An intuitionistic proof of a discrete form of the Jordan Curve Theorem formalized in Coq with combinatorial hypermaps

Jean-François Dufourd[1]

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

Galapagos Days, Strasbourg, December 17-18, 2008

Introduction

- *Objective*:
  - 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 polyhedra
- *"Benchmark" of real size*:
  - Discrete Jordan Curve Theorem (JCT)

Outline

Related work

- Statement of the Jordan Curve Theorem (JCT) [C. Jordan, 1887]
- First correct proof of the JCT [O. Veblen, 1905]
- Comb. maps and hypermaps [W.T. Tutte, R. Cori..., 1970-]
- Discrete JCT with combinatorial maps [W.T. Tutte, 1979]
- Planar graphs and triangulations formalized in Isabelle [G. Bauer, T. Nipkow, 2003]
- Formalized proof of the classical JCT in the MIZAR project [A. Kornilowicz, 2005]
- Formalized proof of the JCT for rectangular grids in the Flyspeck project [T. Hales, 2005]
- Specification of hypermaps and proof in Coq of the Four Colour Theorem [G. Gonthier et al., 2005]
- Other hypermap specification and proof in Coq of Genus Theorem and Euler Formula [J.-F. Dufourd, 2006]

Mathematical aspects

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*)

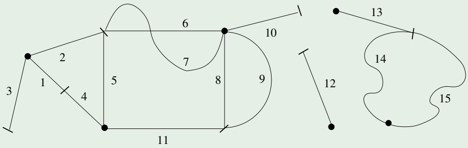
The *topological cells* of a dart x in a hypermap are dart sets which are traversed while iterating some operations from x:
- *edge* of x: iteration of α_0
- *vertex* of x: iteration of α_1
- *face* of x: iteration of $\phi = \alpha_1^{-1} \circ \alpha_0^{-1}$
- *connected component* of x: iteration of both α_0 and α_1.

(*Projection, embedding*)

- In a *projection* of a hypermap onto a surface: vertices and edges are projected onto points, darts onto open Jordan curves, faces onto open connected regions.
- An *embedding* is a projection without self-intersection which determines a *subdivision* of the surface.

Example: A hypermap projected onto a plane (with a self-intersection)

D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
α0	4	5	3	1	6	7	2	11	8	10	9	12	14	15	13
α1	2	3	1	11	4	7	8	9	10	6	5	12	13	15	14



15 darts, 7 edges (*strokes*), 6 vertices (*bullets*), 6 faces, and 3 connected components.

Let M be a hypermap, and d, e, v, f, c be its *numbers* of *darts, edges, vertices, faces, connected components*, respectively.

Definition (*Euler characteristic, genus, planarity, Euler formula*)

(i) The *Euler characteristic* of M is $\chi = v + e + f - d$.
(ii) The *genus* of M is $g = c - \chi/2$.
(iii) When $g = 0$, the hypermap is said to be *planar*.
(iv) A planar hypermap satisfies the *Euler formula*:

$$\chi = v + e + f - d = 2 * c$$

.

Example

For the example hypermap, $\chi = 6 + 6 + 7 - 15 = 4$ and $g = 3 - \chi/2 = 1$. Thus, the hypermap is non planar.

Theorem ***of the genus***

(i) χ is an even integer.
(ii) g is a non-negative integer.

Interpretation of the genus

The *genus* corresponds with the minimal *number of holes* in an orientable closed surface the hypermap can be embedded onto (without self-intersection).

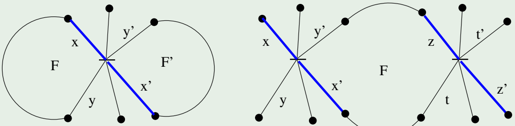
Example

The example hypermap with genus 1 cannot be embedded onto a *plane*. But it can be embedded onto a *torus* with 1 hole.

Definition (*Double-link and adjacencies*)

(1) A *double-link* is a pair of darts (x, x') where x and x' are distinct and belong to the same edge.
(2) The faces F and F' of M are said to be *adjacent by the double-link* (x, x') when $y = \alpha_0(x)$ is a dart of F and $y' = \alpha_0(x')$ a dart of F'.
(3) The double-links (x, x') and (z, z') are said to be *adjacent by the face* F when $\alpha_0(x')$ and $\alpha_0(z)$ are in F.

Example: Double-link and adjacencies



Double−link (x, x').
F and F' are adjacent by (x, x').

The double−links (x, x') and (z, z') are adjacent by the face F.

Definition (*Ring of faces in a hypermap*)

A *ring of faces* R of length n in M is a non-empty sequence of double-links (x_i, x'_i), for $i = 1, \ldots, n$, with the following properties, where E_i is the edge of x_i and F_i the face of $y_i = \alpha_0(x_i)$:

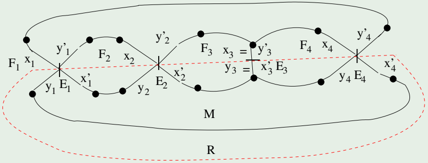
(0) *Unicity:* E_i and E_j are distinct, for $i, j = 1, \ldots, n$ and $i \neq j$;

(1) *Continuity:* F_i and F_{i+1} are adjacent by the double-link (x_i, x'_i), for $i = 1, \ldots, n-1$;

(2) *Circularity*, or *closure:* F_n and F_1 are adjacent by the double-link (x_n, x'_n);

(3) *Simplicity:* F_i and F_j are distinct, for $i, j = 1, \ldots, n$ and $i \neq j$.

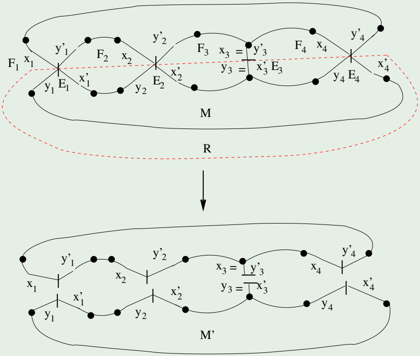
Example: A ring R of length $n = 4$ in hypermap M



Definition (*Break of a hypermap along a ring*)

Let R be a ring $(x_i, x'_i)_{1 \leq i \leq n}$ of faces in M, with $y_i = \alpha_0(x_i)$, and $M_i = (D, \alpha_{0,i}, \alpha_1)_{0 \leq i \leq n}$ be the hypermap sequence, where the $\alpha_{0,i}$ are recursively defined by:
(1) $i = 0$: $\alpha_{0,0} = \alpha_0$;
(2) $1 \leq i \leq n$: for each $z \in D$,
$\alpha_{0,i}(z)$ = if $z = x_i$ then y'_i else if $z = x'_i$ then y_i else $\alpha_{0,i-1}(z)$.
Then, $M_n = (D, \alpha_{0,n}, \alpha_1)$ is said to be obtained from M by a *break along* R.

Example: Break of M along the ring R giving M'



Discrete Jordan Curve Theorem

Let M be a *planar* hypermap with c components, R be a ring of faces in M, and M' be the break of M along R. The number c' of components of M' is such that $c' = c + 1$.

Hypermap specifications in Coq

Inductive definition of a type `dim` of *dimensions*

```
Inductive dim:Set:= zero: dim | one: dim.
```

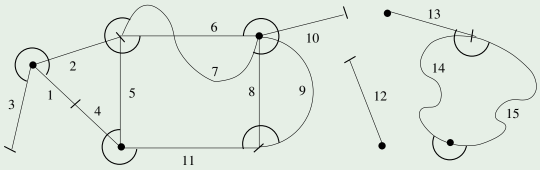
Definition of a type `dart` of *darts* as a renaming of `nat`

```
Definition dart:= nat.
Definition nil:= 0.
```

Inductive definition of a type `fmap` of *free maps* (free algebra of terms)

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

Example: Description of the previous hypermap. Links are represented by *arcs of circle*. The orbits are (intentionally) *incompletely linked*



Observers of free maps

- *Existence* `(exd m z)` *of a dart* `z` in a free map `m`:

```
Fixpoint exd(m:fmap)(z:dart){struct m}:Prop
 := match m with
      V => False
    | I m0 x _ => z=x \/ exd m0 z
    | L m0 _ _ _  => exd m0 z
    end.
```

- *Partial operation* α_k: `(A m k z)` returns the successor of `z` in `m` at the dimension `k` if it exists, otherwise `nil`.
- *Existence* `(succ m k z)` *of a successor* of `z` for `A`:

```
Definition succ(m:fmap)(k:dim)(z:dart):=
  A m k z <> nil.
```

Observers of free maps (continued)

- *Extremities* of orbits: `(bottom m k z)` and `(top m k z)` give the extremities of the (open) `k`-orbit of `z`
- *Complete operation* α_k, named `cA`: realizes the *closures* of the `k`-orbits
- *Face traversal*: *partial successor* `F`, *complete successor* `cF`

Destructors in a free map

- *Deletion* `D` of a dart `z` in `m`: `(D m z)`
- *Break* `B` in `m` of a link starting from `z` at dimension `k` *(deletion of an arc of circle)*: `(B m k z)`

Inverses or *symmetrical operations*

`A_1`, `pred`, `cA_1`, `B_1`, `F_1`, `cF_1`...

Hypermaps as a subtype of the free maps

- *Precondition* on `I`: `(prec_I m x)` expresses that `x` is different from `nil` and does not exist in `m`.
- *Precondition* on `L`: `(prec_L m k x y)` expresses that `x` and `y` both exist in `m`, `x` has no `k`-successor, `y` has no `k`-predecessor, and that their `k`-orbit will stay open.
- *Invariant* of the *hypermaps* (with *open* edges and vertices):

```
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.
```

- *Properties*: for any `m` and `k`, `cA` is a *permutation* and `cA_1` is its *inverse*.

Orbits for permutations

- They are genericly defined by Coq *signatures* and *modules* for any inverse bijections `f` and `f_1` in a hypermap.
- A *specialization* is done for `(cA m zero)`, `(cA m one)`, `(cF m)` and their inverses.
- The *existence of a path* in an orbit from dart `x` to dart `y` is easy to define, e.g. *in a face*: `expf m x y`

Connectivity

- The membership of `x` and `y` to the *same connected component* is expressed by: `eqc m x y`

Proven properties

- Each orbit is *periodic* with a *uniform lowest period*.
- `(expf m)` and `(eqc m)` are *decidable equivalences*.

Numbers of darts, edges, vertices, faces, components

```
Fixpoint nd(m:fmap):Z:=
 match m with
    V => 0
  | I m0 x _ => nd m0 + 1
  | L m0 _ _ _ => nd m0
 end.

Fixpoint ne(m:fmap):Z:=
 match m with
    V => 0
  | I m0 x _ => ne m0 + 1
  | L m0 zero x y => ne m0 -
   if eq_dart_dec (cA m0 zero x) y then 0 else 1
  | L m0 one x y => ne m0
 end.

 (* Idem for nv, nf, nc *)
```

Euler characteristic, genus, planarity

```
Definition ec(m:fmap): Z:=
   nv m + ne m + nf m - nd m.

Definition genus(m:fmap): Z:=
  (nc m) - (ec m)/2.

Definition planar(m:fmap): Prop:=
   genus m = 0.
```

Genus Theorem and Euler Formula

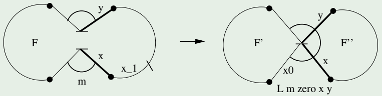
```
Theorem Genus_Theorem: forall m:fmap,
 inv_hmap m -> genus m >= 0.
Proof. (* by induction on m *).

Lemma Euler_Formula: forall m:fmap,
 inv_hmap m -> (planar m <-> ec m / 2 = nc m).
Proof. (* trivial *).
```

Planarity and connectivity criteria

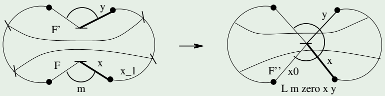
Constructive *criterion of planarity*

```
Theorem planarity_crit_0:
  forall(m:fmap)(x y:dart),
  inv_hmap m -> prec_L m zero x y ->
(planar (L m zero x y) <->
 (planar m /\
   (~eqc m x y \/ expf m (cA_1 m one x) y))).
```

Example: *Linking* x *and* y *at dimension* 0 when connected



a. Planar 0–linking inside a face F giving 2 faces F' and F''.



b. Non–planar 0–linking between 2 faces F and F' giving face F''.

Destructive *criterion of planarity*

```
 Theorem planarity_crit_B0:
   forall(m:fmap)(x:dart),
   inv_hmap m -> succ m zero x ->
    let m0 := B m zero x in
    let y := A m zero x  in
(planar m <->
 (planar m0 /\
  (~eqc m0 x y \/ expf m0 (cA_1 m0 one x) y))).
```

Destructive *criterion of connectivity*

```
  Theorem disconnect_planar_criterion_B0:
   forall (m:fmap)(x:dart),
    inv_hmap m -> planar m -> succ m zero x ->
     let y := A m zero x in
     let x0 := bottom m zero x in
       (expf m y x0 <-> ~eqc (B m zero x) x y).
```

Rings of faces

Modeling a ring

Each ring of faces is viewed as a *list of darts* of type:

```
Inductive list:Set :=
   lam: list
 | cons: dart -> dart -> list -> list.
```

The double-links satisfy:

```
 Definition double_link(m:fmap)(x x':dart)
:Prop:= x <> x' /\ expe m x x'.
```

where `expe m x x'` expresses that `x` and `x'` are in the same edge.

Rings of faces (continued)

Modeling a ring (continued)

Thus, the fact that `l` is really a list of double-links is easily expressed by:

```
Fixpoint double_link_list(m:fmap)
    (l:list){struct l}:Prop:=
  match l with
    lam => True
  | cons x x' l0 => double_link m x x' /\
                    double_link_list m l0
end.
```

Finally, a *ring* satisfies:

```
Definition ring(m:fmap)(l:list):Prop:=
 ~emptyl l /\ double_link_list m l /\
     pre_ring0 m l /\ pre_ring1 m l /\
         pre_ring2 m l /\ pre_ring3 m l.
```

Ring Condition (0): *unicity*

All the darts of a ring are distincts:

```
Fixpoint pre_ring0(m:fmap)(l:list)
  {struct l}:Prop:=
 match l with
    lam => True
  | cons x _ l0 => pre_ring0 m l0 /\
               distinct_edge_list m x l0
 end.
```

Adjacency by a face

Adjacency by a face of `m` for two double-links `(x,x')` and `(xs,xs')`:

```
Definition face_adjacent(m:fmap)
        (x x' xs xs':dart): Prop:=
  let y':= cA m zero x' in
  let ys:= cA m zero xs in
 expf m y' ys.
```

Ring Condition (1): *continuity*

Two successive faces in `l` are adjacent:

```
Fixpoint pre_ring1(m:fmap)(l:list)
     {struct l}:Prop:=
 match l with
    lam => True
  | cons x x' l0 =>
     match l0 with
        lam => True
      | cons xs xs' l' =>
          pre_ring1 m l0 /\
           face_adjacent m x x' xs xs'
     end
 end.
```

Ring Condition (2): *circularity* or *closure*

The *last* and *first* double-links in `l` are adjacent by a face:

```
Definition pre_ring2(m:fmap)(l:list):Prop:=
 match l with
    lam => True
  | cons x x' l0 =>
      match (last l) with (xs,xs') =>
        face_adjacent m xs xs' x x'
      end
 end.
```

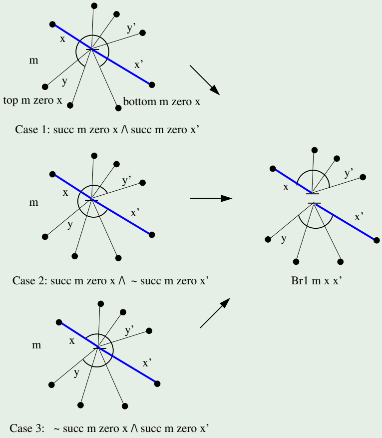
Ring Condition (3): *simplicity*

All faces of `l` are distinct:

```
Fixpoint pre_ring3(m:fmap)(l:list)
      {struct l}:Prop:=
 match l with
    lam => True
  | cons x x' l0 =>
     pre_ring3 m l0 /\
       distinct_face_list m x x' l0
 end.
```

Breaking an edge at a double-link

Example: ***breaking an edge at a double-link***



Breaking a hmap along a ring

Breaking an edge at a double$_l$ink

```
Definition Br1(m:fmap)(x x':dart):fmap:=
  if succ_dec m zero x
  then if succ_dec m zero x'
       then B (L (B m zero x)
           zero (top m zero x)
           (bottom m zero x)) zero x'
       else B m zero x
  else B m zero x'.
```

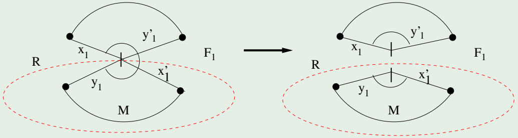
Breaking a hypermap along a ring

```
Fixpoint Br(m:fmap)(l:list){struct l}:fmap:=
 match l with
    lam => m
  | cons x x' l0 => Br (Br1 m x x') l0
 end.
```

First case: ring of length $n = 1$

```
  Lemma Jordan1:forall(m:fmap)(x x':dart),
  inv_hmap m -> planar m ->
   let l:= cons x x' lam in
    ring m l -> nc (Br m l) = nc m + 1.
Proof. (* with the destructive planarity and
           connectivity lemmas *)
```

Example: *break along a ring of length* 1



General case: ring of length $n > 1$

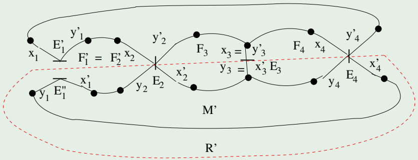
- *Lemma 1: when* $n >= 2$*, a link break preserves the connectivity*

```
  Lemma ring1_ring3_connect:
   forall(m:fmap)(x x' xs xs':dart)(l:list),
     let l1:= cons x x' (cons xs xs' l) in
     let y  := cA m zero x in
     let y' := cA m zero x' in
   inv_hmap m -> planar m ->
      double_link_list m l1 ->
         pre_ring1 m l1 -> pre_ring3 m l1 ->
           ~ expf m y y'.
Proof. (* by induction on l *).
```

- *Lemma 2: when it entails no disconnection, a link break preserves the ring property*

```
   Lemma ring_Br1: forall(m:fmap)(l:list),
     inv_hmap m -> planar m ->
      let x:= fst (first l) in
      let x' := snd (first l) in
      let m1 := Br1 m x x' in
   ring m l ->
     (emptyl (tail l) \/ ring m1 (tail l)).
```

Example: *break along a ring of length* > 1



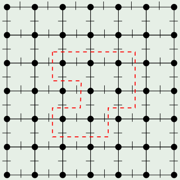
Finally, the Jordan Curve Theorem

```
Theorem Jordan: forall(l:list)(m:fmap),
  inv_hmap m -> planar m ->
     ring m l ->
        nc (Bl m l) = nc m + 1.
Proof. (* by induction on l
          with the previous lemmas *).
```

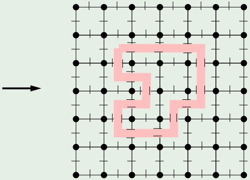
Validity of the theorem, case of the oriented maps

- Our ring specification and our Jordan Curve Theorem formalization are *complete* with respect to our mathematical definitions.
- The best way to see the effective application of this work is to consider *combinatorial oriented maps* (where α_0 is an involution).

Example: ***Jordan Curve in a pixel grid***



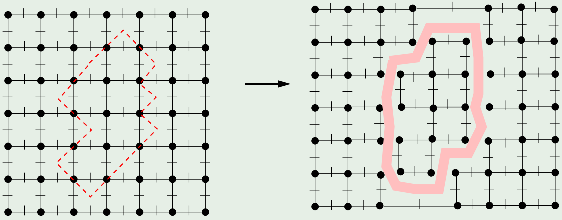
a. A grid map and a ring of faces



b. Grid map in two components after the break along the ring

- Symmetrically: *breaking vertices*

Example: ***Application of the Jordan Curve theorem in a pixel grid in order to break vertices***

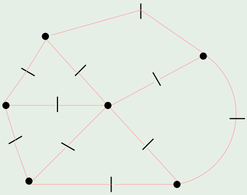


a. Grid map and symmetric ring of faces

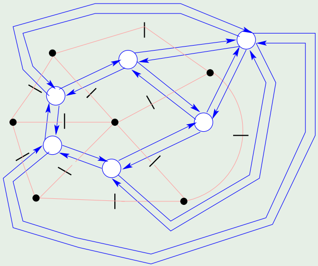
b. Grid map in two components after the break along the ring

- Duality: *dual representations* of maps.

Example: *Conventional and primal-dual representations*



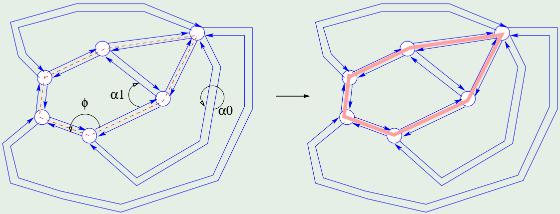
a. A primal representation of a map



b. Primal (light) and dual (dark) representations of a map

- Application of the JCT.

Example: ***Application of the discrete Jordan curve theorem in a dual representation***

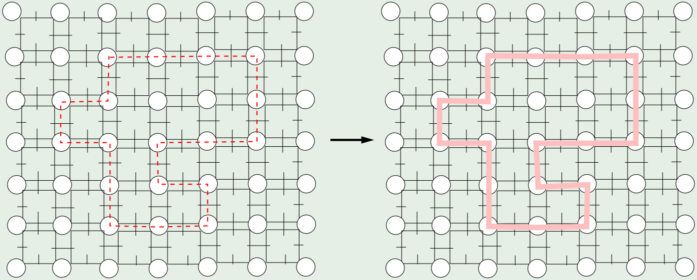


(a) Dual representation of a map and ring

(b) Dual map in two components after the break along the ring

- Application in a dual grid.

Example: ***Application of the Jordan Curve Theorem in a dual grid***



a. A connected grid map in dual representation and a ring

b. Grid map with two components after the break along the ring

Conclusions

We have:

- a *framework of hypermap specifications* *from scratch* of about 17,000 Coq lines to:
  - reason with the topology of surface subdivisions
  - guide implementation and real size programming
- a statement and a constructive proof of a *version of the Discrete Jordan Curve Theorem*, completely formalized and verified: 5,000 Coq lines, 25 definitions and 50 lemmas.

Future work

We want to study:

- in *combinatorial topology*: other models (like map strings, simplicial complexes, cellular complexes) in n-D
- in *geometric modeling, computational geometry*: several embeddings, round-off numerical problems, and how to bypass them
- in *topological-geometrical program construction*: the extraction of certified programs from constructive proofs thanks to the Curry-Howard isomorphism.