Formal study of plane Delaunay triangulation

Jean-François Dufourd[1] Yves Bertot[2]

[1]LSIIT, UMR CNRS 7005, Université de Strasbourg, France

[2]INRIA, Centre de Sophia-Antipolis Méditerranée, France
(Thanks: French ANR "white" project GALAPAGOS)

ITP 2010, Edinburgh, July 11-14, 2010

Outline

Introduction

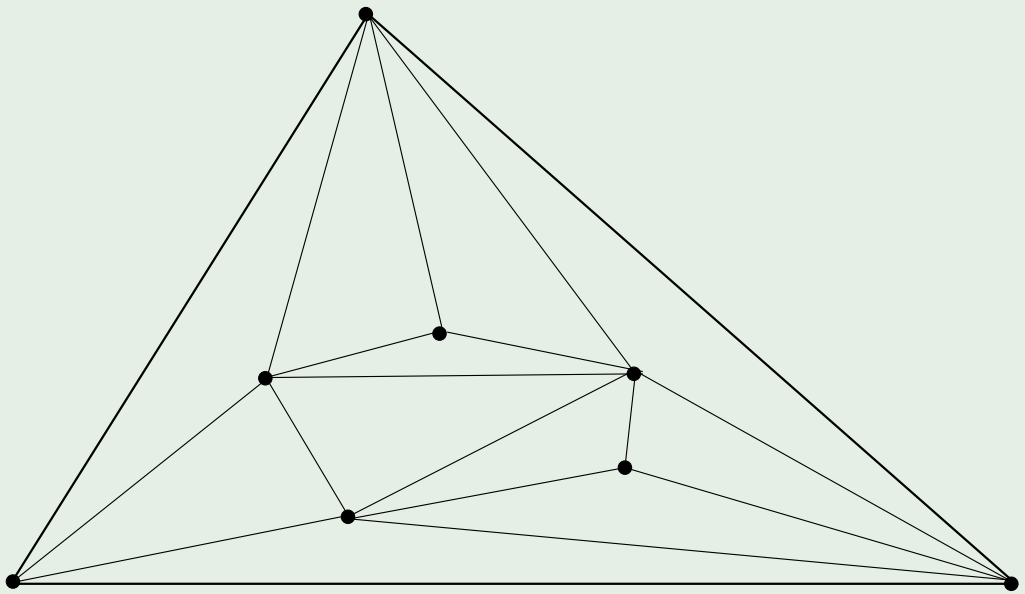
- *Aim*:
  - Building a framework for modeling, reasoning and programming with surface subdivisions (polyhedral surfaces, or polyhedra)
- *Means*:
  - Formal specifications in the Calculus of Inductive Constructions
  - Interactive proofs in the Coq proof assistant (INRIA)
  - A combinatorial hypermap model of surface subdivisions
- *"Benchmark" of good size*:
  - An algorithm of Delaunay triangulation (a "corner stone" in computational geometry)

Related work

- Planar graphs and triangulations formalized in Isabelle [G. Bauer & T. Nipkow, 2003]
- Combinatorial maps and hypermaps [W.T. Tutte, R. Cori..., 1970-]
- Specification of hypermaps and proof in Coq of the Four Colour Theorem [G. Gonthier et al., 2005]
- Hypermap specification and proof in Coq of Genus Theorem and Euler Formula [J.-F. Dufourd, 2006]
- Proof in Coq of a discrete Jordan Curve Theorem [J.-F. Dufourd, 2009]
- Formalized proofs of convex hull algorithms [D. Pichardie & Y. Bertot, 2001, L.I. Meikle & J. Fleuriot, 2004, C. Brun et al., 2009]
- Studies on Delaunay triangulation [L. Guibas & J. Stolfi, 1985, D.E. Knuth, 1992, H. Edelsbrunner 2000...]

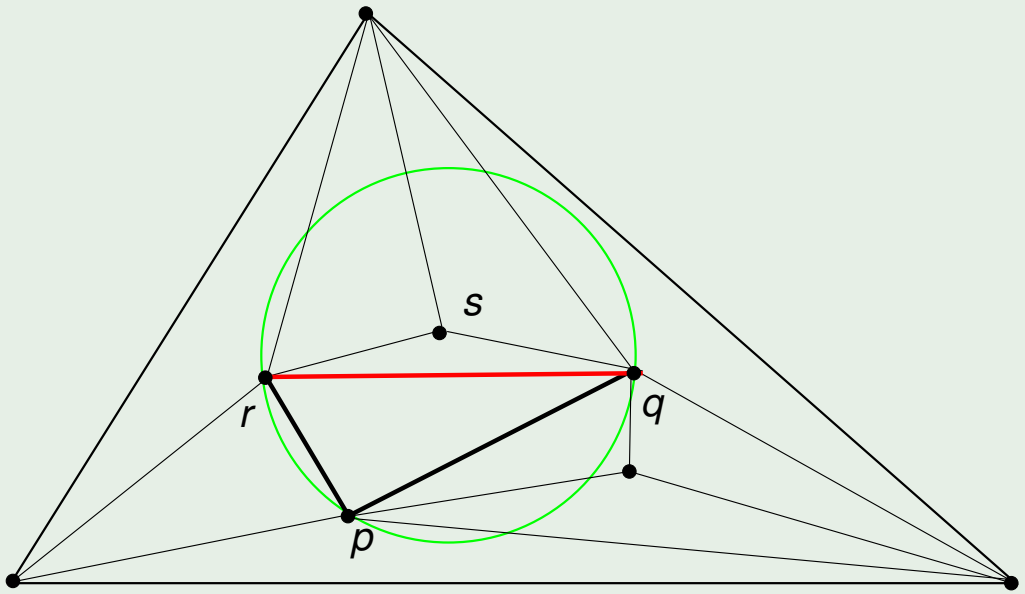
What is the problem?

Informal algorithm: Starting with a plane triangulation



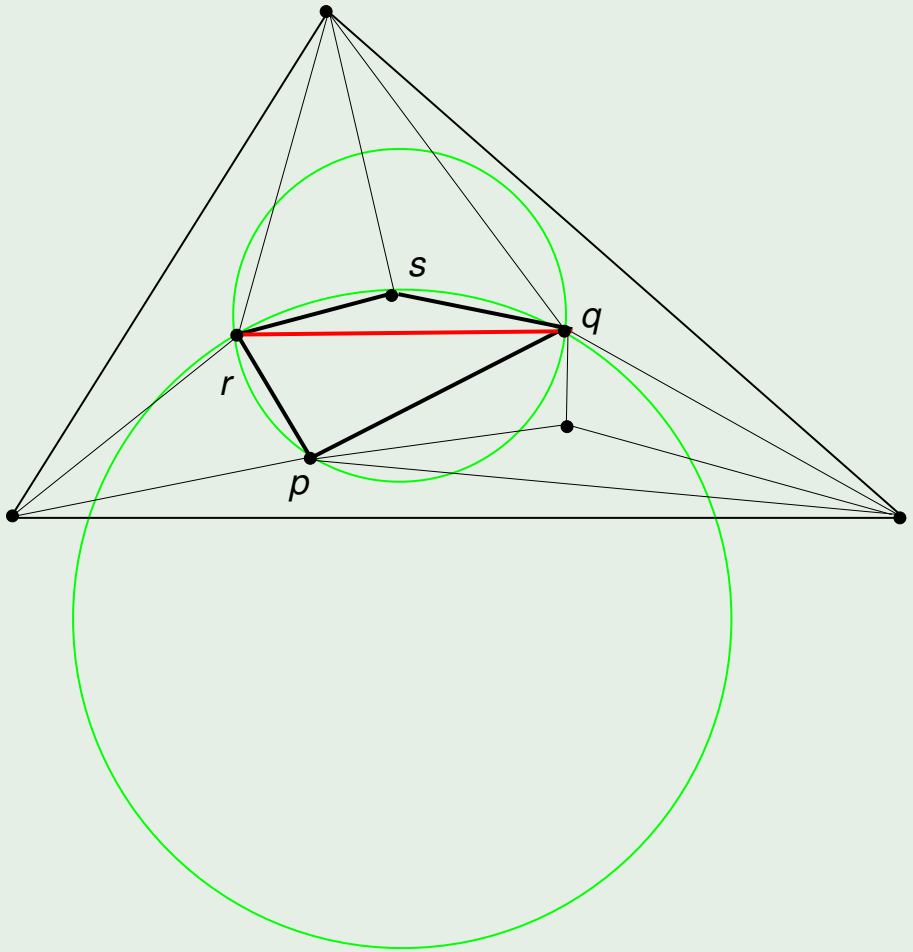
A plane triangulation

Detecting an illegal edge if any: circumcircle criterion



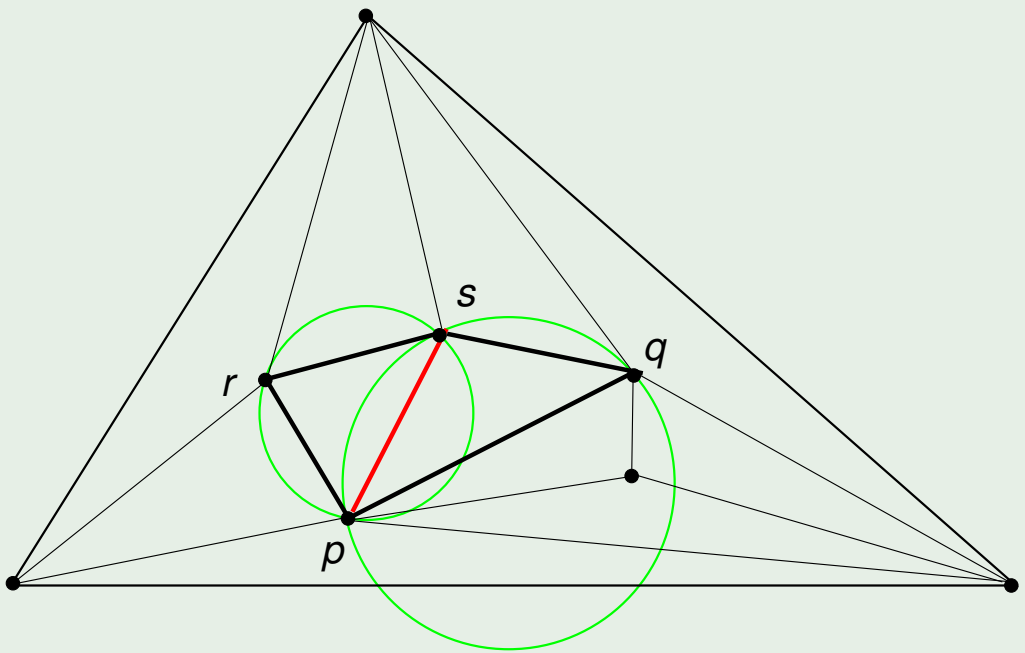
s is in the circumcircle of triangle (p, q, r): rq is illegal

Symmetry between the 2 adjacent triangles



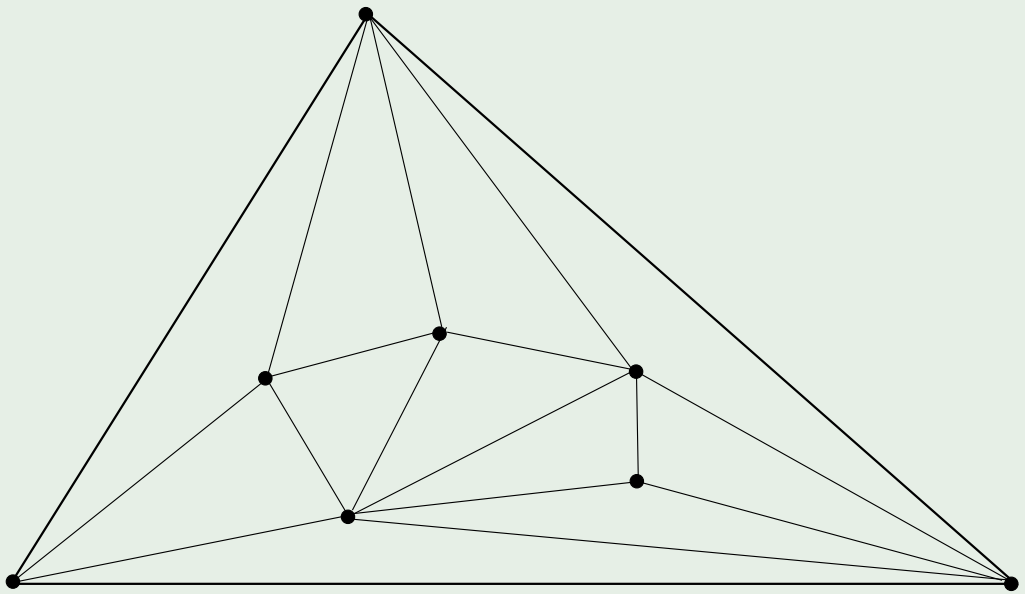
p is also in the circumcircle of (q, s, r)

Flipping the common edge



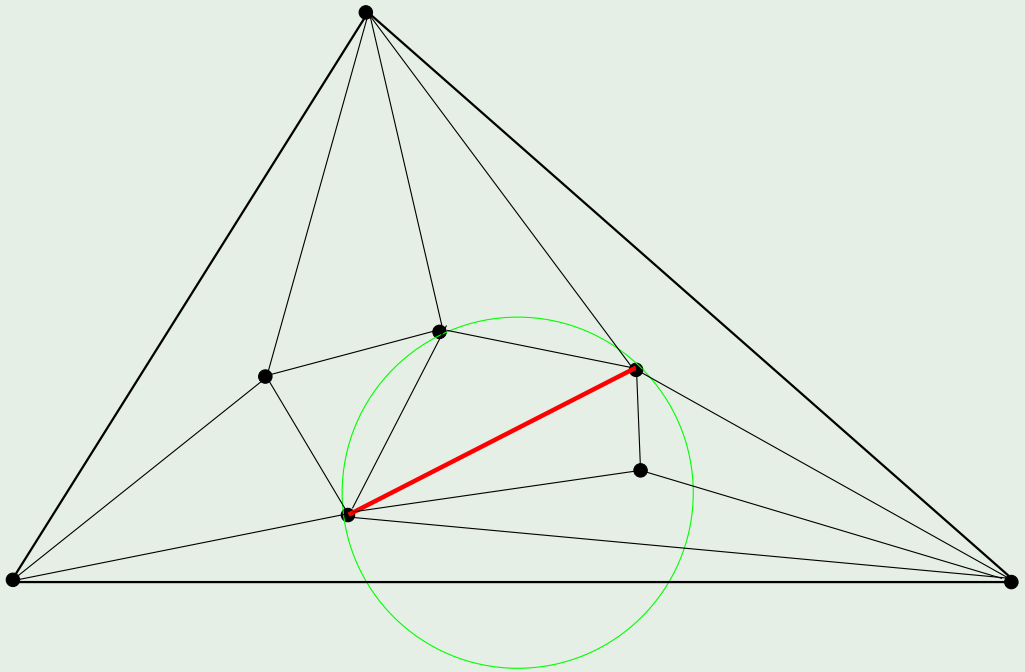
Swapping *rq* and *ps*

New triangulation



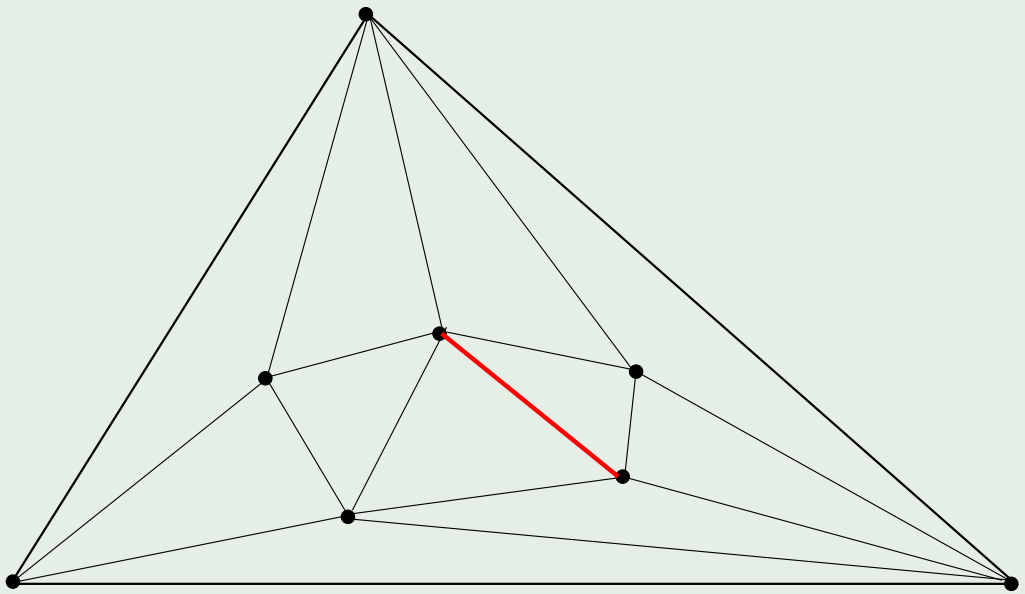
New triangulation

Iterating until there is no more illegal edge



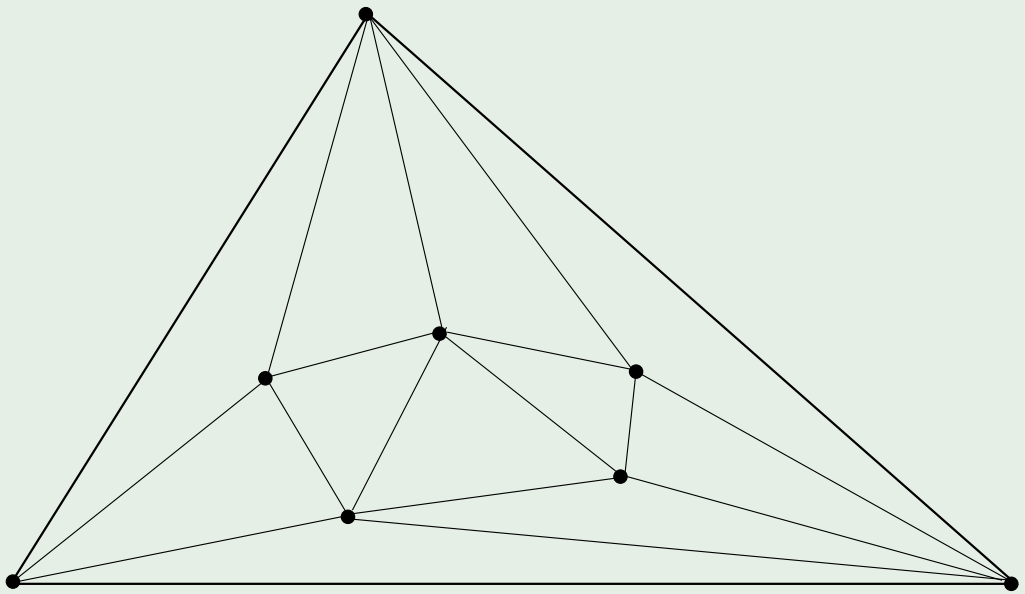
Another illegal edge

Iterating until there is no more illegal edge



Flipping the illegal edge

Delaunay triangulation



Delaunay triangulation: no illegal edge

Questions:

- How to formalize plane subdivisions / triangulations?
- How to define edge flipping and to prove its correctness?
- How to specify edge flipping sequences and to prove their termination and partial correctness?

Hypermaps in Coq

Definition (*hypermap*)

A *hypermap* is an algebraic structure $M = (D, \alpha_0, \alpha_1)$, where D is a finite set, the elements of which are called *darts*, and α_0, α_1 are permutations on D.

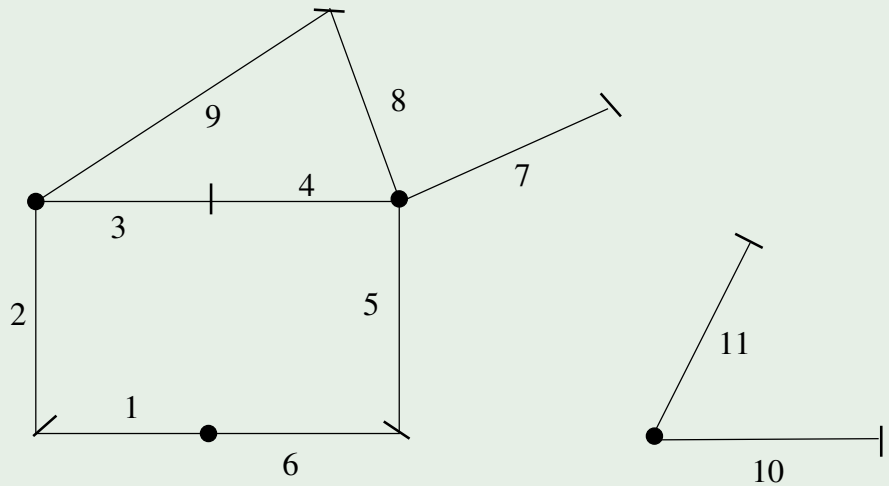
Definition (*Topological cells, orbits*)

The *topological cells* of a dart x in a hypermap are dart sets – named *orbits* – which are traversed while iterating some operations from x:

- *edge, vertex, face* of x: iteration of α_0, α_1, $\phi = \alpha_1^{-1} \circ \alpha_0^{-1}$;
- *connected component* of x: iteration of both α_0 and α_1.

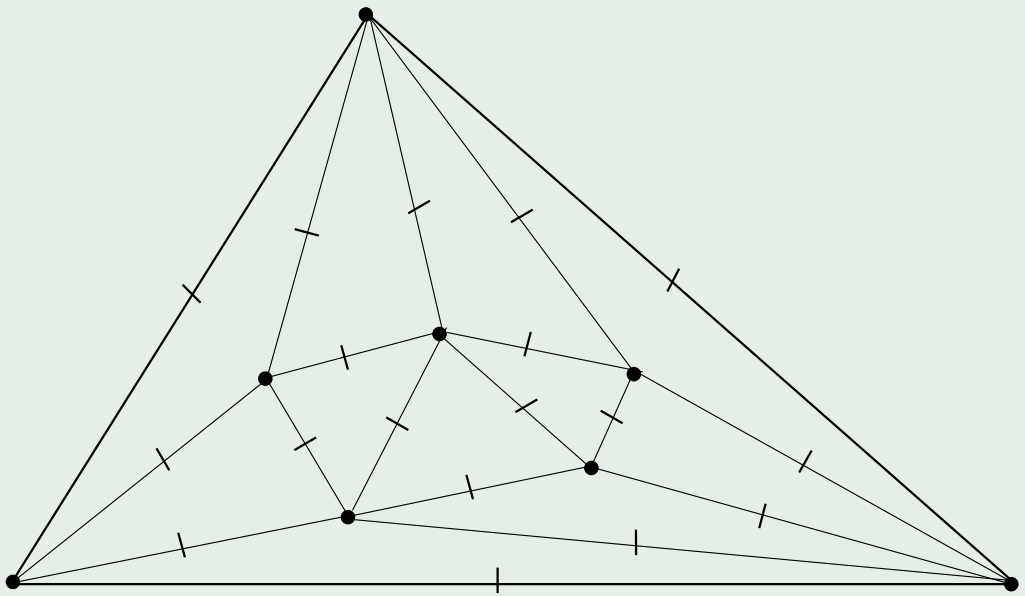
Example: A hypermap embedded onto the plane

D	1	2	3	4	5	6	7	8	9	10	11
α0	2	1	4	3	6	5	7	9	8	10	11
α1	6	3	9	5	7	1	8	4	2	11	10



11 darts (*half-line segments*), 7 edges (*strokes*), 4 vertices (*bullets*), 4 faces, and 2 connected components. (Convention: α_0, α_1, ϕ "turn" counterclokwisely)

Example: A hypermap for the former triangulation



Hypermap specification in Coq

Inductive definition of a type `fmap` *of* free maps *embedded in the plane (free algebra of terms)*

```
Inductive fmap:Set:=
    V : fmap
  | I : fmap->dart->point->fmap
  | L : fmap->dim->dart->dart->fmap.
```

Observers of a free map, inductively defined

- *Existence of a dart* `z` in a free map `m`: `(exd m z)` .
- *Operation* α_k: `(cA m k z)` is the *k-successor* of `z` in `m`.
- *Operation* ϕ: `(cF m z)` is the *face-successor* of `z` in `m`.
- *Vertex point* of `z` in `m`: `(fpoint m z)`.
- *Inverses*: `cA_1`, `cF_1`...

Examples in the previous hypermap

- `exd m 2, ~exd m 15`
- `cA m zero 1 = 2, cA m one 2 = 3`
- `cF m 6 = 4, cF m 2 = 4`
- `cA_1 m one 3 = 2, cF_1 m 4 = 2`

Destructors in a free map

- *Deletion* `(D m z)` of a dart `z` in `m`
- *Break* `(B m k z)` in `m` of a link starting from `z` at dimension `k`
- *Inverse* of `B`: `B_1`

Hypermaps as a subtype of the free maps

After preconditions for `I` and `L`, an *invariant* so that `m` is a hypermap is defined by filtering:

```
Fixpoint inv_hmap(m:fmap):Prop:= match m with
   V => True
 | I m0 x _ =>     inv_hmap m0 /\ prec_I m0 x
 | L m0 k0 x y => inv_hmap m0 /\ prec_L m0 k0 x y
  end.
```

Some proven properties

- For any (true) hypermap `m` and dimension `k`, `cA` is a *permutation* and `cA_1` is its *inverse*.
- Idem for `cF` and `cF_1`.

Orbits

- The fact that two darts `x` and `y` belong to the same *edge, vertex, face, connected components* of `m` is denoted by: `expe m x y, expv m x y, expf m x y, eqc m x y` (resp.). These relations are proven to be equivalences.
- The number of darts in an orbit of `x` in `m` is its *degree*: e.g. *for an edge, a vertex, a face*: `degreee m x, degreev m x, degreef m x`. The degree is proven to be the same for each dart of an orbit.
- The *numbers* of *edges, vertices, faces, components* are easily defined. Then, the *planarity* predicate `(planar m)` of a hypermap `m` is defined thanks to the *Euler formula*.

Splitting a vertex

Splitting a k-orbit

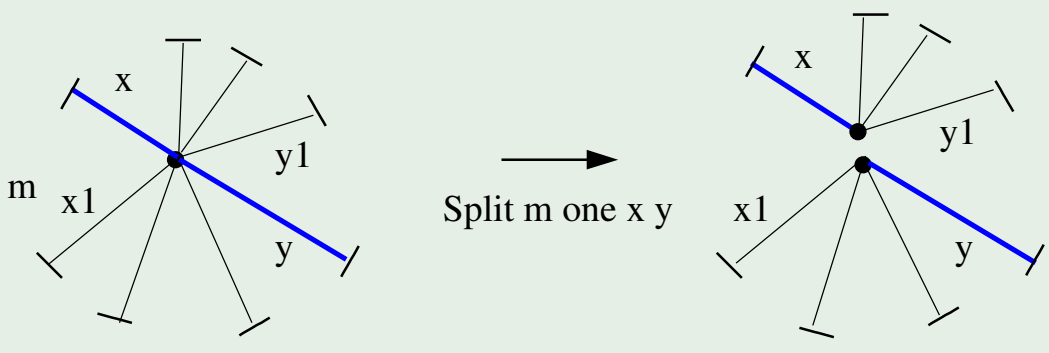
```
Definition Split(m:fmap)(k:dim)(x y:dart):fmap:=...
```

Splitting a vertex

Precondition:

```
exd m x /\ exd m y /\ x <> y /\ expv m x y
```

Example



Merging two vertices

Merging two k-orbits

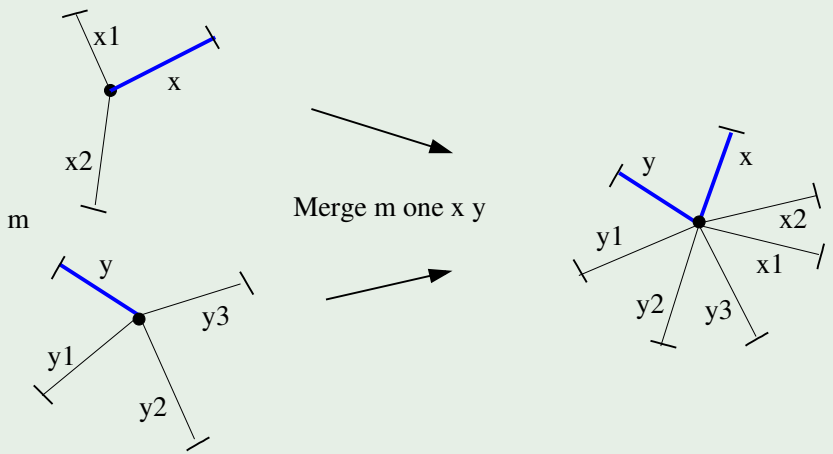
```
Definition Merge(m:fmap)(k:dim)(x y:dart):=...
```

Merging two vertices

Precondition:

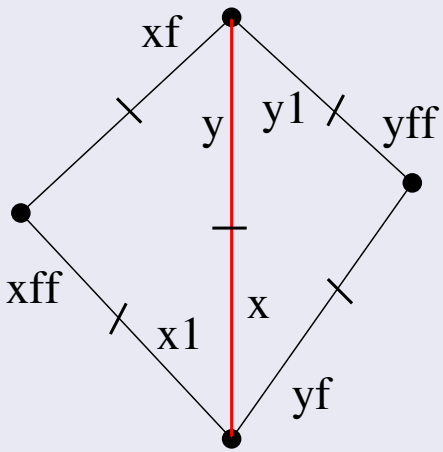
```
exd m x /\ exd m y /\ ~expv m x y
```

Example



Flipping an edge

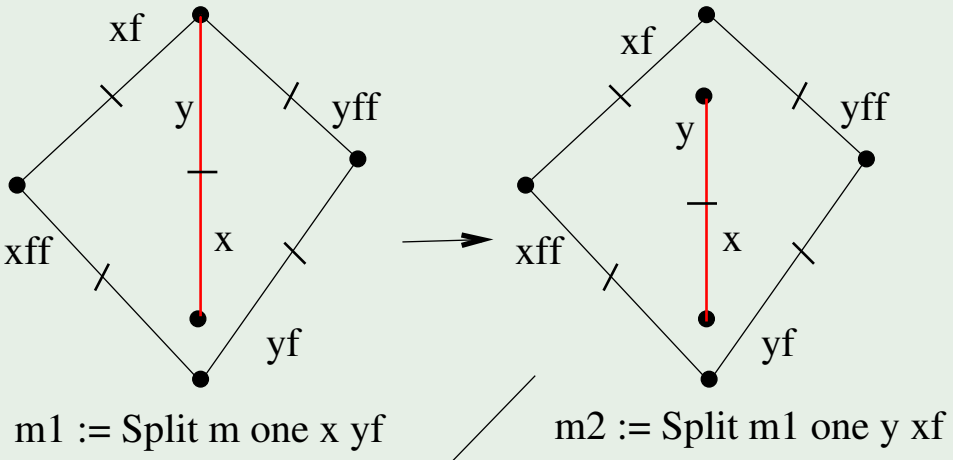
Flip configuration



Precondition:

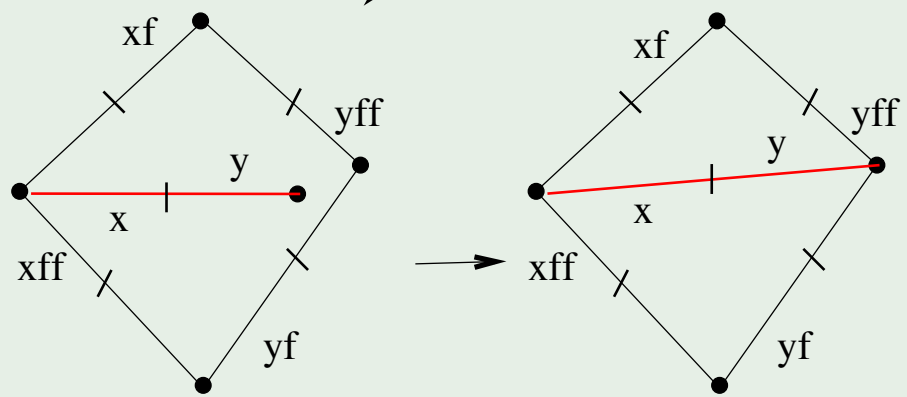
```
Definition prec_Flip(m:fmap)(x:dart):Prop:=
  let y:= cA m zero x in
   exd m x /\ ~ expf m x y /\ ~ expv m x y /\
     3 <= degreev m x /\ 3 <= degreev m y.
```

Flip illustration



m1 := Split m one x yf

m2 := Split m1 one y xf



m3 := Merge m2 one xff x

m4 := Merge m3 one yff y

Flip definition

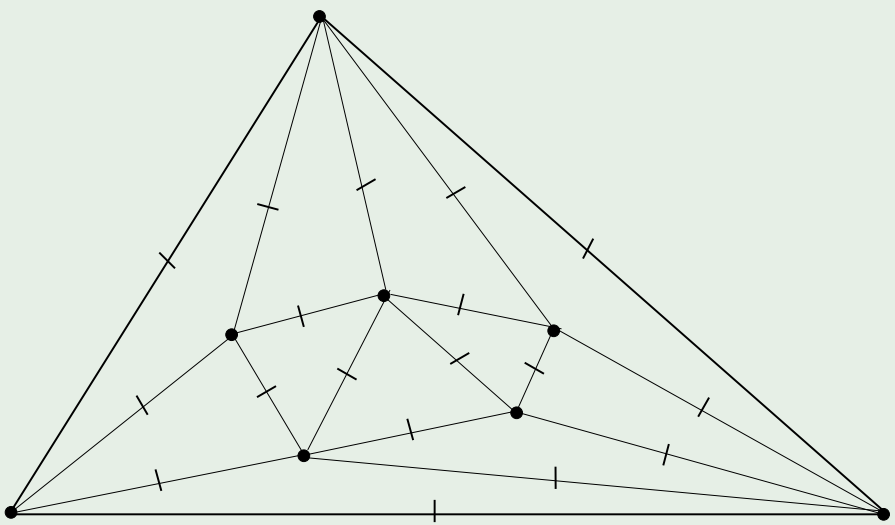
```
Definition Flip(m:fmap)(x:dart):fmap:=
  let y := cA m zero x in
  let xf := cF m x in
  let xff := cF m xf in
  let yf := cF m y in
  let yff := cF m yf in
  let pxff := fpoint m xff in
  let pyff := fpoint m yff in
  let m1 := Split m one x yf in
  let m2 := Split m1 one y xf in
  let m3 := Merge m2 one xff x in
  let m4 := Merge m3 one yff y in
  let m5 := Chp m4 x pxff in
  let m6 := Chp m5 y pyff in
in m6.
```

Flipping in a topological triangulation

Definition (*Topological triangulation*)

A *topological (uniform) triangulation* is a hypermap such that:
(i) each *edge* contains exactly two darts;
(ii) each *vertex* contains at least two darts;
(iii) each *face* contains exactly three darts, i.e. is a *triangle*.

Illustration



A (connected) triangulation

Formalization

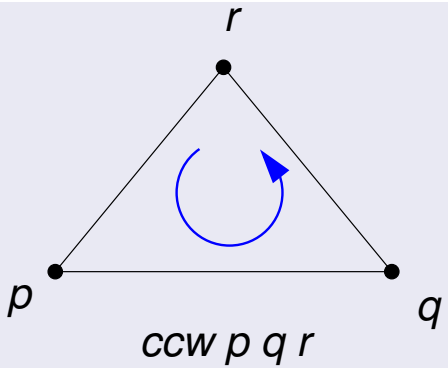
```
Definition inv_Triangulation(m:fmap):Prop:=
  inv_hmap m /\ isMap m /\ isPoly m /\
     isTriangulation m.
```

Topological properties: invariant, planarity

```
Theorem inv_Triangulation_Flip:
 forall(m:fmap)(x:dart),
   inv_Triangulation m -> prec_Flip m x ->
      inv_Triangulation (Flip m x).

Theorem planar_Flip:
  forall(m:fmap)(x:dart),
   inv_Triangulation m -> prec_Flip m x ->
     planar m -> planar (Flip m x).
```

Flipping in a plane embedded triangulation

Triangle counterclockwise orientation: `ccw` *predicate, Knuth's axioms*



Definition of `ccw` *in the Euclidean plane model*

- Triangle (p, q, r) is *counterclockwise oriented iff* the following determinant is positive:

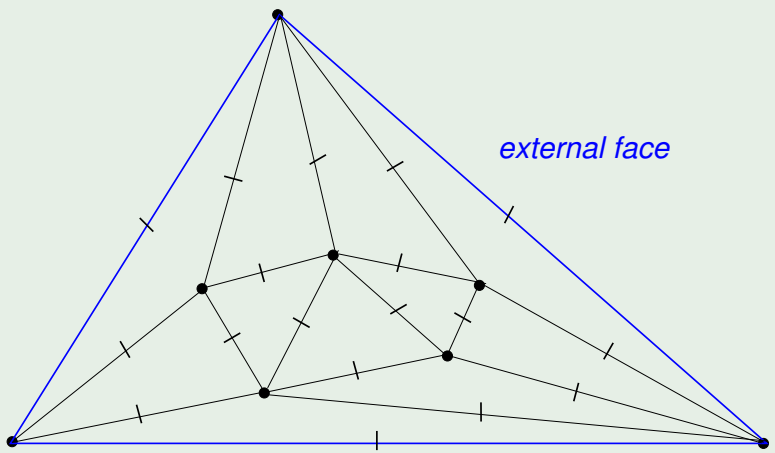
$$\begin{vmatrix} x_p & y_p & 1 \\ x_q & y_q & 1 \\ x_r & y_r & 1 \end{vmatrix}$$

- Then, *Knuth's 5 axioms* on `ccw` are easily proven.

Definition (*Well-embedded triangulation*)

A topological triangulation is *well-embedded on the plane* if:
(i) the (two) darts of each *edge* are embedded on distinct points;
(ii) the darts of each *vertex* are embedded on the same point;
(iii) the (three) darts of each *face* are embedded as a counterclockwise oriented triangle, except for one of them – the *external face* – which is embedded as a clockwise oriented triangle.

Illustration



A (connected) triangulation (with a triangular external face)

Translation in Coq's hypermap

```
Definition isWellembed(m:fmap):Prop:=
 exd m 1 /\ isWellembede m /\
   isWellembedv m /\ isWellembedf m.
```

Precondition of Flip enforced

```
Definition prec_Flip_emb(m:fmap)(x:dart):Prop:=
   let y := cA m zero x in
   let px := fpoint m x in
   let py := fpoint m y in
   let pxff := fpoint m (cF m (cF m x)) in
   let pyff := fpoint m (cF m (cF m y)) in
      ccw px py pxff /\ ccw py px pyff /\
         ccw px pyff pxff /\ ccw py pxff pyff.
```

Preservation of a well-embedded triangulation by `Flip`

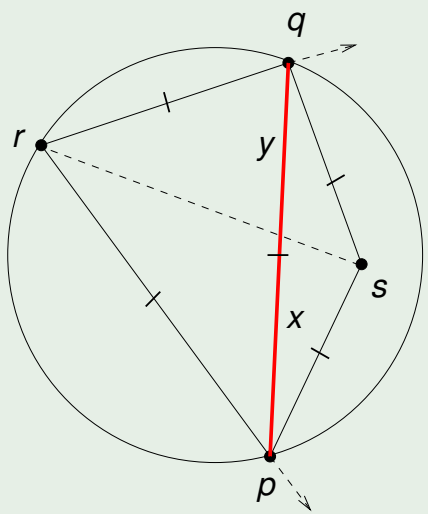
```
Theorem prec_Flip_emb_prec_Flip:
 forall (m:fmap)(x:dart),
  inv_Triangulation m -> isWellembed m ->
    exd m x -> prec_Flip_emb m x ->
       prec_Flip m x.

Theorem isWellembed_Flip:
 forall (m:fmap)(x:dart),
  inv_Triangulation m -> isWellembed m ->
    exd m x -> prec_Flip_emb m x ->
     isWellembed (Flip m x).
```

Delaunay criterion

Definition (*Illegal edge in a triangulation*)

An edge is *illegal* in a well-embedded plane triangulation when:
(i) its two incidental triangles are counterclockwise oriented (which excludes the external face);
(ii) the vertex of one of the two triangles which is not extremity of the common edge is strictly in the other triangle circumcircle.

Illustration



Point s is in the circumcircle of (p, q, r)

`incircle` *predicate*

Point s is in the circumcircle of (p, q, r) *iff* the following determinant is positive:

$$\begin{vmatrix} x_p & y_p & x_p^2 + y_p^2 & 1 \\ x_q & y_q & x_q^2 + y_q^2 & 1 \\ x_r & y_r & x_r^2 + y_r^2 & 1 \\ x_s & y_s & x_s^2 + y_s^2 & 1 \end{vmatrix}$$

`illegal` *predicate and property*

```
Lemma illegal_prec_Flip_emb:
  forall (m:fmap)(x:dart),
   illegal m x -> prec_Flip_emb m x.
```

Properties of Flip

Properties after `Flip`

- (1) The two new triangles are ccw-oriented:

```
Theorem ccw_exchange: forall (p q r s: point),
  ccw p q r -> ccw q p s -> incircle p q r s ->
        (ccw r s q /\ ccw s r p).
Proof.
  (* By polynomial algebra, using Ring
   and Stengle's Positivstellensatz in R *)
Qed.
```

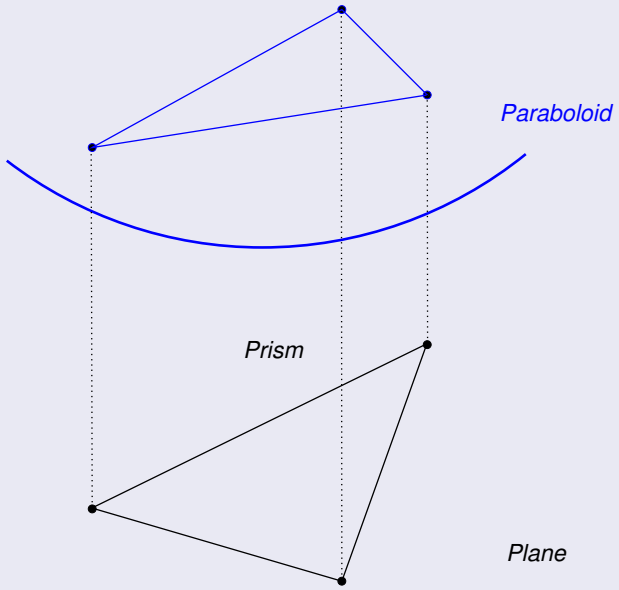
- (2) Their common edge is legal:

```
Theorem ccw_exchange: forall (p q r s: point),
  incircle p q r s -> ~incircle p q s r.
Proof.
  (* By exchange of determinant lines *)
Qed.
```

Termination based on finiteness

A strict order on triangulations

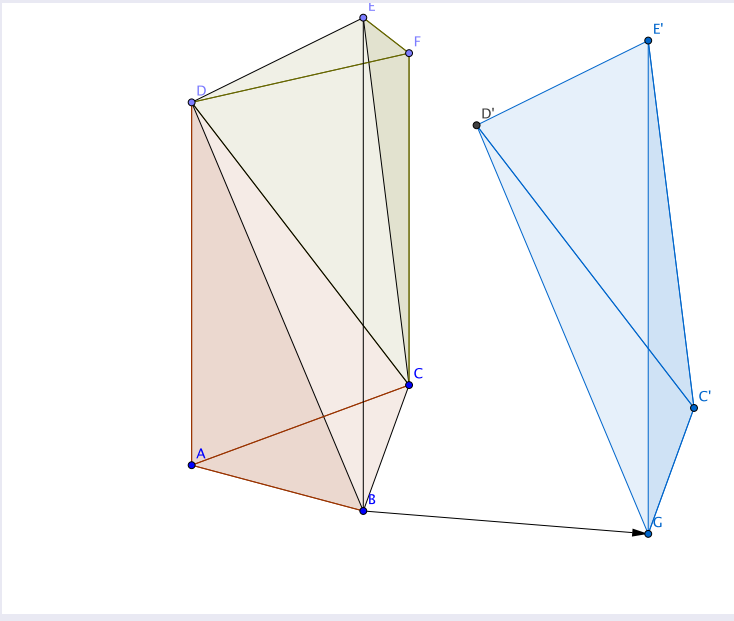
- At each triangle is associated the *volume of the "prism"* who lies between the triangle and its triangular orthogonal projection on the paraboloid with equation $z = x^2 + y^2$:



"Prism" between plane and paraboloid.

- Such a prism can be decomposed into 3 tetrahedra, the volume of each being computed thanks to a determinant like this one for `incircle`.

A strict order on triangulations (cont'd)



"Prism" decomposed into 3 tetrahedra.

A strict order on triangulations (cont'd)

- It is proven that, during a `Flip`, the *sum of the prism volumes* for the *two starting triangles* – incidental to the illegal edge – is *stricly smaller than* the sum for the two final triangles.
- It is the same for the *sum of the prism volumes associated with all triangles* of a triangulation.
- But, this is not a correct argument to establish the finiteness of any `Flip` sequence.
  A *Noetherian ordering* is necessary, for instance the standard ordering in the natural numbers.

Generic library for finite sets

```
Record fset (T:Type) := mkfs {
  prd :> T -> Prop;
  fs_enum : list T;
  _ : forall x, prd x -> In x fs_enum
}.
```

- *Double point of view* (predicate, list) inspired from the `ssreflect` package.
- Finiteness is preserved by: *cartesian product, disjoint sum, inclusion, inverse image through an injection, construction of lists of fixed length, construction of lists of bounded length, construction of lists without duplication*.

Finite set of triangulations

- From any hypermap $\mathtt{m}$, the *finite set of all hypermaps* with the same dart set and point set ones as $\mathtt{m}$ is first built.
- Then, it is restricted to the *finite set of all well-embedded triangulations*.
- Finally, it is shown that this triangulation set is *invariant by Flip* on $\mathtt{m}$.

Termination of the `Flip` *sequence*

- So, for each current triangulation `m`, a *natural measure* is defined as the cardinal of the subset of the triangulations the volume sum of which is smaller than `m`'s one.
- This measure is proven to be *strictly decreasing* at each `Flip`.
- Then, once the strategy to choose the illegal edge at each step is fixed, the *sequence of* `Flip` until there is no more illegal edge is proven to be *finite*.

Solving the Delaunay problem

Generic solution and Delaunay function

```
Function rf (t : good)
 {measure nat_measure} : fmap :=
   if final_dec (good_to_fmap t)
   then good_to_fmap t
   else rf (step_good t).

Definition Delaunay(m:fmap)
(IT: inv_Triangulation m)(WE:isWellembed m) :=
   rf (exist ... m (conj IT WE)).
```

Partial correctness

```
Theorem correctness_Delaunay :
 forall(m: fmap)(IT: inv_Triangulation m)
       (WE: isWellembed m),
 let m1:= (Delaunay m IT WE) in
   inv_Triangulation m1 /\
      isWellembed m1 /\
         no_dart_illegal m1.

Theorem planar_Delaunay :
 forall (m : fmap)(IT: inv_Triangulation m)
       (WE: isWellembed m),
    planar m -> planar (Delaunay m IT WE).
```

Extraction of the OCaml Delaunay program

```
(* Coq directives: *)
Extract Inductive sumbool =>
     "bool" [ "true" "false" ].
Extraction "Delaunay.ml"  Delaunay.

(** Extracted OCaml program: **)
let rec rf_terminate t =
 if final_dec then t
 else rf_terminate t

let rf t = rf_terminate t

let delaunay m = rf m
```

- In the functional (necessarily correct) final program, all the logical features which were necessary for the proof of termination – in particular *finite sets* – are disappeared.

Conclusions and Future work

Conclusions

- We proved the *total correctness* of a *naive classical Delaunay algorithm*
- The reasoning uses *hypermaps* which help to separate topological and geometrical aspects
- The underlaying *hypermap library* counts about $65,000$ Coq lines (250 definitions and 600 lemmas/theorems)
- The *specific development for Delaunay* represents $5,000$ Coq lines (about 50 definitions and 100 lemmas/theorems)
- The proofs intensively use generic *finite sets*, the Coq *real library*, new Coq features to decide polynomial systems (*Positivstellensatz*).

Future work

- The use of the Coq real library must be revisited in order to integrate *floating point rounding errors*
- Our *naive 2D Delaunay algorithm* must be supplemented by a preliminary triangulation starting from points; the case with a general external convex hull has to be treated
- Other classical *2D or 3D Delaunay algorithms* must be studied with hypermaps and similar proof techniques...