
0.97641
```

A correct construction of a pentagon

Construction

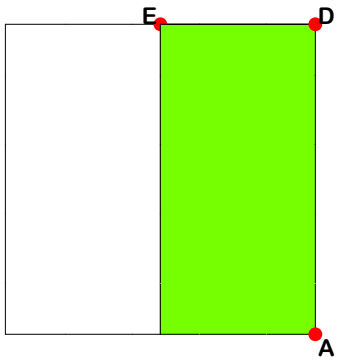
```
BeginOrigami[];
```

Step 1



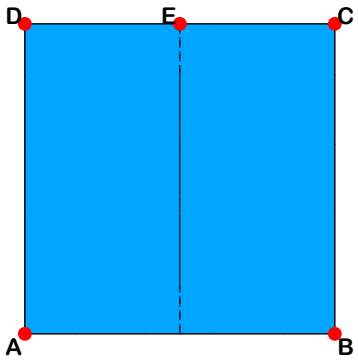

```
HO["A", "B", MarkPointOn → "CD"];
```

Step 2



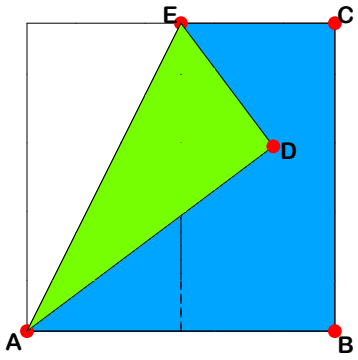

```
Unfold[];
```

Step 3



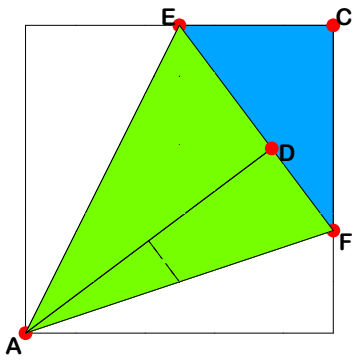

```
HO["D", Through → {"E", "A"}];
```

Step 4



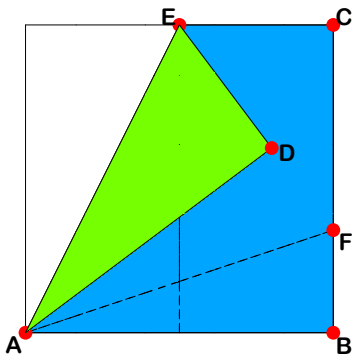

```
HO["B", "D", MarkPointOn → "BC"];
```

Step 5

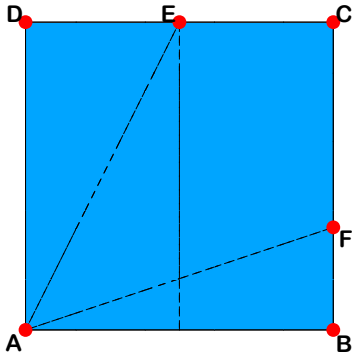



```
UnfoldAll[];
```

Step 6

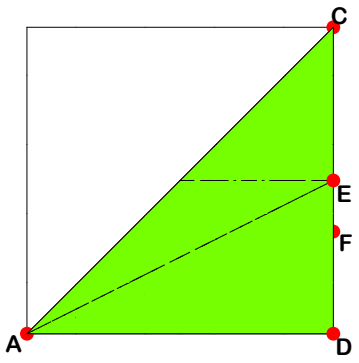


Step 7



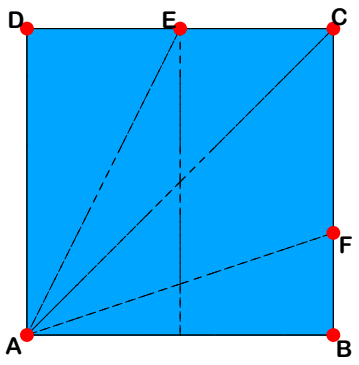

```
HO["D", "B", MarkPointOn → False];
```

Step 8



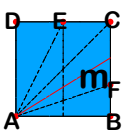

```
Unfold[];
```

Step 9

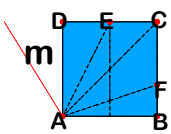



```
HO["F", "AC", Through → "A", MarkPointOn → "BC"];
```

Case 1

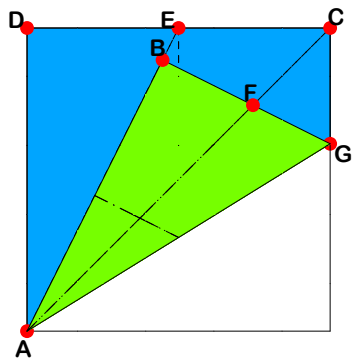


Case 2



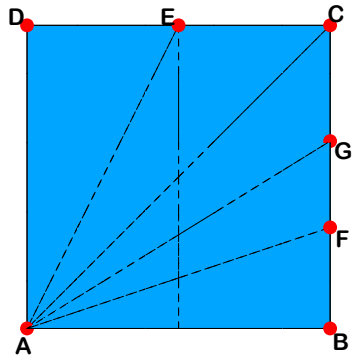
```
HO["F", "AC", Through → "A", MarkPointOn → "BC", Case → 1];
```

Step 10



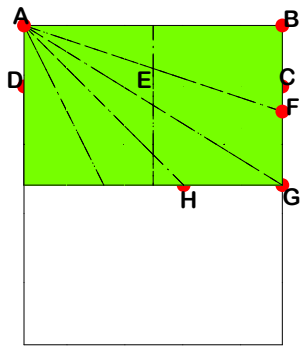

```
Unfold[];
```

Step 11



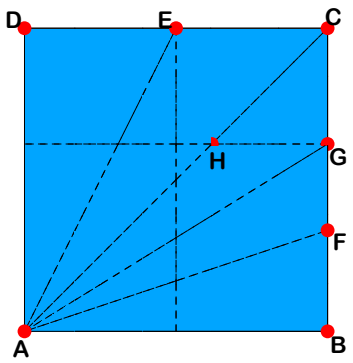

```
HO["BC", Through → "G", MarkPointOn → "AC"];
```

Step 12



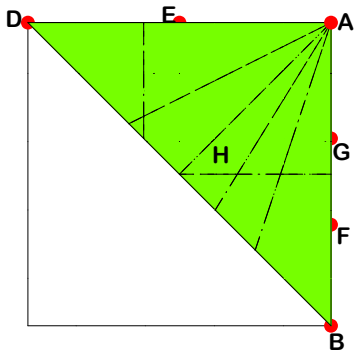

```
Unfold[];
```

Step 13



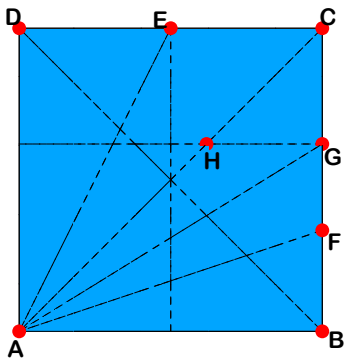

```
HO["A", "C", MarkPointOn → False];
```

Step 14



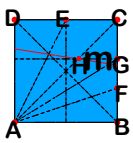

```
Unfold[];
```

Step 15

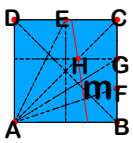



```
HO["C", "BD", Through → "H", MarkPointOn → {"AB", "CD"}];
```

Case 1

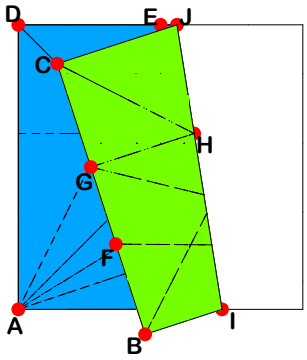


Case 2



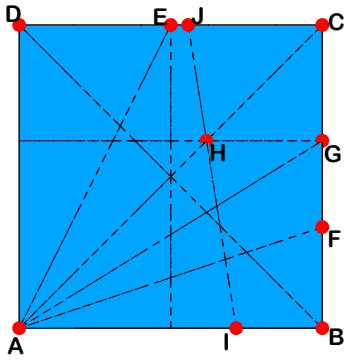
```
HO["C", "BD", Through → "H",
  MarkPointOn → {"AB", "CD"}, Case → 2];
```

Step 16



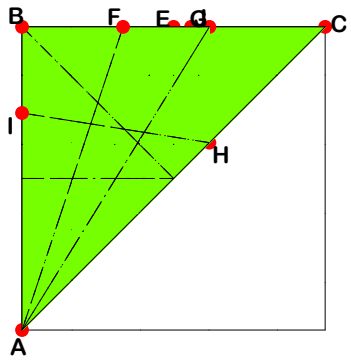

```
Unfold[];
```

Step 17




```
HO["B", Through → {"A", "C"}];
```

Step 18

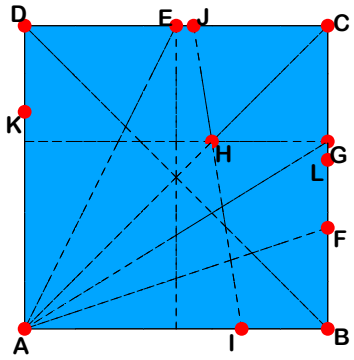



```
DupPoint[{"I", "J"}]
```

```
{{{K, Origami479}}, {{L, Origami479}}}
```

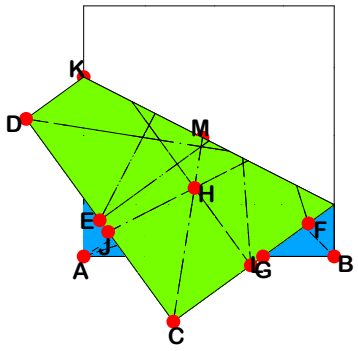
```
Unfold[];
```

Step 19




```
HO["L", "I", MarkPointOn → "AC"];
```

Step 20

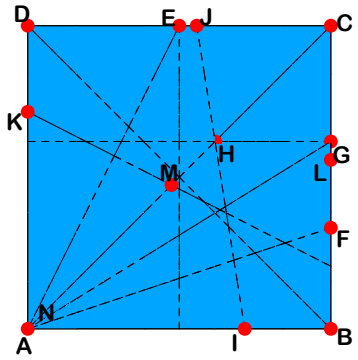



```
DupPoint["J"]
```

```
{{{N, Origami485}}}
```

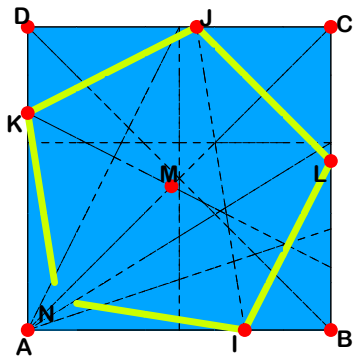
```
Unfold[];
```

Step 21




```
ShowFolded[ShowMarkPoints →
    {"A", "B", "C", "D", "M", "J", "K", "N", "I", "L"},
  More → {Thickness[0.01], Hue[0.2],
    GraphicsLine[{J, K, N, I, L}]}];
```

Step 21



■ Proof

In[91]:=

```
Goal[∀α,α∈Num (VectorToComplex[α "MJ"⃗ - "MK"⃗] == 0 ⇒
    (VectorToComplex[α^2 "MJ"⃗ - "MN"⃗] == 0 ⋀
      VectorToComplex[α^3 "MJ"⃗ - "MI"⃗] == 0 ⋀
      VectorToComplex[α^4 "MJ"⃗ - "ML"⃗] == 0 ⋀
      α^5 == 1))];
```

```
Prove["Max Pentagon", GroebnerBasis →
   {MonomialOrder → DegreeReverseLexicographic}];
```

```
Groebner basis computation
  started at 2012/09/22 16:31:46 BST.
Proof by Groebner basis method is successful.

   CPU time used for Groebner basis computation is
 1.852 seconds.
```

Conclusion

- By examples, I showed the interplay of modelling, construction and formal reasoning.
- This would hopefully illustrate the importance of computer-assisted formal reasoning, in particular theorem proving (interactive and automated).

Initialize