J Autom Reasoning (2009) 43:19–51
DOI 10.1007/s10817-009-9117-x

An Intuitionistic Proof of a Discrete Form of the Jordan Curve Theorem Formalized in Coq with Combinatorial Hypermaps

Jean-François Dufourd

Received: 20 March 2008 / Accepted: 12 February 2009 / Published online: 6 March 2009

Abstract This paper presents a completely 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 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.

Keywords Formal specifications · Computer-aided proofs · Coq system · Computational topology · Planar subdivisions · Combinatorial hypermaps · Discrete Jordan Curve Theorem

1 Introduction

This paper presents formal specifications and proofs assisted by the Coq system in combinatorial topology. It deals with surface subdivisions, planarity and a discrete version of the famous Jordan Curve Theorem. 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 Jordan Curve Theorem we deal with states that in a finite subdivision of the plane, breaking a ring R of faces increases by one 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

This research is supported by the "white" project GALAPAGOS, French ANR, 2007.

J.-F. Dufourd (✉)
UFR de Mathématique et d'Informatique, Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection (UMR CNRS-ULP 7005), Université de Strasbourg, Pôle API, Boulevard Sébastien Brant, 67400 Illkirch, France
e-mail: dufourd@dpt-info.u-strasbg.fr

in computational geometry and discrete geometry for imaging, where connection is the essential information [14, 25]. 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* [7, 32]. We thus propose a purely combinatorial proof of Jordan Curve Theorem based on this structure. The hypermap framework is entirely formalized and the proofs are developed interactively and verified by the Coq proof assistant [5]. 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 Jordan Curve Theorem and a simple constructive proof with algorithmic extensions. It is also a large benchmark for the software specification we have been developing for many years with the intention to safely use map models in geometric modeling and computer imagery [3, 9, 10].

Note that a first part of this paper resumes a hypermap specification and a formalization of the genus theorem and of the planarity, or its equivalent, the Euler formula, we have published in [11]. However, the present specification is lighter and goes further with the traversal of permutation orbits. It allows to obtain new results, like the periodicity of orbits, the symmetry in their traversal, and a complete characterization of the planarity. Moreover, a preliminary version of the proof of the discrete Jordan Curve Theorem has been presented in [12]. But, in the corresponding work, the theorem statement is only complete for the combinatorial oriented maps (a subclass of the hypermaps), its formalization is rather far from the mathematical expression, and the proof is too long.

The rest of the paper is structured as follows. Section 2 summarizes related work. Section 3 recalls some mathematical materials. Section 4 proposes basic hypermap specifications. Section 5 proves constructive criteria of hypermap planarity and connectivity. Section 6 inductively specifies the rings and their properties. Section 7 proves the discrete Jordan Curve Theorem. Section 8 presents validations of our result. Section 9 discusses our approach of formalization and Section 10 concludes. The useful Coq features are reminded and the whole process is described along the paper, but the full details of the proofs are omitted. Appendices A to D give some technical explanations and proof scripts, but the latters are only readable by Coq experts.

2 Related Work

The Jordan Curve Theorem is a result of classical plane topology, first stated by C. Jordan in 1887, but correctly and (fast) completely proven by O. Veblen, only in 1905 [33]. It is generalized on several occasions, e.g. to higher dimensions or to homeomorphisms, until in the 1960's. Classical proofs are based on compacity, differential calculus or complex analysis.

In 1979, W.T. Tutte proposes operations and properties of combinatorial maps, e.g. planarity and Euler Formula, defines rings and proves a discrete version of the Jordan Curve Theorem [31]. Our theorem statement is comparable, but our framework is modeled differently and all our proofs are formalized and computer-assisted. In 1983, R. Stahl proposes a statement and a proof which is rather close and is the basis of further developments in computer science [30]. Since the late 1980's, a

lot of other discrete forms of the theorem are based on graphs [34], mainly dedicated to digital topology in the plane or in the space [4, 14, 25, 26].

Indeed, the interest for discrete versions of the Jordan Curve Theorem has increased during the last years [23, 29]. One reason is the irresistible development of digital computations on geometric objects in applicative domains, e.g. medical imagery, molecular modeling, robotics, geographical information systems, computer-aided design. However, these calculations mainly use floating numbers and are highly error-prone due to numerical approximations. This is why one observes a flowering of models of discrete topology and geometry based on natural numbers, which allow exact calculations. In these frameworks, general concepts of curve and (meshed) surface have no common definition. They are currently approached by (multi-, hyper-)graphs or (hyper-)maps. But, whatever the model, the problems of localization and separation of points by curves in a topological space is crucial. It is essential to algorithmically and correctly respond to the question: on a surface, is a given point (pixel or voxel) on, inside or outside a given closed curve? Consequently, discrete equivalents of the surface and curve notions must be defined, and Jordan Curve Theorems stated and correctly proven. In addition, the validity of such a theorem is the insurance that a discrete model is convenient. Moreover, basic computational algorithms, e.g. to build convex hulls or Voronoï-Delaunay diagrams, must be translated is such models. It would be nice to also prove their correctness. Finally, using hypermaps is certainly a good way to federate an interesting class of discrete topological-geometrical models.

In 2003, in the way of M. Yamamoto et al. [35], 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 Jordan Curve Theorem. In 2005, A. Kornilowicz completes for the MIZAR project a semi-automated classical proof of a continuous form of Jordan Curve Theorem in an Euclidean space [24]. In 2005 also, on his way towards the proof of the Kepler conjecture in the Flyspeck projet [19, 20], T. Hales proves the theorem with the HOL Light system in two parts, first for simple polygons, then for general curves. The formal proof for polygons is based on intuitive considerations coming from the graph-paper geometry. The formal proof for the general case uses an approximation argument and the Kuratowski characterization of planarity [21].

In 2005 always, G. Gonthier et al. achieve the impressionnant proof of the Four Colour Theorem using Coq. Plane subdivisions are described by hypermaps, and Euler's Formula is used as a global planarity criterion [15, 17]. A local criterion, called *hypermap Jordan property*, is proven equivalent, and operations to glue and cut hypermaps are defined. 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 appears that, if hypermap cutting is well defined, our form of Jordan Curve Theorem is not explicitly proven there. We will discuss the resemblances and differences with our approach in Section 8. In 2006, in his work about the localization of categories, C. Simpson formalizes in Coq the notion of graph [28]. He particularly pays attention to the definition and manipulation of paths, which play an important role in the composition of arrows in categories, as well as in the traversal of the hypermap orbits [11, 15, 17]. Finally, since 1999, we carry out experiments with Coq for combinatorial map models of space subdivisions [8, 10, 11].

The logical support of the Coq system is the Calculus of Inductive Constructions, or CiC [6, 27], which is a higher-order intuitionistic logic based on type theory, λ-calculus and induction. Proofs are seen as typed lambda-terms according to the Curry-Howard isomorphism. Inductive types are defined in CiC as presentations of algebraic theories with constructors. Syntactical term equality is offered. However, the definition of the exact types we need use the expression of invariants and their satisfaction thanks to preconditions. The specification language Gallina, the system libraries [5] and the tactics have provided an appropriate support for all our studies. For a first glimpse into the Coq system, the reader can follow the on-line tutorial [22]. All the system features are detailed in [5]. For a comprehensive substantial Coq presentation, more oriented towards program certification, the reader can refer to [2].

3 Mathematical Aspects

Definition 1 (Hypermap)

(1) 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.
(2) 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* together.

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 (maybe a simple line segment) 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. If $M = (D, \alpha_0, \alpha_1)$ is a hypermap, its cells can be combinatorially defined, mainly through the classical notion of orbit.

Definition 2 (Orbits and hypermap cells)

(1) Let f be a permutation in D. The *orbit* of $x \in D$ for f is the dart sequence $\langle f \rangle(x) = (x, f(x), f^2(x), \ldots, f^{p-1}(x))$, where p, called the *period* of the orbit, is the smaller integer such that $f^p(x) = x$.
(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 $\phi = \alpha_1^{-1} \circ \alpha_0^{-1}$.

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

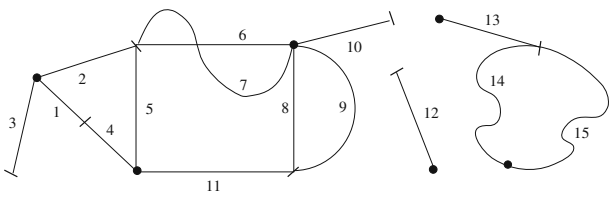


Fig. 1 An example of hypermap

(3) The *(connected) component* of x in hypermap M, denoted by $\langle \alpha_0, \alpha_1 \rangle(x)$, is the connected component of M viewed as a 2-multigraph on D equipped with the two functional binary relations α_0 and α_1.

In Fig. 1 the hypermap contains 7 edges (strokes), 6 vertices (bullets), 6 faces and 3 components. For instance, $\langle \alpha_0 \rangle(5) = (5, 6, 7, 2)$ is the edge of dart 5, $\langle \alpha_1 \rangle(5) = (5, 4, 11)$ its vertex. Faces are defined, through ϕ, for a dart traversal also 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 5 is $\langle \phi \rangle(5) = (5, 1)$ and the (external) face of 13 is $\langle \phi \rangle(13) = (13, 14)$. Let d, e, v, f and c be the numbers of darts, edges, vertices, faces and components of M.

Definition 3 (Euler characteristic, genus, planarity, Euler formula)

(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*.
(4) A planar hypermap satisfies the *Euler formula*: $\chi = 2 * c$.

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

Theorem 1 (of the Genus)

(1) *χ is an even integer.*
(2) *g is a non-negative integer.*

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 [18]. 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. $\langle \phi \rangle(7) = \{7, 10, 9, 4, 3, 2, 6, 11\}$ 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.

3.1 Double-Links, Ring of Faces and Jordan Curve Theorem

To state the version of Jordan Curve Theorem we will prove in a hypermap $M = (D, \alpha_0, \alpha_1)$, we need some other concepts.

Definition 4 (Double-link and adjacencies)

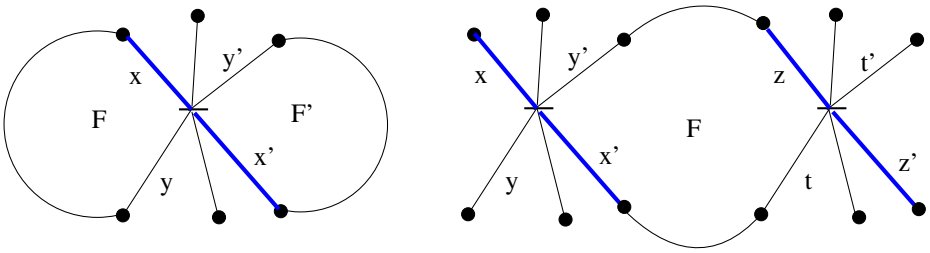
(1) A *double-link* is a pair of darts (x, x') where x and x' are distinct and belong to the same edge.
(2) The faces F and F' of M are said to be *adjacent by the double-link* (x, x') when $y = \alpha_0(x)$ is a dart of F and $y' = \alpha_0(x')$ a dart of F'.
(3) The double-links (x, x') and (z, z')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

These notions are illustrated in Fig. 2 where $t = \alpha_0(z)$ and $t' = \alpha_0(z')$. Hence, a double-link (x, x') prepares a cutting in an edge which entails the merging of two faces, one being incident to y and the other to y'. We choose a face adjacency *by an edge* rather than *by a vertex* as does W.T. Tutte [31]. In fact, due to the homogeneity of dimensions 0 and 1 in a hypermap, both are equivalent.

Definition 5 (Ring of faces) 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 $i \neq j$.

This notion simulates a Jordan curve represented in dotted lines in Fig. 3 above for $n = 4$, with the particular case $x_3 = y'_3$ and $y_3 = x'_3$, where E_3 is reduced to two darts. Of course, such a sequence of double-links, can be viewed as the sequence of the adjacent faces F_i they separe, from where the name "ring of faces". Note that W.T. Tutte uses the term *circular belt*, for this notion [31]. Then, we define the *break along a ring*, illustrated in Fig. 3.



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.

Fig. 2 An example of double-link and faces adjacent by it

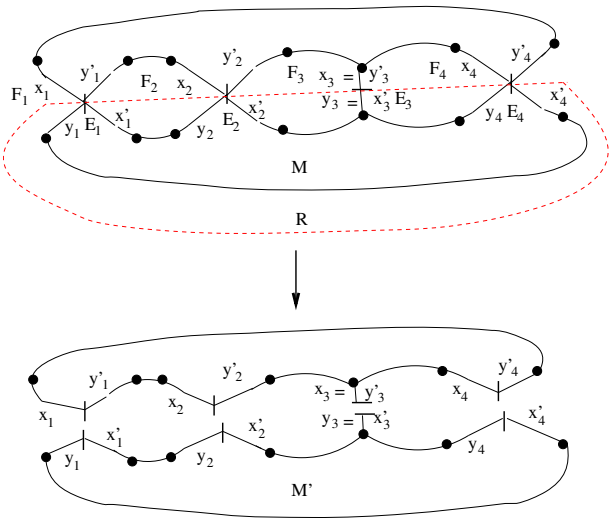


Fig. 3 Ring R of length $n = 4$ in M and break of M along R giving M'

Definition 6 (Break 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*.

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

Theorem 2 (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 $c' = c + 1$.*

4 Hypermap Specifications

4.1 Preliminary Specifications

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

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

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.
```

4.2 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. 4 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. In the following, the use of the constructors will be constrained by preconditions to avoid meaningless free maps. The corresponding subtype of the hypermaps will be characterized by an invariant, called `inv_hmap`, systematically used in conjunction with `fmap` (Section 4.3).

Next, *observers* of free maps can be defined. The predicate `exd` expresses 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

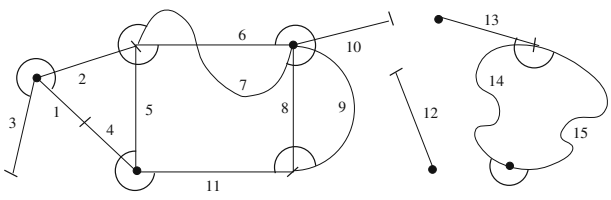


Fig. 4 Hypermap example with its incompletely linked orbits

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 partial version, denoted A, of operation α_k of Definition 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 (non-nil), with the decidabilities succ_dec and pred_dec. In hypermap m of Fig. 4, A m zero 2 = 5, A m zero 6 = nil, succ m zero 2, ~succ m zero 6, A_1 m one 2 = 1. Note that ~ is synonymous with not. 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 [31]. For instance, in Fig. 4, top m one 2 = 3, bottom m one 2 = 1, cA m one 3 = 1, cA_1 m one 1 = 3.

Finally, *destructors* are also recursively defined. First, D m z *deletes* the latest insertion of dart z by I. Second, B m k z (resp. B_1 m k z) *breaks* the latest k-link forward (resp. backward) inserted for dart z by L, if any. On a drawing, the effect is to remove an arc of circle which symbolises a k-link, if any, and nothing otherwise.

4.3 Hypermaps

Preconditions written as predicates are introduced for I and L:

```
Definition prec_I(m:fmap)(x:dart):Prop:=
 x <> nil /\ ~ exd m x.
Definition prec_L(m:fmap)(k:dim)(x y:dart):Prop:=
 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 them, 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 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. 4. 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). So, in Fig. 4, F m 4 = nil, cF m 4 = 3. Properties similar to the ones of A, cA are proven for F, cF and their inverses F_1, cF_1.

4.4 Orbits

Testing if a path exists from a dart to another in a hypermap orbit for a 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 1 and 2) and their inverses. That is why a *signature* Sigf with formal parameters f, f_1 and their properties – being permutations inverse of each other – is first defined.

Then, 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 expresses 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. Note that the definition of paths in orbits by iteration of f is very different from the definition using *n*-tuples given by C. Simpson for graphs [28] but is close to the definition of orbits of [15].

Appropriate modules, called MA0, MA1 and MF, are written to instantiate for (cA m zero), (cA m one), (cF m), and their inverses, definitions and properties of f and f_1. 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. Details are in Appendix A. In the following, MA0.expo is abbreviated into expe, and MF.expo into expf. For instance, in Fig. 4, expe m 6 9, ~expf m 6 5, expf m 1 5, ~expf m 5 3. Finally, a binary relation eqc stating that two darts belong to the same component is easily defined by induction. For instance, in Fig. 4, we have eqc m 1 5 and ~eqc m 1 13. We quickly prove that (eqc m) is a *decidable equivalence*.

4.5 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, e.g. `nc`, which plays a pre-eminent role in the following:

```
Fixpoint nc(m:fmap):Z :=
  match m with
     V => 0
   | I m0 x => nc m0 + 1
   | L m0 _ x y => nc m0 - if eqc_dec m0 x y then 0 else 1
  end.
```

Euler characteristic `ec`, `genus` and `planar` derive. The Genus Theorem is obtained as a corollary of the fact that `ec` is even and satisfies `2 * (nc m) >= (ec m)` [11]. Here, the Euler Formula (for any number `(nc m)` of components) is just a rewriting of the planarity property. 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.
Lemma Euler_Formula: forall m:fmap,
  inv_hmap m -> planar m -> ec m / 2 = nc m.
```

Note that, in the current state, it is impossible to directly obtain these characteristics from a given hypermap term by Coq reductions. That is because some predicates, i.e. `expf_dec` or `eqc_dec`, are not really computable. Even in a program extraction in Caml, they have to be translated in tractable forms. But, once a Coq specification is tested by the proof of crucial properties, this translation is more easy, as it is showed in our image segmentation work [10].

5 Planarity and Connectivity Criteria

A consequence of the genus theorem is a completely constructive *criterion of planarity*, when one correctly *links* with `L` at dimensions 0 or 1:

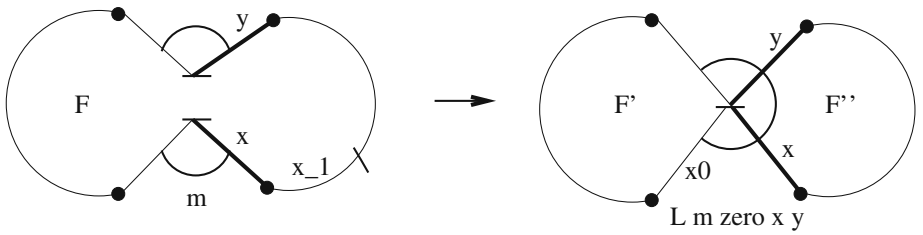
```
Theorem planarity_crit_0: forall (m:fmap)(x y:dart),
   inv_hmap m -> prec_L m zero x y ->
(planar (L m zero x y) <->
    (planar m /\ (~ eqc m x y \/
     expf m (cA_1 m one x) y))).
```

```
Theorem planarity_crit_1: forall (m:fmap)(x y:dart),
   inv_hmap m -> prec_L m one x y ->
  (planar (L m one x y) <->
 (planar m /\ (~ eqc m x y \/ expf m x (cA m zero y)))).
```

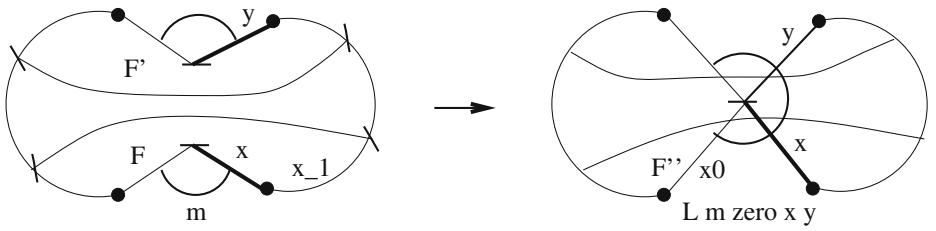
So, at dimension 0 for instance, the planarity of `m` is preserved for `(L m zero x y)` *iff* one of the following two conditions holds: (1) `x` and `y` are not in the same component of `m`; (2) `x_1 = (cA_1 m one x)` and `y` are in the same face of `m`, i.e. the linking operates *inside the face* containing `y`. Figure 5 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 (return the arrows in Fig. 5), but it is much more difficult to obtain. In our specification, it depends on results about the *invariance of the number of faces* when doing the permutation of two linkings. Some of them are particularly long to prove. For instance, the following theorem represents about 9000 Coq lines,



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''.

Fig. 5 Linking at dimension 0 (**a**, **b**)

more than the quarter of our total development, because of the number of cases to considerer:

```
Theorem nf_L0L1: forall (m:fmap)(x y x' y':dart),
  let m1:= L (L m zero x y) one x' y' in
  let m2:= L (L m one x' y') zero x y in
inv_hmap m1 -> nf m1 = nf m2.
```

It would be fruitful to relate the above constructive/destructive criteria with the static one of G. Gonthier et al. [15, 17]. 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`, where `x0 := bottom m zero x`:

```
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).
```

6 Rings of Double-Links

6.1 Modeling a Double-Link

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

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

Now, we have to model the conditions required for list `l` to be a *ring* in hypermap `m`. Translating Definition 5, we have first to define the predicate expressing that a pair of darts `(x,x')` is a double-link in a hypermap `m` by:

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

Here, `expe m x x'` (unfolded into `MA0.expo m x x'`) expresses that there is a path from `x` to `x'` in an edge of `m`, i.e. that both darts are in the same edge since `expe m` is a decidable equivalence. Next, the fact that `l` is really a list of double-links is easily translated into:

```
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.
```

We must write the four conditions of Definition 5, 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 /\ double_link_list m l /\
      pre_ring0 m l /\ pre_ring1 m l /\
           pre_ring2 m l /\ pre_ring3 m l.
```

6.2 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 containing `x`, `pre_ring0 m l` is recursively defined to impose that all edges in `l` are distinct: Condition (0) of Definition 5.

```
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.
```

6.3 Ring Condition (1): *Continuity*

We define *adjacency by a face* of `m` for two double-links `(x,x')` and `(xs,xs')` by:

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

So, the predicate `pre_ring1 m l` recursively specifies that two successive faces in `l` are adjacent: Condition (1) of Definition 5:

```
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.
```

6.4 Ring Condition (2): *Circularity*, or *Closure*

The predicate `pre_ring2 m l` specifies that the *last* and *first* double-links in `l` are adjacent by a face: Condition (2) of circularity in Definition 5:

```
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.
```

6.5 Ring Condition (3): *Simplicity*

Let `(x,x')` and `(xs,xs')` be two double-links in `m`. The predicate `distinct_face` specifying that the faces of `y:= cA m zero x` and `ys:= cA m zero xs` are distinct is easy to write, as well as the predicate `distinct_face_list m x x' l0` expressing that all the faces of `l0` are distinct from the face of `cA m zero x`:

```
Definition distinct_faces(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.

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

Finally, the predicate `pre_ring3 m l` says that all faces of `l` are distinct: Condition (3) of Definition 5:

```
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.
```

6.6 Breaking a Map Along a Ring of Faces

First, we define the break `Br1` of the single double-link `(x,x')` in the hypermap `m` by:

```
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'.
```

In fact, since x and x′ are distinct and intended to be in the same edge, there are only three cases, illustrated in Fig. 6:

- *Case* succ m zero x and succ m zero x′, i.e. x and x′ both have a 0-successor: first, the 0-link from x is broken, next (top m zero x) is 0-linked to (bottom m zero x) in order to restore the edge integrity, finally the 0-link from x′ is broken.
- *Case* succ m zero x and ~succ m zero x′: it is enough to break the 0-link from x.
- *Case* ~succ m zero x: it is easy to prove that succ m zero x′ holds, and the break of the 0-link from x′ is realized.

Hence, in the three cases, the edge is cut in two, exactly at x and x′. Expected properties of Br1 are fast proven, mainly the preservation of the hmap invariant, of the *planarity* and the *commutativity* of x and x′. Moreover, the effect of Br1 on A, cA, cF and their inverses is important in the following. We have for instance:

```
Lemma Br1_comm: forall (m:fmap)(x x':dart),
  inv_hmap m -> double_link m x x' ->
      Br1 m x x' = Br1 m x' x.

Lemma cA0_Br1:forall(m:fmap)(x x' z:dart),
  inv_hmap m -> double_link m x x' ->
    cA (Br1 m x x') zero z =
      if eq_dart_dec x z then cA m zero x'
```

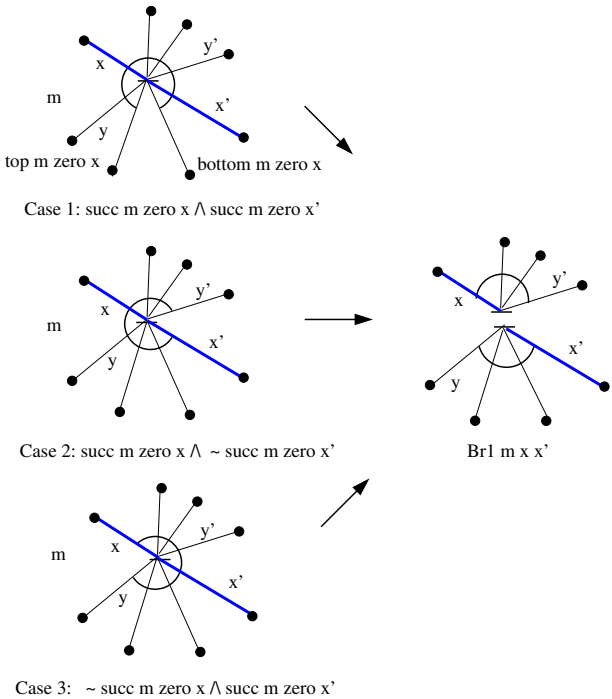


Fig. 6 Breaking an edge at a double-link

```
      else if eq_dart_dec x' z then cA m zero x
           else cA m zero z.
```

Note that an important part of these good results is due to the *observational equality* between `m` and `m0 := L (B m zero x) zero (top m zero x) (bottom m zero x)`, in other words to the fact that many properties of the k-orbits are preserved *when their opening is displaced*. For instance, we have:

```
Lemma cA_L_B: forall (m:fmap)(k:dim)(x z:dart),
 inv_hmap m ->
    let m0:= B (L (B m zero x) zero (top m zero x)
    (bottom m zero x)) in
  cA m0 k z = cA m k z.
```

All these properties allow to rewrite the *connectivity criterion* of Section 5 in a form which will be more easy to handle in the following (see Fig. 7 to have a configuration with disconnection):

```
Theorem disconnect_planar_criterion_Br1:
   forall (m:fmap)(x x':dart),
 inv_hmap m -> planar m -> double_link m x x' ->
   let y  := cA m zero x in
   let y' := cA m zero x' in
 (expf m y y' <-> ~eqc (Br1 m x x') x' y').
```

The effect of `Br1` on the number `nc` of connected components entails:

```
Theorem nc_Br1: forall (m:fmap)(x x':dart),
  inv_hmap m -> planar m -> double_link m x x' ->
    let y  := cA m zero x in
     let y' := cA m zero x' in
  nc (Br1 m x x') = nc m + if expf_dec m y y'
     then 1 else 0.
```

Finally, `Br m l` breaks hypermap `m` along the double-link list `l` in `m`. When `l` has the properties of a `ring`, this operation realizes the *break along a ring* in accordance with the mathematical definition (Section 3):

```
Fixpoint Br(m:fmap)(l:list){struct l}:fmap:=
 match l with
```

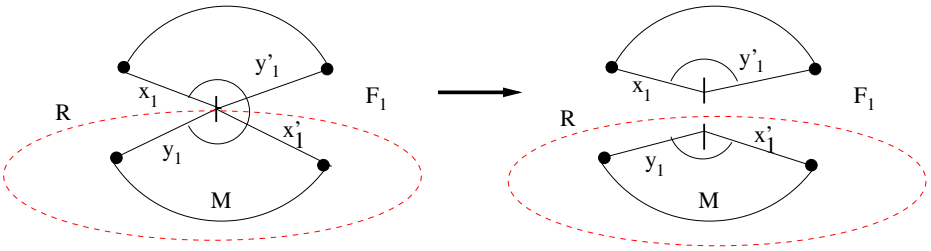


Fig. 7 Break along a ring of length 1

```
    lam => m
  | cons x x' l0 => Br (Br1 m x x') l0
end.
```

7 Proof of the Discrete Jordan Curve Theorem

The general principle of the Jordan Curve Theorem proof for a hypermap `m` and a ring `l` is a *structural induction* on `l`.

- The case where `l` is *empty* is immediately 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 x' lam`, as illustrated in Fig. 7. Then, we prove the following lemma as a direct consequence of the previous theorem `nc_Br1` (The complete proof is proposed in Appendix B for Coq practicers):

```
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.
```

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

```
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'.
```

In this case, thanks to `disconnect_planar_criterion_Br1` (Section 5), the lemma entails that the break of the first ring double-link does never disconnects 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 ring properties are preserved after the first break in `l`:

```
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)).
```

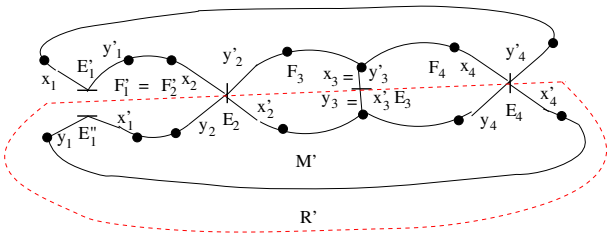


Fig. 8 Map M and ring R of Fig. 3 transformed into $M' = (Br1\ M\ x_1\ x'_1)$ and $R' = (tail\ R)$

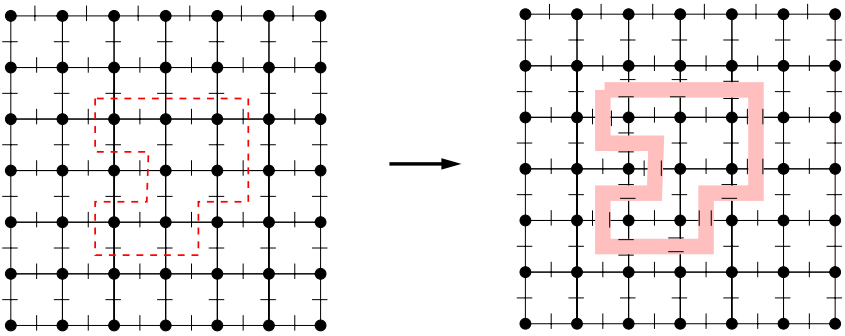
Figure 8 illustrates the lemma for the ring in Fig. 3 where $n > 1$, i.e. with no disconnection after `M' := Br1 M x1 x'1`: edge E_1 is split into edges E'_1 and $E"_1$, faces F_1 and F_2 are merged into $F'_1 = F'_2$, and R' remains a ring. The most difficult is to prove the parts of the lemma concerning `double_link_list` and the `pre_ringk`, for `k` $= 0, \ldots, 3$. The five proofs are led by induction on `l` in separate lemmas. Details are given for each proof in Appendix C (for Coq experts).

Finally, from `Jordan1` and `pre_ring_Br1` above, we have the expected result by a quick reasoning by *two successive inductions* on `l`, where all the links are broken one by one from the first (see Appendix D, for Coq experts):

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

8 Validity of the Theorem, Case of the Oriented Maps

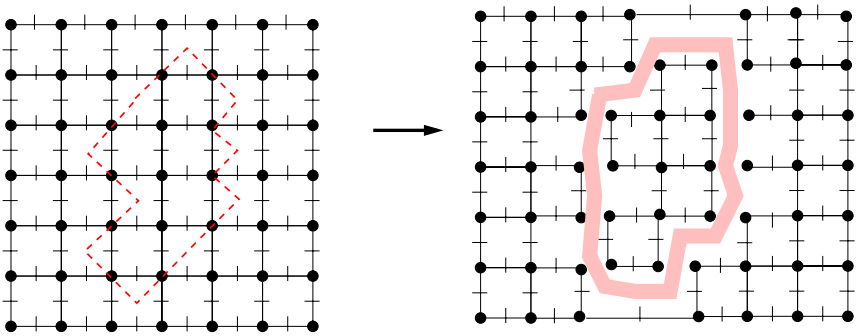
It is clear that, provided any mathematical hypermap M and mathematical ring R conform to Definitions 1 and 5, we can always describe them as terms of our specification in order to apply our theorem. Conversely, with any hypermap term, all the rings can directly be written as terms. Hence, our ring specification and our



a. A grid map and a ring of faces

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

Fig. 9 Application of the Jordan Curve theorem in a pixel grid (**a**, **b**)



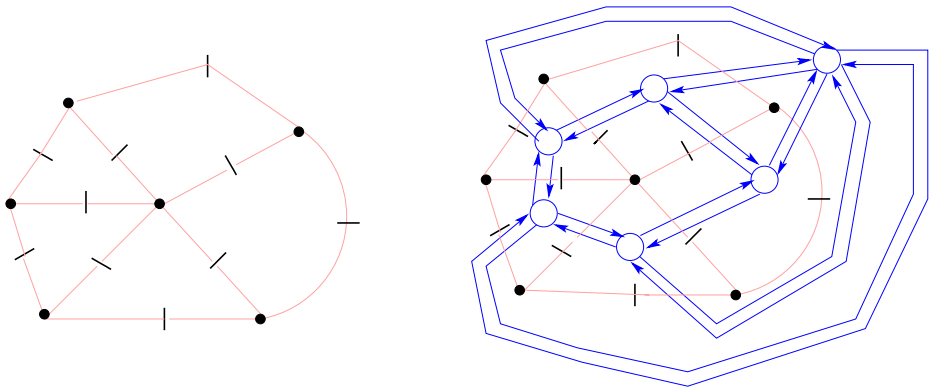
a. Grid map and symmetric ring of faces b. Grid map in two components after the break along the ring

Fig. 10 Application of the Jordan Curve Theorem in a pixel grid in order to break vertices (**a**, **b**)

Jordan Curve Theorem formalization are *complete* with respect to our mathematical definitions.

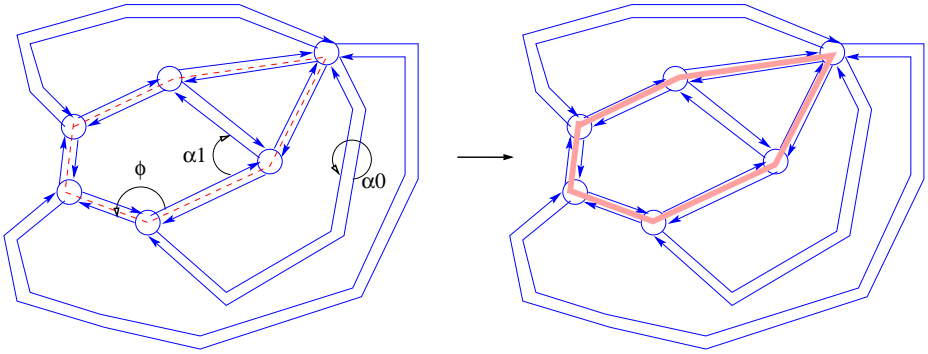
In the particular case where α_0 is an *involution* (i.e. for any dart $x \in D, \alpha_0 \circ \alpha_0(x) = x$), the hypermap $M = (D, \alpha_0, \alpha_1)$ is *a combinatorial oriented map*, in short *a map* [32]. Then, each edge is composed of at most 2 edges, and the embedding is more intuitive than for general hypermaps. Indeed, each edge is embedded into an open Jordan arc, and, when it exactly contains two darts opposed by α_0, they can be considered as 2 *half-edges* having opposite orientations. For instance, a pixel grid can be modeled as in Fig. 9a. The direct application of the Jordan theorem in this grid along the ring in Fig. 9a is illustrated by Fig. 9b. The finicky reader will regret the cutting of pixel outlines together with the appearance of hanging darts. Of course, the latters can be erased by a second pass or by an improvement of the `Br` operation.

One can also apply the theorem symmetrically with respect to the dimensions, that is by exchanging the roles of α_0 and α_1. Then, the operation `Br` would break the



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

Fig. 11 Conventional representation (**a**) and primal-dual representations (**b**) of a map



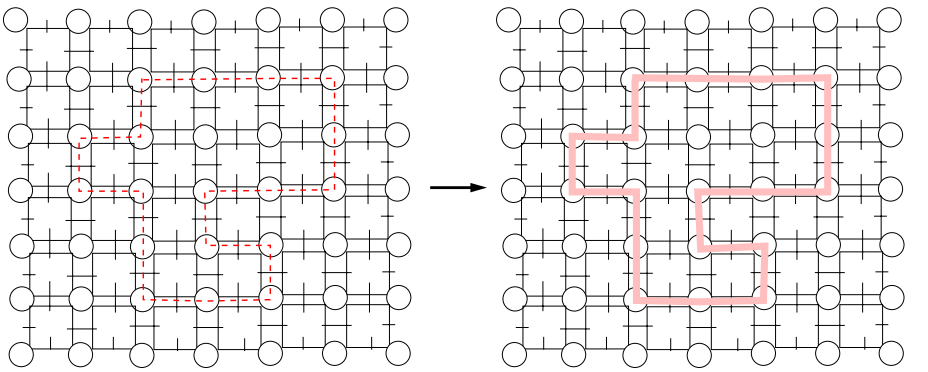
(a) Dual representation of a map and ring (b) Dual map in two components after the break along the ring

Fig. 12 Dual representation and application of the discrete Jordan Curve Theorem (**a**, **b**)

vertices instead of the edges. This is illustrated in Fig. 10 for our pixel grid. However, the same finicky reader would probably emit the same objection about pixel cutting and hanging darts, and the remedy could be analogous.

Finally, there is a third way to tackle the question, namely the *duality*. A combinatorial oriented map $M = (D, \alpha_0, \alpha_1)$ is *dually represented* if the representations of α_1 and ϕ, i.e. of vertices and faces, are exchanged. In this case, consider the conventional (primal) representation of a map in Fig. 11a. Its dual representation is superposed in Fig. 11b, and isolated in Fig. 12a. Note that, in the dual, each dart is represented by an arrow, and an edge with 2 darts by 2 arrows in opposite orientations. The application of the Jordan Curve Theorem if illustrated in Fig. 12b.

Note that the characteristics are the same on the dual and on the primal. So, when one of them is planar, the other is planar as well. In this case, a pixel grid with a ring of faces can be represented as in Fig. 13a. The deconnection by application of the Jordan Curve Theorem is illustrated in Fig. 13b. Then, the integrity of the pixels is entirely preserved.



a. A connected grid map in dual representation and a ring b. Grid map with two components after the break along the ring

Fig. 13 Application of the Jordan Curve Theorem in a dual grid (**a**, **b**)

9 Discussion on the Formalization

We are often asked about the similarities between the approach of G. Gonthier et al. [15, 17] and ours. Clearly, both approaches are in discrete combinatorial topology and use hypermaps to deal with the topology of plane subdivisions. In this section, we shall only highlight the key differences between our two formalizations of the (discrete) Jordan Curve Theorem.

They appear from the very *beginning of the specification*. In [15], a basic library contains definitions and properties of finite generic sets, sequences and relations, including the notions of path, cycle, orbit, connectivity, etc. Then, a hypermap is statically represented by a finite dart set equipped with 3 functions – for α_0, α_1 and ϕ – which satisfy an identity – $\alpha_1 \circ \alpha_0 \circ \phi = Id$ –. With such a definition, properties involving permutations, genus or planarity, are easy to state and to prove by classical reasoning. Updating operations inspired by S. Stahl's work are defined later. Thus, this approach is guided by a classical static representation of hypermaps considered as 3-multigraphs. In another way, we represent hypermaps from scratch by a free map algebra defined by induction on 3 constructors `V`,`I` and `L`. At this level we do not even have the concept of finite set. We have to preserve an invariant thanks to preconditions. Our constructors are atomic operations whose properties are proved by induction on the free map structure. The constructors only deal directly with 2 permutations – α_0 and α_1 –, the third one – ϕ – being obtained by composition. The permutations appear like observers, the definition and the properties of which are obtained by induction. Thus, our approach is guided by a technique of abstract algebraic specifications.

In his work, G. Gonthier introduces and uses numerous *concepts*, as do mathematicians. This is fully justified by the broad objective to reach the Four Color Theorem. It appears that we are more cautious as for the new notions, as do programmers to avoid the implementation of multiple types and predicates. Indeed, our objective is not only to prove theorems, but to be able to derive reliable software. More about the model, note that, in [15], some restrictions on the hypermaps exclude fixpoints, bridges and non-connectivity, probably always for the Four Color Theorem. Conversely, we do not impose anything else to our original hypermap definitions.

Regarding the *underlying logic*, in G. Gonthier's work, the proof of the Four Color Theorem appears to be purely intuitionistic, whereas excluded middle and functional extentionality are necessarily used ([15], see footnotes in p. 4) in peripheric arithmetic proofs involving the real plane. Moreover, a big layer of sophisticated technics and tactics is defined to make easier the proofs. The *reflection* is one of the preferred reasoning modes [16]. In our work, the reasoning is fully intuitionistic as well, and remains elementary without any new tactic. As sophisticated tools, only modules and functors are used to deal with orbits using finite dart sets.

Concerning the *Euler formula*, which is synonymous of planarity, G. Gonthier presents a first proof for a rectangular regular grid, then generalizes it to the embedded maps by approximation techniques ([15], p. 49–50). He emphasizes the fact that the (classical continuous) Jordan Theorem is not used in the reasoning, what could be a real pitfall in such a work. In another way, we have proved a criterion of planarity when building/destructing a hypermap using `V`, `I` and `L` and

their inverse operations. So, the Euler formula is exactly verified by some hypermaps we characterize completely in a constructive way.

G. Gonthier proposes an alternative to the Euler formula as planarity criterion, he calls *hypermap Jordan property*. It is based on the definition of *contour paths*, which comes from [30] and allows to specify how to glue two components, thanks to a *patch* operation. This operation needs some smaller operations of *Walkup* to split or merge orbits while adding or removing darts. Operation *patch* is inversible, the inverse allowing to cut a hypermap in two parts. The latters are a disk (*inside*), the border is an edge cycle, and a remaining (*outside*), the border is a node orbit.

In our approach of the Jordan Curve Theorem, we consider a definition of *ring* of faces adjacent by double-links, and a cutting based on the `Br1` and `Br` operations. We do not define two peculiar parts, and their extraction, in accordance with future programming. The important information is the disconnection, i.e. the increasing by 1 of the number of components. Note that there is no need to start with a connected hypermap to obtain this result, since a ring is necessarily connected.

In fact, the ring notion is defined by G. Gonthier in an addendum to the proof of the Four Color Theorem proof (file `rjordan.v`) and a notion of *Moebius band* is defined. The absence of Moebius band in a hypermap is proven equivalent to the Jordan property. However it appeared that it was too difficult to deal with the ring notion in the Four Color Theorem, and the concept of contour path has been taken to play an alternative role [15] (p. 21). Neverthless, the patch inverse could be seen as an analogous of our `Br` operation, as our examples on dual maps suggest it.

In [15], the proof of the equivalence between the Euler formula and the hypermap Jordan property (consequently, the absence of Moebius band) is sketched (p. 23-24) using an induction on dart removing. This equivalence is presented in [15] (p. 21) and [17] (Theorem 2) as a Discrete Jordan Curve Theorem. We do not need such an equivalence because our work is mainly based on our constructive/destructive planarity criteria.

Finally, thank to the *patch* inverse operation, a particular form of discrete Jordan Curve Theorem is stated and proven in G. Gonthier's work on the Four Color Theorem. We have formalized another particular form with rings which is closer to W.T. Tutte's one, and like him, we remain at a combinatorical level. Conversely, and that is valuable additional contribution of his work, G. Gonthier connects his combinatorial statement with the more intuitive classical Jordan Curve Theorem statement involving subsets of the real continuous plane.

10 Conclusion

We have presented a discrete statement of the Jordan Curve Theorem based on hypermaps and rings in the spirit of W.T. Tutte [31], 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, apart from the *minimization*

axioms which are induced by our inductive type definitions for `dim`, `fmap` and `list`. In fact, it is basically the same *constructive* framework as the one we have designed to develop geometric modelers via algebraic specifications [3]. 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. We used intensively its facilities to define inductive types, to deal with type polymorphism, to create and instantiate parameterized functions and modules, thank to the notion of signature, and to lead reasonings and constructions by structural and noetherian inductions.

The preexistent framework of hypermap specification represents about 25,000 lines of Coq, and the Jordan Curve Theorem development about 5,000 lines, including about 20 new definitions, and 40 lemmas and theorems. So, we have a rather complete basis to tackle any topological problem involving orientable surface subdivisions.

Besides new developments in combinatorial topology, the extensions of this work are in 2D or 3D computational geometry and geometric modeling by introducing embeddings [3, 9], and computer imagery by dealing (as in our examples) with pixels [10] or voxels [26]. Avantages of our kind of formalization and proofs are in a well design of safe operations allowing to precisely control the topological characteristics of geometric objects the deal with. In this respect, it is crucial that the implementation could be a direct consequence of the operations of the specification. We have often shown that our basic operations are immediately translatable in functional or imperative languages in an efficient way [3, 9, 10]. The fact that expected properties can be formally deduced from the specification enforces the safety of this approach. However, the correctness of the implementation must be checked, e.g. in the spirit of [13]. Another promising way is the extraction of certified functional programs from our constructive proofs [5].

Finally, W.T. Tutte proposes a generalization where the successive edges (for him vertices) do not need to be distinct in a ring, which he qualifies by *admissible* [31]. In fact, we guest that our condition `double_link_list` can be lightened by the two conditions:

(1) all the darts of the sequence are distinct;
(2) when any two double-links of the sequence, say `(x,x')` and `(z,z')`, are in the same edge, `x` and `x'` are *between* `z` and `z'` when they are traversed by `(cA m 0)` (*between* is a predicate we have formalized).

In this case, the discrete Jordan Curve Theorem is still valid in a more general form, which is far to be intuitive because this does not work in the continuous case. The formalized assisted corresponding proof is a next challenging task. Note that the condition of *admissibility* of W.T. Tutte is expressed by an analogue of *disjoint angles* which seems less general than our two conditions.

Acknowledgements The author sincerely thanks the two reviewers and the editors who made criticisms, requests and suggestions having allowed to complete and improve this work appreciably.

Appendix A

This appendix gives first the signature `Sigf`, next the structure of the module `Mf(M:Sigf)`. Finally, it presents modules instantiating module `Mf` for edges, vertices and faces:

```
(* SIGNATURE FOR PERMUTATIONS f AND f_1: *)
Module Type Sigf.
Parameter f : fmap -> dart -> dart.
     (* PERMUTATION *)
Parameter f_1 : fmap -> dart -> dart.
     (* INVERSE PERMUTATION *)
Axiom exd_f:forall (m:fmap)(z:dart),
     (* AXIOMS OF f AND f_1 *)
  inv_hmap m -> (exd m z <-> exd m (f m z)).
Axiom bij_f : forall m:fmap,
  inv_hmap m -> bij_dart (exd m) (f m).
Axiom exd_f_1:forall (m:fmap)(z:dart),
  inv_hmap m -> (exd m z <-> exd m (f_1 m z)).
Axiom bij_f_1 : forall m:fmap,
  inv_hmap m -> bij_dart (exd m) (f_1 m).
Axiom f_1_f : forall (m:fmap)(z:dart),
  inv_hmap m -> exd m z -> f_1 m (f m z) = z.
Axiom f_f_1 : forall (m:fmap )(z:dart),
  inv_hmap m -> exd m z -> f m (f_1 m z) = z.
End Sigf.

(* GENERIC MODULE, OR FUNCTOR, Mf,
GROUPING THE PROPERTIES OF f-ORBITS: *)
Module Mf(M:Sigf)<:Sigf
   with Definition f:=M.f
   with Definition f_1:=M.f_1...
Definition f:=M.f.
Definition f_1:=M.f_1.
Definition exd_f:=M.exd_f.
Definition exd_f_1:=M.exd_f_1.
Definition bij_f:=M.bij_f.
Definition bij_f_1:=M.bij_f_1.
Definition f_1_f:=M.f_1_f.
Definition f_f_1:=M.f_f_1.
...
Definition...(* about f or f_1 *)
...
Lemma...     (* about f or f_1 *)
...
Theorem...   (* about f or f_1 *)
...
End Mf.
```

```
(* SIGNATURE FOR A DIMENSION k: *)
Module Type Sigd.
Parameter k:dim.
End Sigd.

(* FUNCTOR FOR A PERMUTATION AT A DIMENSION Md.k: *)
Module McA(Md:Sigd)<:Sigf.
Definition f := fun(m:fmap)(z:dart) => cA m Md.k z.
Definition f_1 := fun(m:fmap)(z:dart) => cA_1 m Md.k z.
Definition exd_f := fun(m:fmap)(z:dart) => exd_cA m Md.k z.
Definition exd_f_1 := fun(m:fmap)(z:dart)
     => exd_cA_1 m Md.k z.
Definition bij_f := fun(m:fmap)(h:inv_hmap m)
     => bij_cA m Md.k h.
Definition bij_f_1 := fun(m:fmap)(h:inv_hmap m)
     => bij_cA_1 m Md.k h.
Definition f_1_f := fun(m:fmap)(z:dart) => cA_1_cA m Md.k z.
Definition f_f_1 := fun(m:fmap)(z:dart) => cA_cA_1 m Md.k z.
End McA.

(* INSTANCIATIONS OF Sigd FOR DIMENSIONS 0 AND 1: *)
Module Md0<:Sigd.
Definition k:=zero.
End Md0.

Module Md1<:Sigd.
Definition k:=one.
End Md1.

(* INSTANCIATIONS OF McA FOR DIMENSIONS 0 AND 1: *)
Module McA0:=McA Md0.
Module MA0:= Mf McA0.
Module McA1:=McA Md1.
Module MA1:= Mf McA1.

(* TO CATCH THE ORBITS IN FACES: *)
Module McF<:Sigf.
Definition f := cF.
Definition f_1 := cF_1.
Definition exd_f := exd_cF.
Definition exd_f_1 := exd_cF_1.
Definition bij_f := bij_cF.
Definition bij_f_1 := bij_cF_1.
Definition f_1_f := cF_1_cF.
Definition f_f_1 := cF_cF_1.
End McF.

Module MF:= Mf McF.
```

Appendix B

This appendix gives the complete script of the proof of the Jordan Curve Theorem in the case of a ring of length 1. It is intended to readers which are familiar with the syntax and tactics of Coq.

```
(* JORDAN CURVE THEOREM FOR A RING OF LENGTH 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.
unfold ring in |- *.
unfold pre_ring0 in |- *.
unfold pre_ring1 in |- *.
unfold pre_ring2 in |- *.
unfold pre_ring3 in |- *.
unfold double_link_list in |- *.
unfold double_link in |- *.
unfold distinct_face_list in |- *.
unfold distinct_edge_list in |- *.
unfold face_adjacent in |- *.
simpl in |- *. intros.
decompose [and] H1. clear H1 H2 H6 H5 H9 H3 H10 H12.
set (y := cA m zero x) in |- *.
set (y' := cA m zero x') in |- *.
fold y in H8. fold y' in H8.
rewrite nc_Br1 in |- *.
 fold y in |- *. fold y' in |- *.
   elim (expf_dec m y y'). tauto.
 intro. elim b. apply expf_symm. tauto.
 tauto. tauto.
unfold double_link in |- *. tauto.
Qed.
```

Appendix C

This appendix presents proof schemes corresponding to the preservation of the dart-link list property and of the Conditions (0) to (3), when breaking double-link `(x,x')` in hypermap `m`. This double-link is the first of the ring `l`, containing at least one element. We have to begin with Condition (0), because the preservation of the dart-link list property depends on it.

- **Condition (0): unicity**

We first prove a lemma which asserts that, after the break of the first double-link of a list with at least two elements, the second edge is distinct from all the edges in the list

obtained by removing the first two double-links. The result is obtained by induction on the list:

```
Lemma distinct_edge_list_Br1:
 forall(m:fmap)(x x' xs xs':dart)(l:list),
 inv_hmap m -> planar m ->
   pre_ring0 m (cons x x' (cons xs xs' l)) ->
     distinct_edge_list (Br1 m x x') xs l.
```

Then, always by induction on the list `l`, we obtain the expected result, i.e. the preservation of the property `pre_ring0` for `(Br1 m x x')` and `(tail l)`, `(x,x') := first l` being the first double-link in `l`:

```
Lemma pre_ring0_Br1: forall(m:fmap)(l:list),
 inv_hmap m -> planar m ->
   pre_ring0 m l ->
   let (x,x') := first l in
      pre_ring0 (Br1 m x x') (tail l).
```

• Invariant of the dart-link lists

We first prove a sufficient condition for the preservation of the existence of a path between two darts in an edge when operating with `B`:

```
Lemma expe_expe_B0: forall(m:fmap)(x z t:dart),
  inv_hmap m -> exd m x ->
  let m0 := B m zero x in
    ~ expe m x z -> expe m z t-> expe m0 z t.
```

Then, we prove a lemma which asserts that, after the break of the first double-link of a list, the remaining list always satisfies the invariant of the double-link list. The result is obtained by induction on the list. As we said above, `pre_ring0` figures in the premises:

```
Lemma double_link_list_Br1: forall(m:fmap)(l:list),
 inv_hmap m ->
   double_link_list m l -> pre_ring0 m l ->
     let (x,x') := first l in
        double_link_list (Br1 m x x') (tail l).
```

At this stage only, `l` being a true double-link list, the preservation of the *planarity* by `Br m l` can be obtained under condition (0):

```
Lemma planar_Br:forall(l:list)(m:fmap),
 inv_hmap m -> planar m -> pre_ring0 m l ->
   double_link_list m l ->
     planar (Br m l).
```

• **Condition (1): continuity**

We have first to prove a useful lemma asserting that, in the case where the break `(Br1 m x x')` entails no disconnection, i.e. where `~expf m y y'`, two darts in the same face of `m` are always in the same face of `(Br1 m x x')`:

```
Lemma expf_Br1:forall(m:fmap)(x x' z t:dart),
   inv_hmap m -> planar m ->
    double_link m x x' ->
    let y:= cA m zero x in
    let y':= cA m zero x' in
    ~expf m y y' ->
        (expf m z t -> expf (Br1 m x x') z t).
```

Then, the core of the reasoning is the fact that `(Br1 m x x')`, where `(x,x')` is the first double-link in the list, does not affect the *adjacency* of the two following faces. This reasoning concerns 3 successive faces, or representative darts in the list, with the fact that `(Br1 m x x')` can interfere with the last two. Then, by induction on the ring, we obtain the expected result, i.e. the preservation of the property `pre_ring1` for `(Br1 m x x')` and `(tail l)`, `(x,x'):= first l`, where `l` is a true double-link list satisfying Conditions (0) and (1):

```
Lemma pre_ring1_Br1: forall(m:fmap)(l:list),
  inv_hmap m -> planar m ->
 double_link_list m l ->  pre_ring0 m l -> pre_ring1 m l ->
     let (x,x') := first l in
     let y  := cA m zero x in
     let y' := cA m zero x' in
  ~expf m y y' -> pre_ring1 (Br1 m x x') (tail l).
```

• **Condition (2): circularity**

The following property, which expresses the *opening* of a face when breaking by `Br1` the first double-link of a list, turned out to be very useful:

```
Lemma expf_Br1_link:forall (m : fmap) (x x': dart),
  inv_hmap m -> planar m -> double_link m x x' ->
    let y :=cA m zero x in
    let y':=cA m zero x' in
  ~expf m y y' -> expf (Br1 m x x') y y'.
```

Thank to this property, always by induction on the ring, we obtain the expected result, i.e. the preservation of the property `pre_ring2`, for `(Br1 m x x')` and `(tail l)`, under all previous proven properties:

```
Lemma pre_ring2_Br1: forall(m:fmap)(l:list),
 inv_hmap m -> planar m ->
  double_link_list m l -> pre_ring0 m l ->
```

```
    pre_ring1 m l -> pre_ring2 m l ->
     let (x,x') := first l in
     let y  := cA m zero x in
     let y' := cA m zero x' in
  ~expf m y y' -> pre_ring2 (Br1 m x x') (tail l).L
```

The reasoning is a structural induction on `l` which is rather difficult because it takes in play the beginning and the end of `l`.

• **Condition (3): simplicity**

Here, a new lemma about the effect of `Br1` on `expf` is useful. It is rather difficult to prove using the characteristic property `expf_B0_CNS`:

```
Lemma not_expf_Br1:forall (m:fmap)(x x' z t:dart),
 inv_hmap m -> planar m -> double_link m x x' ->
   let y  := cA m zero x in
   let y' := cA m zero x' in
         (~expf m y z /\ ~expf m y' z
       \/ ~expf m y t /\ ~expf m y' t) ->
   ~expf m z t -> ~expf (Br1 m x x') z t.
```

Then, we obtain a result asserting that, for 4 consecutive double-links in the ring, `(x,x')` being the first one, `(Br1 m x x')` does not affect the fact that the last two, namely `(z,z')` and `(zs,zs')`, determine distinct faces:

```
Lemma distinct_faces_Br1:
 forall(m:fmap)(x x' xs xs' z z' zs zs':dart)(l:list),
  inv_hmap m -> planar m ->
    let l1:= cons x x' (cons xs xs' (cons z z'
              (cons zs zs' l))) in
    double_link_list m l1 ->
      pre_ring0 m l1 ->
        pre_ring3 m l1 ->
         face_adjacent m x x' xs xs' ->
   distinct_faces (Br1 m x x') z z' zs zs'.
```

Finally, by induction on the ring (Condition (2) being useless), we obtain in the planar case the expected result, i.e. the preservation of the property `pre_ring3` for `(Br1 m x x')` and `(tail l)`, `(x,x') := first l` being the first double-link in ring `l`:

```
Lemma pre_ring3_Br1: forall(m:fmap)(l:list),
 inv_hmap m -> planar m ->
     let (x,x') := first l in
   double_link_list m l ->
   pre_ring0 m l ->  pre_ring1 m l -> pre_ring3 m l ->
        pre_ring3 (Br1 m x x') (tail l).
```

It is sufficient to collect the five above results to immediately prove lemma `ring_Br1`, which leads to the general case of Jordan's Curve Theorem (Appendix D).

Appendix D

The general Jordan Curve Theorem is obtained by a quick reasoning by induction on the ring `l` using `Jordan1`, `ring1_ring3_connect` and `ring_Br1`:.

```
Theorem Jordan: forall(l:list)(m:fmap),
   inv_hmap m -> planar m -> ring m l ->
      nc (Br m l) = nc m + 1.
Proof.
induction l.         (* 1. FIRST INDUCTION ON l *)
 unfold ring in |- *. (* 1.1. CASE: l IS EMPTY *)
  simpl in |- *.
    tauto.
rename d into x.     (* 2.1. CASE: l IS NON-EMPTY *)
  rename d0 into x'.
 simpl in |- *.
  intros.
   induction l.      (* SECOND INDUCTION On l *)
 simpl in |- *.      (* 2.1.1. CASE: (new) l IS EMPTY:
                       APPL. OF Jordan1 *)
   generalize (Jordan1 m x x').
   simpl in |- *.
    tauto.
rename d into xs.    (* 2.1.2. CASE: (new) l IS NON-EMPTY *)
  rename d0 into xs'.
  set (y := cA m zero x) in |- *.
  set (y' := cA m zero x') in |- *.
  assert (~ expf m y y').
  unfold y in |- *. unfold y' in |- *. unfold ring in H1.
   apply (ring1_ring3_connect m x x' xs xs' l). tauto.
  tauto. tauto. tauto. tauto.
rewrite IHl in |- *.   (* USE OF THE INDUCTION HYPOTHESIS *)
 rewrite nc_Br1 in |- *. fold y in |- *. fold y' in |- *.
    elim (expf_dec m y y').   (* a. CASE: y AND y' ARE IN
                                 THE SAME FACE)
    tauto.
  intro. omega. tauto. tauto.
 unfold ring in H1.
   simpl in H1. tauto.
apply inv_hmap_Br1.          (* b. CASE: y AND y' ARE NOT
                                 IN THE SAME FACE)
   tauto.
apply planar_Br1.
  tauto. tauto.
unfold ring in H1.
  simpl in H1. unfold double_link in H1. tauto.
generalize (ring_Br1 m (cons x x' (cons xs xs' l)) H H0 H1).
  simpl in |- *. tauto.
Qed.
```

References

1. Bauer, G., Nipkow, T.: The 5 colour theorem in Isabelle/Isar. In: Theorem Proving in HOL Conf. LNCS, vol. 2410, pp. 67–82. Springer, New York (2002)
2. Bertot, Y., Castéran, P.: Interactive theorem proving and program development—Coq'art: the calculus of inductive construction. In: Text in Theoretical Computer Science, An EATCS Series. Springer, New York (2004)
3. Bertrand, Y., Dufourd, J.-F.: Algebraic specification of a 3D-modeler based on hypermaps. Graph. Models Image Process. **56**, 1 29–60 (1994)
4. Chen, L.: Note on the discrete Jordan Curve Theorem. In: SPIE Conf. on Vision Geometry VIII, vol. 3811, pp. 82–94. SPIE, Bellingham (1999)
5. The Coq Team Development-TypiCal Project: The Coq proof assistant reference manual—version 8.1. INRIA, Le Chesnay http://coq.inria.fr/doc-fra.html (2007)
6. Coquand, T., Huet, G.: Constructions: a higher order proof system for mechanizing mathematics. In: EUROCAL. LNCS, vol. 203. Springer, New York (1985)
7. Cori, R.: Un code pour les graphes planaires et ses applications. Astérisque **27** (1970) (Société Math. de France)
8. Dehlinger, C., Dufourd, J.-F.: Formalizing the trading theorem in Coq. Theor. Comp. Sci. **323**, 399–442 (2004)
9. Dufourd, J.-F., Puitg, F.: Functional specification and prototyping with combinatorial oriented maps. Comput. Geom. Theory Appl. **16**, 129–156 (2000)
10. Dufourd, J.-F.: Design and certification of a new optimal segmentation program with hypermaps. Pattern Recogn. **40**, 2974–2993 (2007). doi:10.101b/j.patcog.2007.02.013
11. Dufourd, J.-F.: Polyhedra genus theorem and Euler formula: a hypermap-formalized intuitionistic proof. Theor. Comp. Sci. **403**, 133–159 (2008). doi:10.101b/j.tcs.2008.02.012
12. Dufourd, J.-F.: Discrete Jordan Curve Theorem: a proof formalized in Coq with hypermaps. In: Weil, P. (ed.) Symp. on Theoretical Aspects on Computer Science, 12 pp (2008). doi:hal-archives-ouvertes.fr/hal-002211501~v1
13. Filliâtre, J.C.: Formal proof of a program: FIND. Sci. Comput. Program. **24**(3), 332–340 (2006)
14. Françon, J.: Discrete combinatorial surfaces. CVGIP, Graph. Models Image Process. **57**(1), 20–26 (1995)
15. Gonthier, G.: A computer-checked proof of the four colour theorem, 57 pp. Microsoft Research, Cambridge. http://research.microsoft.com/~gonthier (2005)
16. Gonthier, G., Mahboubi, A.: A small scale reflection extension for the Coq system, 75 pp. RR 6455, Microsoft Research-INRIA. http://hal.inria.fr/inria-00258384/ (2007)
17. Gonthier, G.: Formal proof - the four-Colour theorem. Not. Am. Math. Soc. **55**(11), 1382–1393 (2008)
18. Griffiths, H.: Surfaces. Cambridge University Press, Cambridge (1981)
19. Hales, T.: Formalizing the proof of the Kepler conjecture. In: Theorem Proving in HOL Conf. LNCS, vol. 3223, p. 117. Springer, New York (2004)
20. Hales, T.: A verified proof of the Jordan Curve Theorem. Seminar talk. Dep. of Math., University of Toronto http://www.math.pitt.edu/~thales (2005)
21. Hales, T.: The Jordan Curve Theorem, formally and informally. Am. Math. Mon. **114**(10), 882–893 (2007)
22. Huet, G., Kahn, G., Paulin-Mohring, C.: The Coq proof assistant—a tutorial—version 8.0. Tech. report, INRIA, France http://coq.inria.fr/doc-fra.html (2004)
23. Khachan, M., Chenin, P., Deddi, H.: Digital pseudomanifolds, digital weakmanifolds and Jordan-Brower separation theorem. Discrete Appl. Math. **125**(1), 45–57 (2003)
24. Kornilowicz, A.: Jordan Curve Theorem. Formaliz. Math. **13**(4), 481–491 (2005) (Univ. of Bialystock)
25. Kong, T.Y., Rosenfeld, A.: Digital topology: introduction and survey. Comput. Vis. Image Process. **48**, 357–393 (1989)
26. Malgouyres, R.: A definition of surfaces of Z^3—a new 3D discrete Jordan theorem. Theor. Comp. Sci. **186**, 1–41 (1997)
27. Paulin-Mohring, C.: Inductive definition in the system Coq—rules and properties. In: Typed Lambda-Calculi and Applications. LNCS, vol. 664, pp. 328–345. Springer, New York (1993)
28. Simpson, C.: Explaining Gabriel-Zisman localization to the computer. J. Autom. Reason. **36** (2006). doi:10.1007/s10817-006-9038-x,259–285
29. Slapal, J.: A digital analog of the Jordan Curve Theorem. Discrete Appl. Math. **139**(1–3), 231–251 (2004)

30. Stahl, S.: A combinatorial analog of the Jordan Curve Theorem. J. Comb. Theory **B 26**, 28–38 (1983)
31. Tutte, W.T.: Combinatorial oriented maps. Can. J. Math. **XXXI**(5), 986–1004 (1979)
32. Tutte, W.T.: Graph theory. In: Encyclopedia of Mathematics and its Applications. Addison Wesley, Reading (1984)
33. Veblen, O.: Theory on plane curves in non-metrical analysis situs. Trans. Am. Math. Soc. **6**, 83–98 (1905)
34. Vince, A., Little, C.H.C.: Discrete Jordan Curve Theorems. J. Comb. Theory **B 47**, 251–261 (1989)
35. Yamamoto, M., Nishizaki, S., Hagiya, M., Tamai, T.: Formalization of planar graphs. In: Theorem Proving in HOL Conf. LNCS, vol. 971, pp. 369–384. Springer, New York (1995)