The Origami Computational Model

Christian Lavoie
McGill University
Downtown Campus
christian.lavoie@mail.mcgill.ca
clavoie@christianlavoie.com

November 26, 2002

Abstract

The Origami Computational Model is a new abstract machine based on the ancient Japanese art of paper folding, and recent work on Mathematical and Computational Origami.

Administrative stuff

- Author: Christian Lavoie
- Subject: Origami Computational Model
- What you should know before this talk:
  - Euclid's Edge & Compass
  - Scheme and/or LISP

Structure of the talk

- History and Context
- The Origami Computation Model Definition (the OCM)
- OCM algorithms & lower bounds
- OCM contra other models of computation
- Applications – What's the point?
- Future research – What's next?

Origami

- In Japan: No one seems to know exactly when it appeared, but most agree the Japanese were first
  - *oru* means "to fold"
  - *kami* means "paper"
- In Japan: Recreational use at the beginning of the 17th century, serious ceremonial use before that
- In Europe: Appeared around the 13th century (Spain)
- In England and the States: Appeared in the middle of the 20th century

Mathematical Origami

- Burgeoning of a mathematical view in 1945, in the States
- The axioms of Origami – paper in 1992 "Understanding Geometry through Origami Axioms" by Humiaki Huzita
- Formal definition of *What can be done?* using Origami, from a Mathematical point of view.
- Mainly a counterpoint to Euclid's Edge & Compass
- Interesting results have recently started to come out of the community
  - Topological results about face orientation
  - Origami crease patterns are 2-colourable

Computational Origami

- Algorithms, proofs, and the usual fruits.
- Few practical applications, but interesting ones:
  - Map & solar sails folding
  - Parachutes
  - Crease patterns for raytracing – landscapes
- Defined as the study of folding, of crease patterns and algorithms related to either.
- Current state of the art:
  - Universality results
  - Efficient decision algorithms
  - Computational intractability

Design goals

- An original way to force teachers to accept flying paper airplanes!
- A machine that will use mathematical origami concepts as basic operations
- A machine that will let one use recent Computational Origami results to prove feasability results in the machine itself
- Did I mention a machine that is fun to work with?

Implementation choices

- Bastard child of a LISP machine and a piece of paper

- Basically a computer model that uses:
  - The axioms of origami to compute results and construct internal representations
  - Some basic geometry queries to make decisions
  - LISP/Scheme list processing capabilities

Internal representation

- Uses an internal Doubly Connected Edge List
- Uses the list primitives of LISP and Scheme

Note this is not a complete computer! Missing are

- ways to represent code
- input/output facilities
- complete 'assembly language' definition

Some definitions

Point (Vertex) Euclid's definition: *A point is that which has no parts, or which has no magnitude*. We will use the following *A point is the minimal addressable unit of the OCM*

Line A maximal set of point that the OCM considers to be colinear

Edge A continuous subset of a line

Face The intersection of the half-planes given by a set of lines and directions

Queries & Data manipulation

- *LIS*t *P*rocessing features
  - *car*, *cdr*, *append*
  - *cons*, *list*

- State queries
  - Left/right turn (or colinearity) check on three points
  - Left/right relative position of a point to a line – can 'intersect'
  - Feasibility of various axioms of Origami

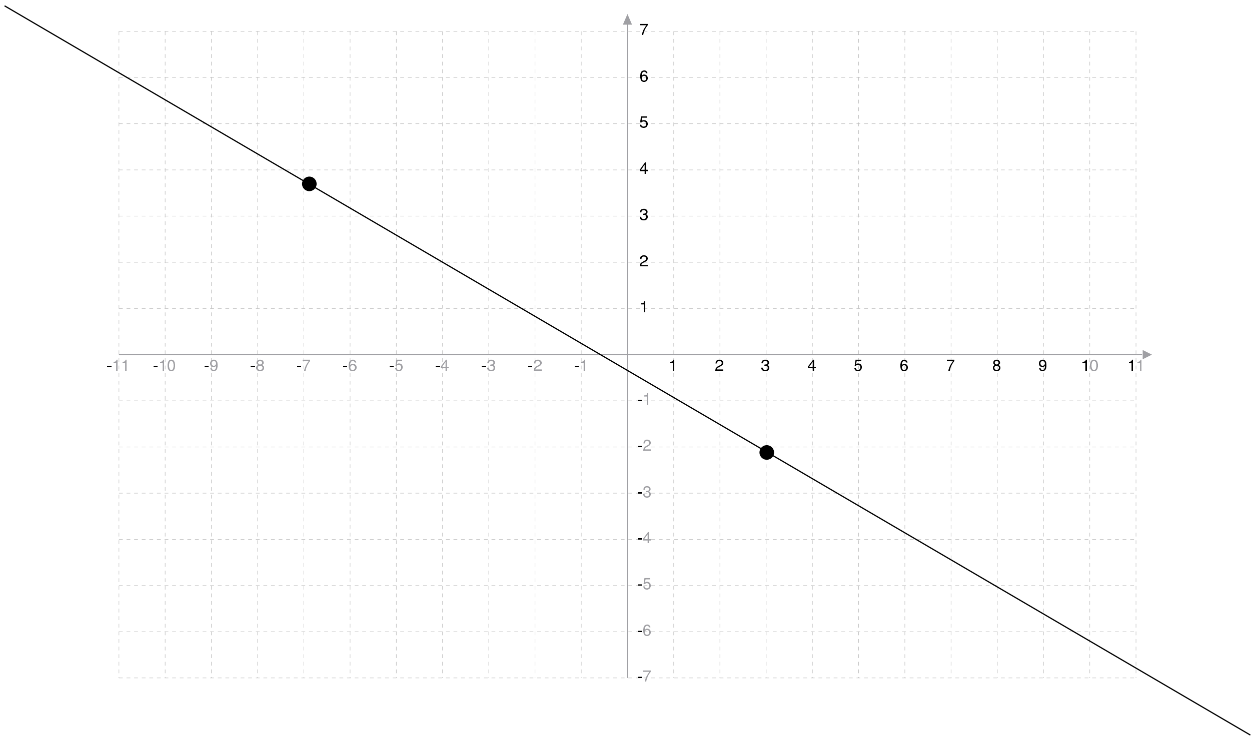
Axioms of Origami

- As stated, mathematically defined operations on points and lines

- Quick list:

  1. Folding through 2 points, or through a line
  2. Folding a point on a point
  3. Folding a line on a line
  4. Folding a line on itself through a point
  5. Folding a point on a line through a point
  6. Folding 2 points on 2 different lines

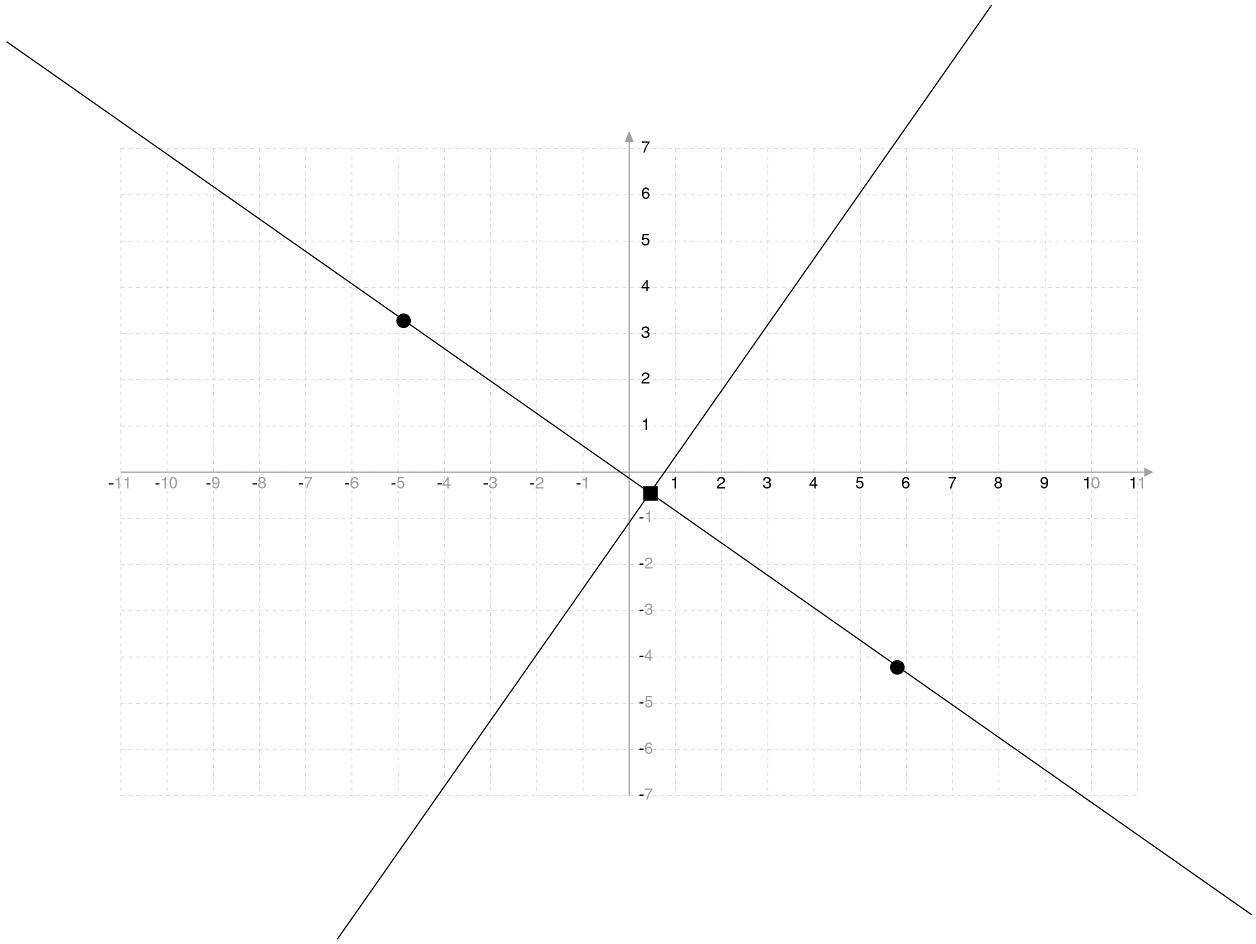
Axiom 1 – Folding through 2 points, or through a line



$$a = \frac{y_2 - y_1}{x_2 - x_1}$$

$$b = y_1 - a * x_1$$

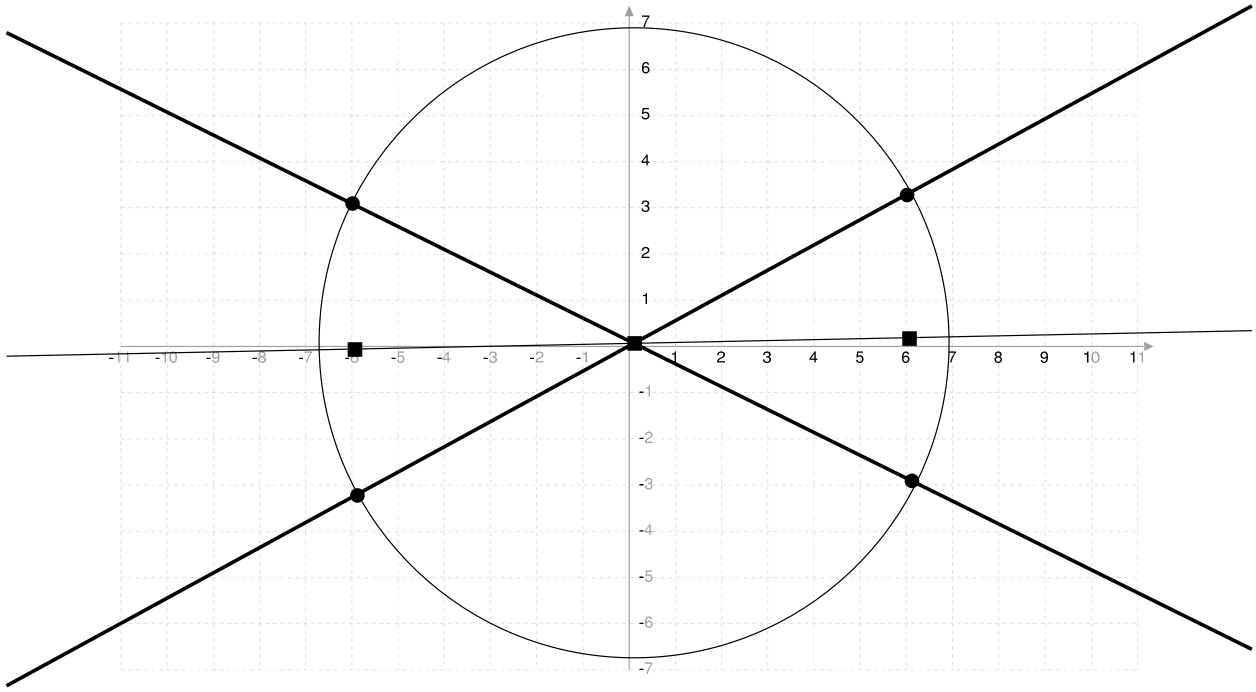
Axiom 2 – Folding a point on a point



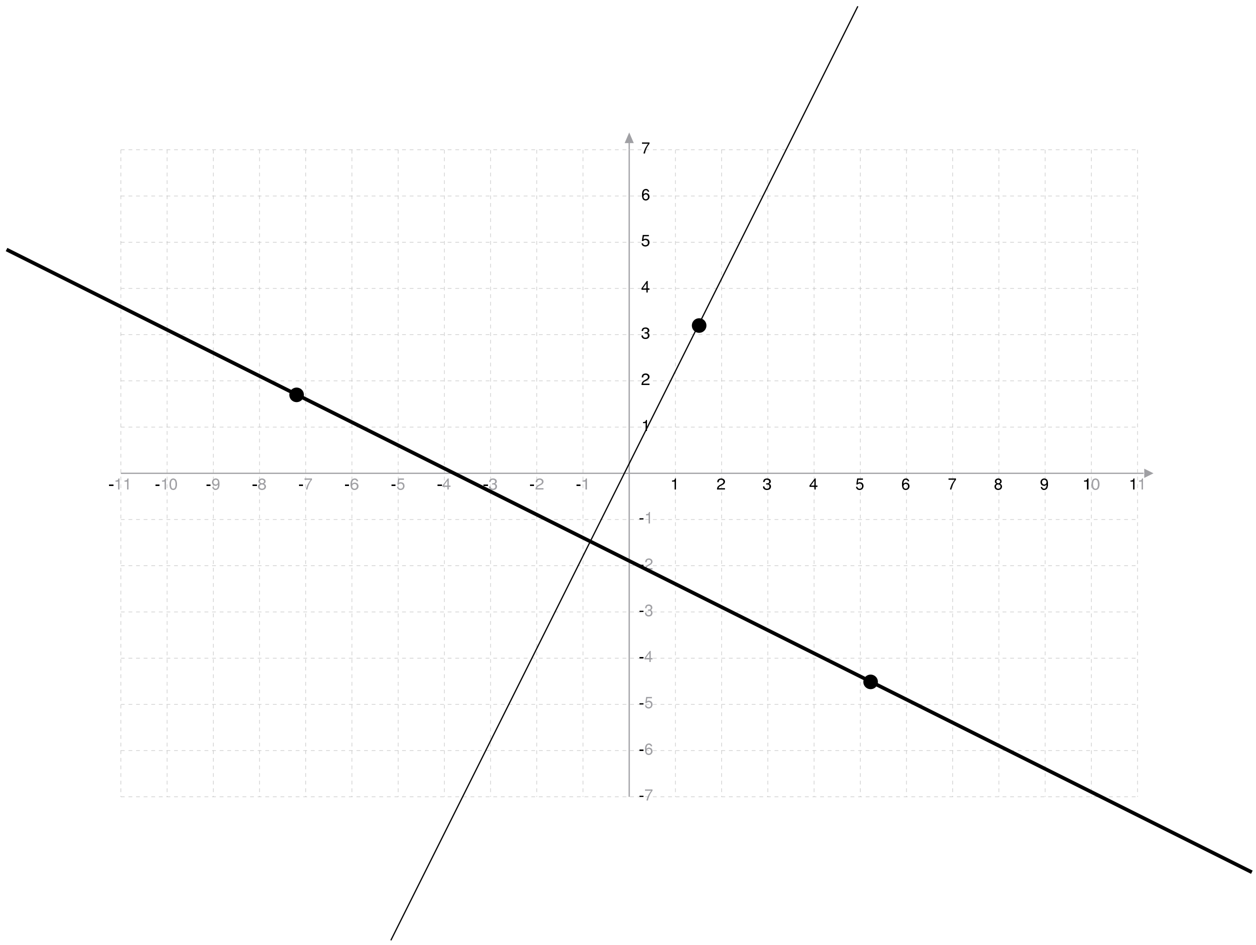
$$a' = -\left(\frac{y_2 - y_1}{x_2 - x_1}\right)^{-1}$$

$$b' = \frac{y_2 - y_1}{2} - a' * \frac{x_2 - x_1}{2}$$

Axiom 3 – Folding a line on a line



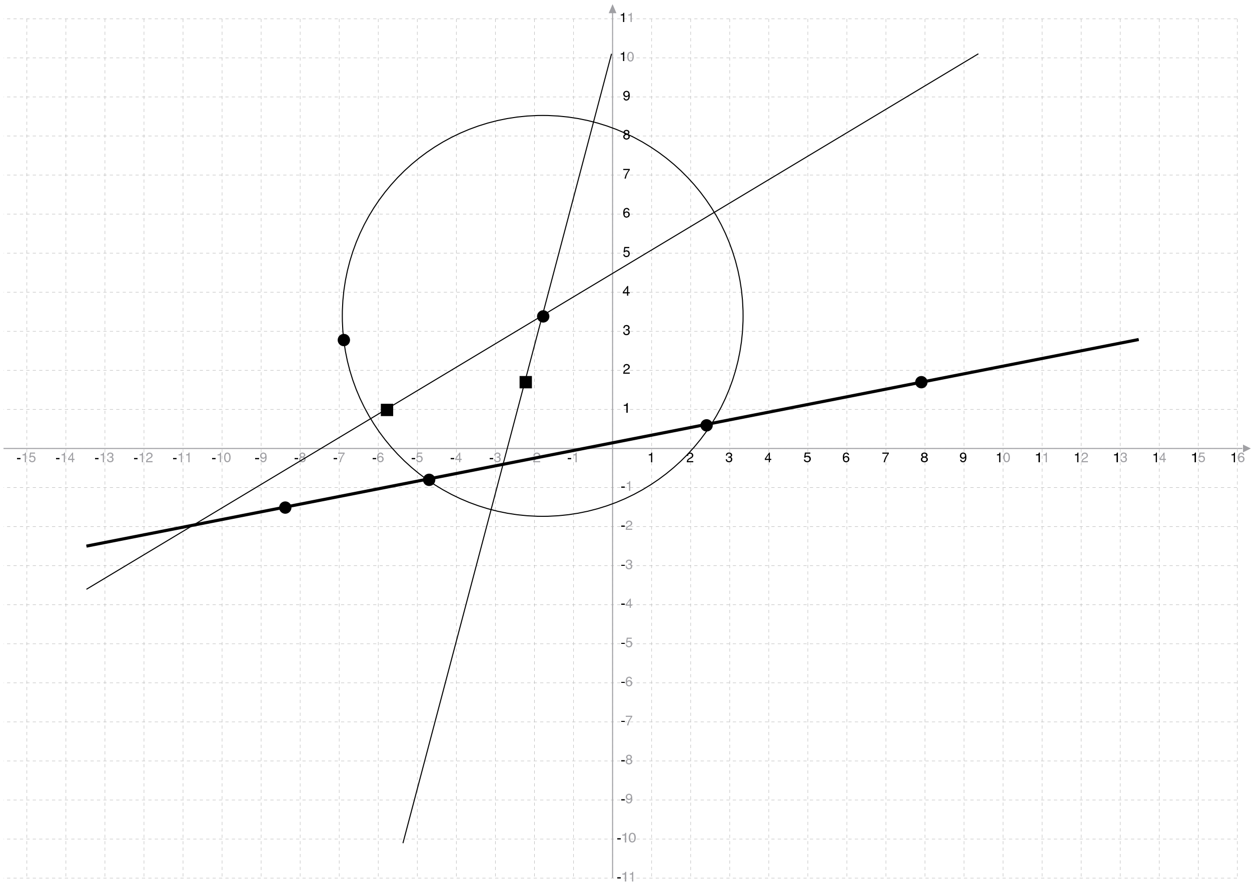
Axiom 4 – Folding a line on itself through a point



$$a^{'} = \frac{1}{-a}$$

$$b^{'} = y - a^{'} * x$$

Axiom 5 – Folding a point on a line through a point



Axiom 5 – Folding a point on a line through a point

A couple of notes on axiom 5

- It's mostly the direct equivalent of Edge & Compass's Compass
- It allows one to solve 2nd degree equations, and thus build numbers which are powers of 2

$x_3 = \frac{2x_2 - 2ab + 2y_2a + 2SQRT}{2(1+a^2)}$

$y_3 = \frac{a(2x_2 - 2ab + 2y_2a + 2SQRT)}{2(1+a^2)} + b$

OR

$x_3 = \frac{2x_2 - 2ab + 2y_2a - 2SQRT}{2(1+a^2)}$

$y_3 = \frac{a(2x_2 - 2ab + 2y_2a - 2SQRT)}{2(1+a^2)} + b$

Where SQRT is defined to be:

$$SQRT = \sqrt{A + B + C + D}$$

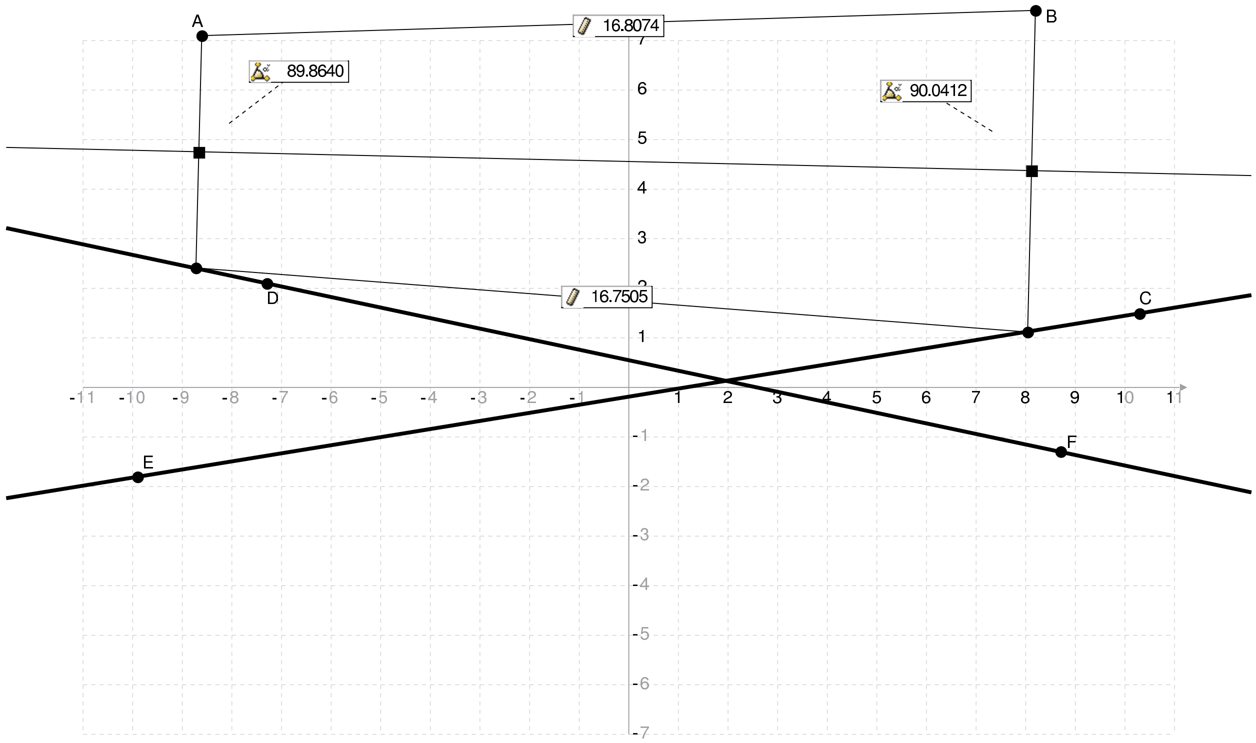
$$A = x_2^2 - 2x_2ba + 2x_2y_2a + y_2^2a^2$$

$$B = x_1^2 - 2x_1x_2 + y_1^2 - 2y_1y_2$$

$$C = -b^2 + 2y_2b + x_1^2a^2 - 2x_1x_2a^2$$

$$D = y_1^2a^2 - 2y_1y_2a^2$$

Axiom 6 – Folding 2 points on 2 different lines



Axiom 6 – Folding 2 points on 2 different lines

A couple of notes on axiom 6

- It allows one to solve 3rd degree equations, and thus build numbers which are powers of 3
- This one is the interesting one... it's impossible in Edge & Compass!

$$y_1' = a_1 x_1' + b_1$$

$$y_2' = a_2 x_2' + b_2$$

$$\frac{y_1 - y_1'}{x_1 - x_1'} = \frac{y_2 - y_2'}{x_2 - x_2'}$$

$$(x_1 - x_2)^2 + (y_1 - y_2)^2 = (x_1' - x_2')^2 + (y_1' - y_2')^2$$

Axiom 6 – solution

Which solves to

$$x_1^{'} = \frac{-a_2+x1-b_1+b_2-y_2+y_1+x_2a_2}{-a_1+a_2}$$

$$y_1^{'} = \frac{-a_2a_1x_1+a_1b_2+a_1y_2+a_1y_1+a_2x_2a_1-a_2b_1}{-a_1+a_2}$$

$$x_2^{'} = \frac{-b_1+b_2+y_1-y_2+a_1x_1+x_2+a_1}{-a_1+a_2}$$

$$y_2^{'} = \frac{-a_2b_1+a_2y_1+a_2y_2-a_2a_1x_1+a_2x_2a_1+a_1b_2}{-a_1+a_2}$$

And now the fun stuff: What can we do with it?

- Convex hulls
  - Jarvis' March
  - Graham's Scan?
- Voronoi Diagrams
  - Hard!
- Linear Programming
  - Two dimensional linear programming in $O(N)$

Jarvis' March

```
(define jarvis-march-internal
  (lambda (stop-point first-point second-point)
    (let ((next-point (jarvis-next first-point second-point)))
      (if (= next-point stop-point)
        next-point
        (cons next-point
(jarvis-march-internal stop-point second-point next-point))))))

(define jarvis-march
  (lambda (points-cloud)
    (let ((first-point (find-lowest-y points-cloud))
          (second-point (jarvis-next lower-right-corner first-point)))
        (cons first-point
(jarvis-march-internal first-point first-point second-point)))))
```

Graham's scan

- Possible, but we need to figure out how to sort
- Sorting on linked lists: merge sort, or $O(N^2)$ sorts
- Then use the 3 coins-algorithm as is
- Without explicit stacks, surprisingly hard

Linear Programming & Voronoi Diagrams

Linear programming:

- Class algorithm works as is!
- But blows up badly

Voronoi Diagrams:

- Class algorithm works as is!
- But blows up

Comments on generic techniques

Randomization No change, but need to code a new pseudo random number generator

Linesweep Works perfectly, remember we can sort on x or y coordinates

Duality Hard to represent as is, as the model is often clutered with random lines, but most certainly feasible

Linear Programming Feasible, as we saw, but most advanced algorithms start to clutch at straws when brought into the OCM

Parallelization Very hard to create a working OCM model for parallel algorithms: folds almost always intersect, and thus very few query we make are unaffected by the other 'threads'

Origami contra Edge & Compass

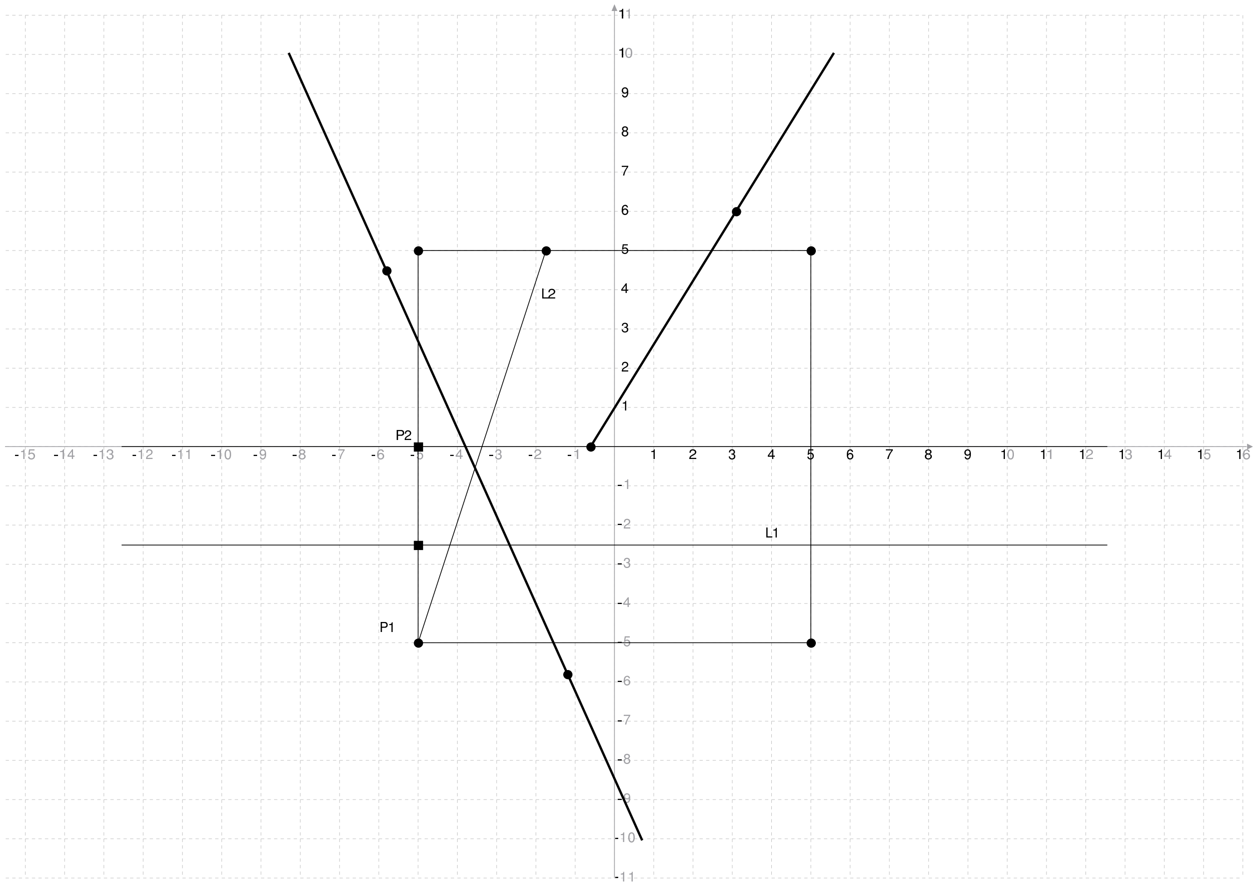
- Origami is better!

- Edge & Compasscan do *almost* all the axioms of Origami: it cannot do Axiom 6: Two lines on two points

- Origami can trisect an angle!

- Origami still cannot square the circle

Technical digression: Angle trisecting



Origami contra Turing Machines & RAM Models

- OCM is not a Turing Machine
  - No notion of tape – data enters the OCM magically
  - Number of state is infinite
- But is the OCM Turing Complete?
- OCM is not remotely a superset of the RAM Model
- OCM doesn't have arithmetic capabilities
- OCM doesn't have pointers & other bit-representations
- For example, OCM cannot do hashing (no $O(1)$ floor function)!

  - Could always use segment intersection and judiciously placed test segments, but this is $O(N)$ in memory, and $O(N)$ in time for setup
  - And we need trigonometry to calculate the segments' positions right, which the OCM won't do for us

- Origami is asymptotically faster than current implementations of the RAM model in some cases though:

  - The OCM can create $O(N)$ lists in $O(1)$
  - The OCM doesn't need $O(bit - size)$ algorithms for calculating the folds' slopes

Thus...

- Origami is better than Edge & Compass, but not necessarily a superset
- OCM is very much an Algebraic Decision Tree
- Turing Machines and RAM Models are better at most things, but the OCM has an infinite parallel DCEL evaluation engine

What's the point?

- Very abstract machine, few direct practical applications
- But Computational Models study is very important!
  - Jeff Erickson has proven that in his model, based on Edge & Compass, P != NP
  - Other models are is use out there, for the same reasons other geometries are out there: Problem context

What's next?

- Applications: Topology
- Extensions: *The Third Dimension* & Fold angles
- Applications: Raytracing

Building a real one

Quite a couple of problems:

- The 'infinite parallel DCEL engine' is impossible, but we can simply invent a new algorithm measurement
- Colinearity happens all the time – infinite precision numerics don't exist