Symposium on Theoretical Aspects of Computer Science 2008 (Bordeaux), pp. 253-264
www.stacs-conf.org

DISCRETE JORDAN CURVE THEOREM: A PROOF FORMALIZED IN COQ WITH HYPERMAPS

JEAN-FRANÇOIS DUFOURD [1]

[1] Université Louis-Pasteur de Strasbourg, UFR de Mathématique et d'Informatique,
Labo. des Sciences de l'Image, de l'Informatique et de la Télédétection (UMR CNRS-ULP 7005),
Pôle API, Boulevard Sébastien Brant, 67400 Illkirch, France
E-mail address: dufourd@dpt-info.u-strasbg.fr

Abstract. This paper presents a formalized proof of a discrete form of the Jordan Curve Theorem. It is based on a hypermap model of planar subdivisions, formal specifications and proofs assisted by the Coq system. Fundamental properties are proven by structural or noetherian induction: Genus Theorem, Euler's Formula, constructive planarity criteria. A notion of ring of faces is inductively defined and a Jordan Curve Theorem is stated and proven for any planar hypermap.

Introduction

This paper presents a formal statement and an assisted proof of a Jordan Curve Theorem (JCT) discrete version. In its common form, the theorem says that the complement of a continuous simple closed curve (a Jordan curve) C in an affine real plane is made of two connected components whose border is C, one being bounded and the other not. The discrete form of JCT we deal with states that in a finite subdivision of the plane, breaking a ring R of faces increases by 1 the connectivity of the subdivision. It is a weakened version of the original theorem where the question of bound is missing. However, it is widely used in computational geometry and discrete geometry for imaging, where connection is the essential information (14; 9). In fact, we only are in a combinatoric framework, where any embedding is excluded, and where bounding does not make sense.

In computational topology, subdivisions are best described by map models, the most general being *hypermaps* (15; 4). We propose a purely combinatorial proof of JCT based on this structure. The hypermap framework is entirely formalized and the proofs are developed interactively and verified by the Coq proof assistant (3). Using an original way to model, build and destruct hypermaps, the present work brings new simple constructive planarity and connectivity criteria. It proposes a new direct expression of JCT and a simple constructive proof with algorithmic extensions. It is also a large benchmark for the software

1998 ACM Subject Classification: I.3.5, E1, D.2.4.
Key words and phrases: Formal specifications - Computational topology - Computer-aided proofs - Coq - Planar subdivisions - Hypermaps - Jordan Curve Theorem.
Acknowledgements: This research is supported by the "white" project GALAPAGOS, French ANR, 2007.

specification framework we have been developing in the last fifteen years for map models used in geometric modeling and computer imagery (2; 7; 8).

The useful Coq features are reminded and the whole process is described, but the full details of the proofs are omitted. Section 1 summarizes related work. Section 2 recalls some mathematical materials. Section 3 proposes basic hypermap specifications. Section 4 proves constructive criteria of hypermap planarity and connectivity. Section 5 inductively specifies the rings and their properties. Section 6 proves the discrete JCT. Section 7 concludes.

1. Related work

The JCT is a result of classical plane topology, first stated by C. Jordan in 1887, but of which O. Veblen gives the first correct proof in 1905. In 1979, W.T. Tutte proposes operations and properties of combinatorial maps, *e.g.* planarity and Euler's Formula, defines rings and proves a discrete JCT (15). Our theorem statement is comparable, but our framework is modeled differently and all our proofs are formalized and computer-assisted.

In 2003, G. Bauer and T. Nipkow specify planar graphs and triangulations in Isabelle/Isar to carry out interactive proofs of Euler's Formula and of the Five Colour Theorem (1). However, they do not approach the JCT. In 2005, A. Kornilowicz designs for the MIZAR project a semi-automated classical proof of a continuous form of JCT in an Euclidean space (13). In 2005 also, on his way towards the proof of the Kepler conjecture in the Flyspeck projet, T. Hales proves the JCT for planar rectangular grids with the HOL Light system, following the Kuratowski characterization of planarity (12).

In 2005 always, G. Gonthier *et al.* prove the Four Colour Theorem using Coq. Plane subdivisions are described by hypermaps, and Euler's Formula is used as a global planarity criterion (10). A local criterion, called *hypermap Jordan property*, is proven equivalent. The main part of this work is the gigantic proof of the Four Colour Theorem with hypermaps and sophisticated proof techniques. The hypermap formalization is very different from ours and it seems that JCT is not explicitly proven there. Finally, since 1999, we carry out experiments with Coq for combinatorial map models of space subdivisions (5; 7; 8).

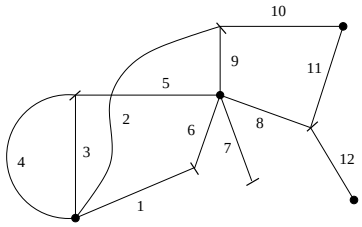
2. Mathematical Aspects

Definition 2.1 (Hypermap)**.** A *hypermap* is an algebraic structure $M = (D, \alpha_0, \alpha_1)$, where D is a finite set whose elements are called *darts*, and α_0, α_1 are permutations on D.

If $y = \alpha_k(x)$, y is the *k-successor* of x, x is the *k-predecessor* of y, and x and y are said to be *k-linked*.

In Fig. 1, as functions α_0 and α_1 on $D = \{1, \ldots, 15\}$ are permutations, $M = (D, \alpha_0, \alpha_1)$ is a hypermap. It is drawn on the plane by associating to each dart a curved arc oriented from a bullet to a small stroke: 0-linked (resp. 1-linked) darts share the same small stroke (resp. bullet). By convention, in the drawings of hypermaps on surfaces, k-successors turn counterclockwise around strokes and bullets. Let $M = (D, \alpha_0, \alpha_1)$ be a hypermap.

Definition 2.2. (Orbits and hypermap cells)
(1) Let $f_1, \ldots, f_n$ be n functions in D. The *orbit* of $x \in D$ for $f_1, \ldots, f_n$ is the subset of D denoted by $\langle f_1, \ldots, f_n \rangle(x)$, the elements of which are accessible from x by any composition of $f_1, \ldots, f_n$.



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

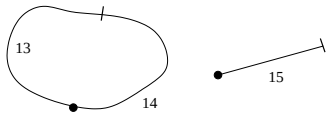


Figure 1: An example of hypermap.

(2) In M, $\langle\alpha_0\rangle(x)$ is the 0-orbit or *edge* of dart x, $\langle\alpha_1\rangle(x)$ its 1-orbit or *vertex*, $\langle\phi\rangle(x)$ its *face* for $\phi = \alpha_1^{-1} \circ \alpha_0^{-1}$, and $\langle\alpha_0, \alpha_1\rangle(x)$ its *(connected) component.*

In Fig. 1 the hypermap contains 7 edges (strokes), 6 vertices (bullets), 6 faces and 3 components. For instance, $\langle\alpha_0\rangle(3) = \{3, 5, 4\}$ is the edge of dart 3, $\langle\alpha_1\rangle(3) = \{3, 4, 1, 2\}$ its vertex. Faces are defined, through ϕ, for a dart traversal in counterclockwise order, when the hypermap is drawn on a surface. Then, every face which encloses a bounded (resp. unbounded) region on its left is called *internal* (resp. *external*). In Fig. 1, the (internal) face of 8 is $\langle\phi\rangle(8) = \{8, 10\}$ and the (external) face of 13 is $\langle\phi\rangle(13) = \{13\}$. Let d, e, v, f and c be the numbers of darts, edges, vertices, faces and components of M.

Definition 2.3. (Euler characteristic, genus, planarity)
(1) The *Euler characteristic* of M is $\chi = v + e + f - d$.
(2) The *genus* of M is $g = c - \chi/2$.
(3) When $g = 0$, M is said to be *planar.*

For instance, in Fig. 1, $\chi = 6 + 6 + 7 - 15 = 4$ and $g = 3 - \chi/2 = 1$. Consequently, the hypermap is non planar. These values satisfy the following results:

Theorem 2.4 (of the Genus)**.** *χ is an even integer and g is a natural number.*

Corollary 2.5 (Euler Formula)**.** *A non empty connected – i.e. with $c = 1$ – planar hypermap satisfies $v + e + f - d = 2$.*

When $D \neq \emptyset$, the *representation* of M on an *orientable closed* surface is a mapping of edges and vertices onto points, darts onto open oriented Jordan arcs, and faces onto open connected regions. It is an *embedding* when every component of M realizes a partition of the surface. Then, the genus of M is the minimum number of *holes* in an orientable closed surface where such an embedding is possible, thus drawing a subdivision, or a polyhedron, by hypermap component (11). For instance, all the components of the hypermap in Fig. 1 can be embedded on a torus (1 hole) but not on a sphere or on a plane (0 hole). When a (planar) hypermap component is embedded on a plane, the corresponding subdivision has exactly one unbounded (external) face. But a non planar hypermap can never be embedded on a plane: in a drawing on a plane, some of its faces are neither internal nor external, *e.g.* $\langle\phi\rangle(1) = \{1, 5, 2, 11, 12, 7, 6, 4, 9\}$ in Fig. 1. Conversely, any subdivision of an *orientable closed* surface can be modeled by a hypermap. In fact, the formal presentation which follows is *purely combinatorial*, *i.e* without any topological or geometrical consideration.

2.1. Rings of faces and Jordan Curve Theorem

To state the version of JCT we will prove, we need the concepts of *double-link*, *adjacent faces* and *ring of faces* in a hypermap $M = (D, \alpha_0, \alpha_1)$.

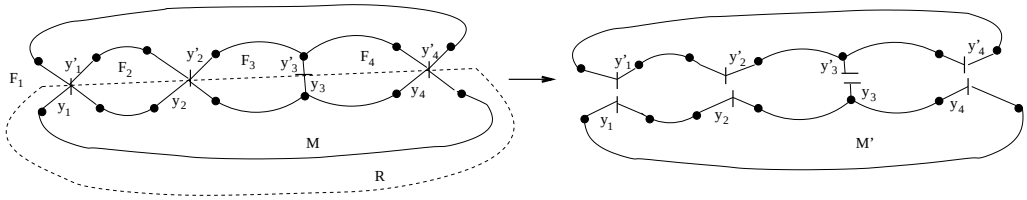


Figure 2: Break of M along a ring R of length $n = 4$ giving M'.

Definition 2.6. (Double-link and adjacent faces)
(1) A *double-link* is a pair of darts (y, y') where y and y' belong to the same edge.
(2) The faces F and F' of M are said to be *adjacent by the double-link* (y, y') if y is a dart of F and y' a dart of F'.

We choose a face adjacency *by an edge* rather than *by a vertex* as does W.T. Tutte (15). In fact, due to the homogeneity of dimensions 0 and 1 in a hypermap, both are equivalent.

Definition 2.7. (Ring of faces)
A *ring of faces* R of length n in M is a non empty sequence of double-links (y_i, y'_i), for $i = 1, \ldots, n$, with the following properties, where E_i and F_i are the edge and face of y_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 (y_i, y'_i), for $i = 1, \ldots, n-1$;
(2) *Circularity*, or *closure:* F_n and F_1 are adjacent by the double-link (y_n, y'_n);
(3) *Simplicity:* F_i and F_j are distinct, for $i, j = 1, \ldots, n$ and $i \neq j$.

This notion simulates a Jordan curve represented in dotted lines in Fig. 2 on the left for $n = 4$. Then, we define the *break along a ring*, illustrated in Fig. 2 on the right.

Definition 2.8. (Break along a ring)
Let R be a ring of faces of length n in M. Let $M_i = (D, \alpha_{0,i}, \alpha_1)$, for $i = 0, \ldots, n$, be a 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 dart z of D: $\alpha_{0,i}(z) =$ if $\alpha_{0,i-1}(z) = y_i$ then y'_i else if $\alpha_{0,i-1}(z) = y'_i$ 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.

Finally, the theorem we will prove in Coq mimics the behaviour of a cut along a simple Jordan curve of the plane (or of the sphere) into two components:

Theorem 2.9 (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 $c' = c + 1$.*

3. Hypermap specifications

3.1. Preliminary specifications

In Coq, we first define an inductive type `dim` for the two dimensions at stake:

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

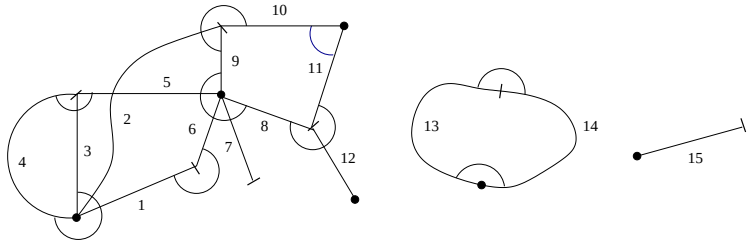


Figure 3: A hypermap with its incompletely linked orbits.

All objects being typed in Coq, `dim` has the type `Set` of all concrete types. Its *constructors* are the constants `zero` and `one`. In each inductive type, the generic equality predicate = is built-in but its decidability is not, because Coq's logic is intuitionistic. For `dim`, the latter can be established as the lemma:

```
Lemma eq_dim_dec: forall i j : dim, {i=j}+{~i=j}.
```

Once it is made, its proof is an object of the sum type {`i=j`}+{`~i=j`}, *i.e.* a function, named `eq_dim_dec`, that tests whenever its two arguments are equal. The lemma is interactively proven with some tactics, the reasoning being merely a structural induction on both `i` and `j`, here a simple case analysis. Indeed, from each inductive type definition, Coq generates an *induction principle*, usable either to prove propositions or to build total functions on the type. We identify the type `dart` and its equality decidability `eq_dart_dec` with the built-in `nat` and `eq_nat_dec`. Finally, to manage exceptions, a `nil` dart is a renaming of `0`:

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

3.2. Free maps

The hypermaps are now approached by a general notion of *free map*, thanks to a free algebra of terms of inductive type `fmap` with 3 constructors, `V`, `I` and `L`, respectively for the *empty* (or *void*) map, the *insertion* of a dart, and the *linking* of two darts:

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

For instance, the hypermap in Fig. 1 can be modeled by the free map represented in Fig.3 where the 0- and 1-links by `L` are represented by arcs of circle, and where the orbits remain open. Again, Coq generates an induction principle on free maps.

Next, *observers* of free maps can be defined. The predicate `exd` express that a dart exists in a hypermap. Its definition is recursive, which is indicated by `Fixpoint`, thanks to a pattern matching on `m` written `match m with....` The attribute {`struct m`} allows Coq to verify that the recursive calls are performed on smaller `fmap` terms, thus ensuring termination. The result is `False` or `True`, basic constants of `Prop`, the built-in type of propositions. Note that terms are in prefix notation and that `_` is a place holder:

```
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.
```

The decidability `exd_dec` of `exd` directly derives, thanks to a proof by induction on `m`. Then, a version, denoted `A`, of operation α_k of Definition 2.1 completed with `nil` for convenience is written as follows, the inverse `A_1` being similar:

```
Fixpoint A(m:fmap)(k:dim)(z:dart){struct m}:dart:=
 match m with
   V => nil | I m0 x => A m0 k z | L m0 k0 x y =>
     if eq_dim_dec k k0 then if eq_dart_dec z x then y else A m0 k z
     else A m0 k z
 end.
```

Predicates `succ` and `pred` express that a dart has a `k`-successor and a `k`-predecessor (not `nil`), with the decidabilities `succ_dec` and `pred_dec`. In hypermap `m` of Fig. 3, `A m zero 4 = 3, A m zero 5 = nil, succ m zero 4 = True, succ m zero 5 = False, A_1 m one 2 = 1`. In fact, when a `k`-orbit remains open, which will be required in the following, we can obtain its `top` and `bottom` from one of its dart `z`. Then, we can do as if the `k`-orbit were closed, thanks to the operations `cA` and `cA_1` which *close* `A` and `A_1`, in a way similar to operation K of W.T. Tutte (15). For instance, in Fig. 3, `top m one 1 = 3, bottom m one 1 = 4, cA m one 3 = 4, cA_1 m one 4 = 3`.

Finally, *destructors* are also recursively defined. First, `D:fmap->dart->fmap` deletes the latest insertion of a dart by `I`. Second, `B, B_:fmap->dim->dart->fmap` break the latest `k`-link inserted for a dart by `L`, forward and backward respectively.

3.3. Hypermaps

Preconditions written as predicates are introduced for `I` and `L`:

```
Definition prec_I(m:fmap)(x:dart):= x <> nil /\ ~ exd m x.
Definition prec_L(m:fmap)(k:dim)(x y:dart):=
 exd m x /\ exd m y /\ ~ succ m k x /\ ~ pred m k y /\ cA m k x <> y.
```

If `I` and `L` are used under these conditions, the free map built necessarily has open orbits. In fact, thanks to the closures `cA` and `cA_1`, it can always be considered as a true hypermap exactly equipped with operations α_k of Definition 2.1. It satisfies the *invariant*:

```
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.
```

Such a hypermap was already drawn in Fig. 3. Fundamental proven properties are that, for any `m` and `k`, `(A m k)` and `(A_1 m k)` are *injections* inverse of each other, and `(cA m k)` and `(cA_1 m k)` are *permutations* inverse of each other, and are closures. Finally, traversals of faces are based on function `F` and its closure `cF`, which correspond to ϕ (Definition 2.2). So, in Fig. 3, `F m 1 = nil, cF m 1 = 5`. Properties similar to the ones of `A`, `cA` are proven for `F`, `cF` and their inverses `F_1`, `cF_1`.

3.4. Orbits

Testing if there exists a path from a dart to another in an orbit for a hypermap permutation is of prime importance, for instance to determine the number of orbits. The problem is exactly the same for α_0, α_1 or ϕ (Definitions 2.1 and 2.2). That is why a *signature* `Sigf` with formal parameters `f`, `f_1` and their properties is first defined.

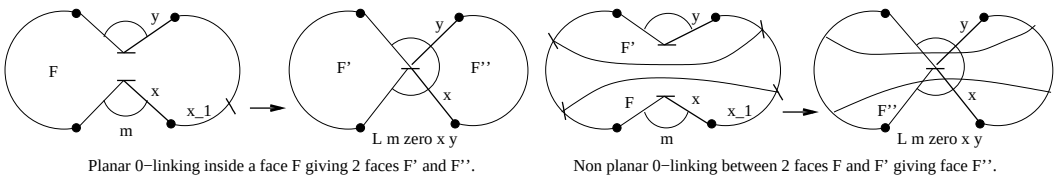


Figure 4: Linking at dimension 0.

Next, a *generic module* (or *functor*) `Mf(M:Sigf)`, the formal parameter `M` being a *module* of type `Sigf`, is written in Coq to *package* generic definitions and proven properties about `f` and `f_1`. Among them, we have that each `f`-orbit of `m` is *periodic* with a positive smallest uniform period for any dart `z` of the orbit. The predicate `expo m z t` asserts the *existence of a path* in an `f`-orbit of `m` from a dart `z` to another `t`, which is proven to be a *decidable equivalence*. Note that most of the properties are obtained by *noetherian induction* on the length of iterated sequences of `f`-successors, bounded by the period.

Appropriate modules, called `MA0`, `MA1` and `MF`, are written to instantiate for `(cA m zero)`, `(cA m one)` and `(cF m)` definitions and properties of `f`. So, a generic definition or property in `Mf(M)` has to be prefixed by the module name to be concretely applied. For instance, `MF.expo m z t` is the existence of a path from `z` to `t` in a face. In the following, `MF.expo` is abbreviated into `expf`. For instance, in Fig. 3, `expf m 1 5 = True, expf m 5 3 = False`. Finally, a binary relation `eqc` stating that two darts belong to the same component is easily defined by induction. For instance, in Fig. 3, we have `eqc m 1 5 = True, eqc m 1 13 = False`. We quickly prove that `(eqc m)` is a *decidable equivalence*.

3.5. Characteristics, Genus Theorem and Euler Formula

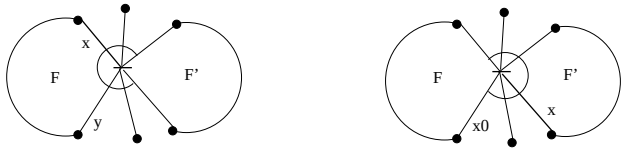
We now count cells and components of a hypermap using the Coq library module `ZArith` containing all the features of `Z`, the integer ring, including tools to solve linear systems in Presburger's arithmetics. The numbers `nd`, `ne`, `nv`, `nf` and `nc` of darts, edges, vertices, faces and components are easily defined by induction. Euler's characteristic `ec` and `genus` derive. The Genus Theorem and the Euler Formula (for any number `(nc m)` of components) are obtained as corollaries of the fact that `ec` is even and satisfies `2 * (nc m) >= (ec m)` (8). Remark that `->` denotes a functional type in `Set` as well as an implication in `Prop`:

```
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.
Theorem Genus_Theorem: forall m:fmap, inv_hmap m -> genus m >= 0.
Theorem Euler_Formula: forall m:fmap, inv_hmap m -> planar m ->
     ec m / 2 = nc m.
```

4. Planarity and connectivity criteria

A consequence of the previous theorems is a completely constructive *criterion of planarity*, when one correctly *links* with `L` at dimensions 0 or 1, *e.g.* for 0:

```
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))).
```



a. Double−link coded by (x, true) : face F contains y. b. Double−link coded by (x, false) : face F contains x0.

Figure 5: Coding a double-link and identifying a face.

So, the planarity of `m` is preserved for `(L m zero x y)` *iff* one of the following two conditions holds: (1) `x` and `y` are not in the same component of `m`; (2) `x_1 = (cA_1 m one x)` and `y` are in the same face of `m`, *i.e.* the linking operates *inside the face* containing `y`. Fig. 4 illustrates 0-linking inside a face, giving two new faces, and between two (connected) faces, giving a new face, thus destroying planarity. Finally, after a long development, we prove the expected *planarity criterion*, when breaking a link with `B`, at any dimension, *e.g.* for 0:

```
Lemma 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))).
```

Such a lemma is easy to write/understand as a *mirror form* of the 0-linking criterion, but it is much more difficult to obtain. It would be fruitful to relate these constructive/destructive criteria with the static one of G. Gonthier (10). Finally, some useful results quickly characterize the effect of a link break on the *connectivity* of a planar hypermap. For instance, when 0-breaking `x`, a disconnection occurs *iff* `expf m y x0`:

```
Lemma 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).
```

5. Rings of faces

5.1. Coding a double-links and identifying a face

Since an edge is always open in our specification, when doing the backward break of a unique 0-link from `y` or `y'`, we in fact realize a double-link break, as in Definition 2.8. So, we choose to identify a double-link by the unique dart, we called `x`, where the 0-link to be broken begins. In fact, with respect to the face `F` *on the left of* the double-link in the ring, there are two cases, depending on the position of `x` and its forward 0-link, as shown in Fig. 5 (a) and (b). We decided to distinguish them by a Boolean `b`. Then, a double-link is coded by a pair `(x, b)`. So, we implicitely identify each ring face `F` by the double-link coding on its right in the ring. In Fig. 5 (a), face `F` is identified by `(x, true)` and contains `y:= A m zero x`, whereas in Fig. 5 (b), face `F` is identified by `(x, false)` and contains `x0:= bottom m zero x`. These modeling choices considerably simplify the problems. Indeed, in closed orbits, a true double-link break would entail 2 applications of `B` followed by 2 applications of `L`, and would be much more complicated to deal with in proofs.

5.2. Modeling a ring of faces

First, we inductively define linear lists of pairs of booleans and darts, with the two classical constructors `lam` and `cons`, and usual observers and destructors, which we do not give, because their effect is directly comprehensible:

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

Such a list is composed of couples `(x, b)`, each identifying a face `F`: if `b` is `true`, `F` is represented by `y:= A m zero x`, otherwise by `x0:= bottom m zero x` (Fig. 5). In the following, `Bl m l` breaks all the 0-links starting from the darts of list `l` in a hypermap `m`. Now, we have to model the conditions required for list `l` to be a ring of hypermap `m`. Translating Definition 2.7, we have four conditions, called `pre_ringk m l`, for `k` $= 0, \ldots, 3$, which we explain in the following sections. Finally, a predicate `ring` is defined by:

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

5.3. Ring Condition (0): unicity

The predicate `distinct_edge_list m x l0` saying that the edges of `l0` are distinct in `m` from a given edge of `x`, `pre_ring0 m l` is defined recursively on `l` to impose that all edges in `l` are distinct: Condition (0) of Definition 2.7. It also imposes that each dart in `l` has a 0-successor, in order to have well defined links, which is implicit in the mathematical definition, but not in our specification whith open orbits.

```
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 /\ succ m zero x
 end.
```

5.4. Ring Condition (1): continuity

Then, we define adjacency between two faces identified by `xb = (x, b)` and `xb' = (x', b')`, along the link corresponding to `xb`:

```
Definition adjacent_faces(m:fmap)(xb xb':dart*bool):=
 match xb with (x,b) => match xb' with (x',b') =>
 let y := A m zero x in let y':= A m zero x' in
 let x0 := bottom m zero x in let x'0:= bottom m zero x' in
   if eq_bool_dec b true
   then if eq_bool_dec b' true then expf m x0 y' else expf m x0 x'0
   else if eq_bool_dec b' true then expf m y y' else expf m y x'0
 end end.
```

This definition is illustrated in Fig. 6 for the four possible cases of double-link codings. So, the predicate `pre_ring1 m l` recursively specifies that two successive faces in `l` are adjacent: Condition (1) in Definition 2.7:

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

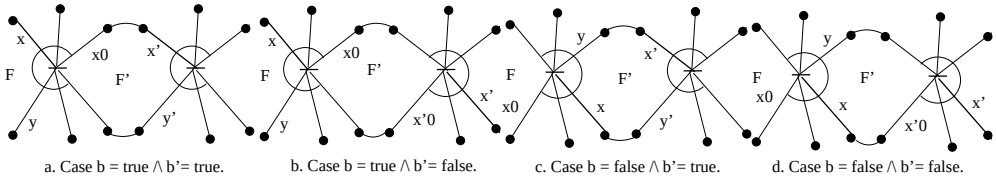


Figure 6: Four cases of face adjacency.

5.5. Ring Condition (2): circularity, or closure

The predicate `pre_ring2 m l` specifies that the last and first faces in `l` are adjacent: Condition (2) of circularity in Definition 2.7:

```
Definition pre_ring2(m:fmap)(l:list):Prop:=
 match l with
    lam => True | cons xb l0 =>
     match xb with (x,b) => let y := A m zero x in match l0 with
       lam => let x0 := bottom m zero x in expf m y x0
     | cons _ l' => let xb':= last l0 in adjacent_faces m xb' xb
     end end
 end.
```

5.6. Ring Condition (3): simplicity

The predicate specifiying that the faces of `m` identified by `xb` and `xb'` are distinct is easy to write by cases on the Booleans in `xb` and `xb'`. The predicate `distinct_face_list m xb l0` expressing that the face identified by `xb` is distinct from all faces of list `l0` entails. Then, the predicate `pre_ring3 m l` says that all faces of `l` are distinct: Condition (3) in Definition 2.7:

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

6. Discrete Jordan Curve Theorem

The general principle of the JCT proof for a hypermap `m` and a ring `l` is a structural induction on `l`. The case where `l` is empty is immediatly excluded because `l` is not a ring by definition. Thus the true first case is when `l` is *reduced to one element*, *i.e.* is of the form `cons (x, b) lam`. Then, we prove the following lemma as a direct consequence of the planarity criterion `planarity_crit_B0` and the criterion `face_cut_join_criterion_B0`:

```
Lemma Jordan1:forall(m:fmap)(x:dart)(b:bool), inv_hmap m -> planar m ->
   let l:= cons (x,b) lam in ring m l -> nc (Bl m l) = nc m + 1.
```

When a ring `l1` contains *at least two elements*, we prove that the condition `~expf m y x0` must hold with the first element `(x,b)` of `l1` (in fact, conditions (1) and (3) are enough):

```
Lemma ring1_ring3_connect:
 forall(m:fmap)(x x':dart)(b b':bool)(l:list), inv_hmap m ->
 let l1:= cons (x,b) (cons (x',b') l) in
 let y:=A m zero x in let x0:= bottom m zero x in
   planar m -> pre_ring1 m l1 -> pre_ring3 m l1 -> ~expf m y x0.
```

In this case, thanks to `disconnect_planar_criterion_B0` (Section 4), the lemma entails that the break of the first ring link does never disconnect the hypermap. Then, after examining the behavior of `pre_ringk`, for `k` $= 0, \ldots, 3$, we are able to prove the following lemma which states that the four ring properties are preserved after the first break in `l`:

```
Lemma pre_ring_B: forall(m:fmap)(l:list), inv_hmap m -> planar m ->
 let x := fst (first l) in let y := A m zero x in
 let x0 := bottom m zero x in let m1 := B m zero x in
~expf m y x0 -> ring m l -> (pre_ring0 m1 (tail l) /\ pre_ring1 m1 (tail l)
    /\ pre_ring2 m1 (tail l) /\ pre_ring3 m1 (tail l)).
```

The most difficult is to prove the part of the result concerning `pre_ringk`, for `k` $= 0, \ldots, 3$. The four proofs are led by induction on `l` in separate lemmas. For `pre_ring0`, the proof is rather simple. But, for the other three, the core is a long reasoning where 2, 3 or 4 links are involved in input. Since each link contains a Boolean, sometimes appearing also in output, until $2^4 = 16$ cases are to be considered to combine the Boolean values.

Finally, from `Jordan1` and `pre_ring_B` above, we have the expected result by a quick reasoning by induction on `l`, where links are broken one by one from the first:

```
Theorem Jordan: forall(l:list)(m:fmap),
   inv_hmap m -> planar m -> ring m l -> nc (Bl m l) = nc m + 1.
```

It is clear that, provided a mathematical hypermap M and a mathematical ring R conform to Definitions 2.1 and 2.7, we can always describe them as terms of our specification framework in order to apply our JCT. Conversely, given a hypermap term, some mathematical rings cannot directly be written as terms. To do it, our ring description and our JCT proof have to be slightly extended. However, that is not necessary for the *combinatorial maps* (where α_0 is an involution) terms, for which our ring specification and our JCT formalization are *complete*. This is more than enough to affirm the value of our results.

7. Conclusion

We have presented a new discrete statement of the JCT based on hypermaps and rings, and a formalized proof assisted by the Coq system. Our hypermap modeling with *open orbits* simplifies and precises most of known facts. It also allows to obtain some new results, particularly about hypermap construction/destruction, connection/disconnection and planarity. This work involves a substantial framework of hypermap specification, which is built *from scratch*, *i.e.* exempt from any proper axiom. It is basically the same as the one we have designed to develop geometric modelers via algebraic specifications (2). So, we know how to efficiently implement all the notions we formally deal with.

The Coq system turned out to be a precious auxiliary to guide and check all the process of specification and proof. The preexistent framework of hypermap specification represents about 15,000 lines of Coq, and the JCT development about 5,000 lines, including about 25 new definitions, and 400 lemmas and theorems. Note that all results about the dimension 0

were actually proven, but some planarity properties about dimension 1, which are perfectly symmetrical, have just been admitted. However, the JCT formal proof is complete.

So, we have a solid foundation to tackle any topological problem involving orientable surface subdivisions. Extensions are in 2D or 3D computational geometry and geometric modeling by introducing embeddings (6; 2), and computer imagery by dealing with pixels (7) or voxels.

References

[1] Bauer, G., Nipkow, T.: The 5 Colour Theorem in Isabelle/Isar. In *Theorem Proving in HOL Conf.* (2002). LNCS **2410**, Springer-Verlag, 67–82.

[2] Bertrand, Y., Dufourd, J.-F.: Algebraic specification of a 3D-modeler based on hypermaps. *Graphical Models and Image Processing* **56**:1 (1994), 29–60.

[3] The Coq Team Development-LogiCal Project: The Coq Proof Assistant Reference Manual - Version 8.1, INRIA, France (2007). `http://coq.inria.fr/doc/main.html`.

[4] Cori, R.: Un Code pour les Graphes Planaires et ses Applications. *Astérisque* **27** (1970), Société Math. de France.

[5] Dehlinger, C., Dufourd, J.-F.: Formalizing the trading theorem in Coq. *Theoretical Computer Science* **323** (2004), 399–442.

[6] Dufourd, J.-F., Puitg, F.: Fonctional specification and prototyping with combinatorial oriented maps. *Comp. Geometry - Th. and Appl.* **16** (2000), 129–156.

[7] Dufourd, J.-F.: Design and certification of a new optimal segmentation program with hypermaps. *Pattern Recognition* **40** (2007), 2974–2993.

[8] Dufourd, J.-F.: A hypermap framework for computer-aided proofs in surface subdivisions: Genus theorem and Euler's formula. In: *22nd ACM SAC* (2007), 757–761.

[9] Françon, J.: Discrete Combinatorial Surfaces. CVGIP : Graphical Models and Image Processing **57**:1, (1995), 20–26.

[10] Gonthier, G.: A computer-checked proof of the Four Colour Theorem. Microsoft Research, Cambridge, `http://coq.inria.fr/doc/main.html` (2005), 57 pages.

[11] Griffiths, H.: *Surfaces.* Cambridge University Press (1981).

[12] Hales, T.: A verified proof of the Jordan curve theorem. Seminar Talk. Dep. of Math., University of Toronto (2005), `http://www.math.pitt.edu/~thales`.

[13] Kornilowicz A.: Jordan Curve Theorem. In: *Formalized Mathematics* **13**:4 (2005), Univ. of Bialystock, 481–491.

[14] Rosenfeld, A.: Picture Languages - Formal Models for Picture Recognition. In: *Comp. Science and Appl. Math.* **series**. Academic Press, New-York (1979).

[15] Tutte, W.T.: Combinatorial oriented maps. *Can. J. Math.*, **XXXI**:5 (1979), 986–1004.