University of British Columbia – Technical Report

Point-Curve-Surface Complex: A Cell Decomposition for Non-Manifold Two-Dimensional Topological Spaces

Boris Dalstein[1], Rémi Ronfard[2] and Michiel van de Panne[1]

[1]University of British Columbia, Canada
[2]Laboratoire Jean Kuntzmann, University of Grenoble & Inria, France

July 25, 2014

Abstract

We introduce the point-curve-surface complex (PCS complex), a cell complex tailored for a canonical decomposition of "regular" non-manifold two-dimensional topological spaces (i.e., the class of topological spaces that admits a simplicial 2-complex decomposition). PCS complexes are obtained by gluing together points, curves and surfaces, very similarly to CW complexes, except that the cells are not restricted to be topological disks. Also, gluing constraints that cells must satisfy to form a valid complex are much stricter than with CW complexes. We conjecture that this specific combination of flexibility and constraints ensures uniqueness of a minimal decomposition, obtained via atomic simplifications performed in any order.

The PCS complex admits a combinatorial description, called the abstract PCS complex. It is made of vertices, closed edges, open edges defined by a start and an end vertex, and faces defined by an orientability, a genus, and a boundary defined as a sequence of cycles, where a cycle is either a vertex, an oriented closed edge possibly repeated, or a looping sequence of consecutive oriented open edges. It provides a compact language to describe non-manifold two-dimensional topological spaces up to homeomorphism. We detail the create/delete, glue/unglue and cut/uncut topological operators on this combinatorial structure.

A vector graphics complex (VGC) is an abstract PCS complex without the information of orientability and genus, since it is irrelevant for vector graphics. Therefore, these two concepts are fundamentally equivalent, and topological operators on abstract PCS complexes can also be applied to VGCs. As such, this report provides insight and theoretical relevance to most of the design decisions behind the VGC, as well as a thorough description of the topological operators on the VGC, described in terms of abstract PCS complexes.

1 Introduction

Let us start by a few words on the nature of this technical report. It is an early dissemination of research in progress, to provide a more in-depth and theoretical perspective on the concepts presented in the article "Vector Graphics Complexes" (VGC) [Dalstein et al., 2014]. It is only provided in the hope that it is useful, but we do not consider it a complete research work publishable as is. A quite significant amount of time has been spent to make it as complete, rigorous and correct as possible, but in order to dissemate this work in time alongside [Dalstein et al., 2014], there are still a few missing arguments, proofs and discussions here and there. Also, since it has not yet been thoroughly reviewed, it may contain a few mistakes. If you find any, we do apologize and would be grateful if you could report them so we could fix them in later work. When we use the wording "it can be shown that [...]", this means that we have a formal proof on paper that we did not have time to typeset in the report, or at the very least that we have very convincing arguments and a sketch of the proof, such that we would be highly surprised if it turns out to be a wrong statement. In all other cases that resort more to intuition than an actual proof, we use the wording "we believe that [...]".

We present in this report the notion of *PCS complex*. It is a novel decomposition of two-dimensional topological spaces into a new kind of *cells*, using a formalism deeply rooted in algebraic topology. After reading a few paragraphs introducing the concept, one may rightfully ask: *"Wait, how is this complicated algebraic topology related to the VGC?"*. Here is the background story: the VGC is defined as a *combinatorial* structure, on top of which 2D geometry is added to render it. Then, topological operators are defined to alter this combinatorial structure, such as glue/unglue and cut/uncut. The key question is: how to *define* these topological operators? Essentially, a topological operator acting on the VGC is a function that takes as input a valid VGC, and outputs a valid VGC. Hopefully, the output is something that corresponds to what we "expect" from the operator. However, what we "expect" does not formally define the requirements of the operator. In other words, since the structure is purely combinatorial, what the structure *represents* is a matter of interpretation, and then the definition of the operator is actually the algorithm itself, and one has to take design decisions as to what the output should be depending on the input. This departs fundamentally from a variety of other topological data-structures (including planar maps and the selective geometric complex), that have a "pointset" formalism that comes first. Hence, the topological operators can be defined in terms of pointsets, and the actual algorithm has the requirement to correspond to the pointset definition. With a combinatorial structure, things are fundamentally different. The only *formal* requirement of any operator is to output a valid VGC, to which we add the *informal* requirement to "make sense". When designing the algorithm for the cut topological operator, we came across a few cases where several valid VGCs all seemed to be reasonable candidates that could "make sense" as output of the algorithm. However, a choice had to be made and hence the question "which of these outputs makes *more* sense?" was often raised. The PCS complex is a side product of trying to answer this question. It is the structure toward which we converged after several attempts to define a "geometric realization" for the VGC, i.e. an interpretation of the VGC as a non self-intersecting pointset. The insight provided by this research has helped refine the VGC definition and algorithms to what they are now.

The report is organized as follows. In Section 2, we recall the notions of algebraic topology that are required for the understanding of the report, and we give pointers to relevant textbooks for the interested reader. In Section 3, we informally motivate the usefulness of PCS complexes, independently to their link with the VGC. In Section 4, we formally define the notion of PCS complex, an "actual" topological structure, i.e. defining a topology in a pointset sense. In Section 5, we independently define the notion of *abstract PCS complex*, a combinatorial structure which generalizes the VGC by adding information about orientability and genus of faces. This additional information is irrelevant for 2D vector graphics rendered using winding numbers, but is necessary to make the connection between the combinatorial structure (abstract PCS complex) and the actual pointset topology (PCS complex). More specifically, for each abstract PCS complex, we define a PCS complex called its *geometric realization*, and conversely we see that every PCS complex is homeomorphic to the geometric realization of some abstract PCS complex. In Section 6, we introduce a few basic algebraic operations on halfedges, paths and cycles, a preliminary for Section 7, in which we present in depth the topological operators create/delete, glue/unglue, and cut/uncut acting on abstract PCS complexes, with both theoretical discussion and pseudocode. At last, in Section 8, we bridge the gap between abstract PCS complexes and VGCs, by explaining how exactly they relate to each other, and why this PCS digression was useful to better understand and define the VGC and its topological operators. We conclude the report with Section 9, in which we introduce the notion of simplification and equivalence of PCS complexes, related to the cut/uncut operators. It lays the ground for future work, by formalizing the conjecture of uniqueness of a minimal PCS decomposition.

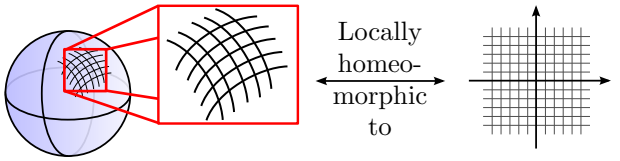


Figure 1: The sphere $\mathbb{S}^2 = \{x \in \mathbb{R}^3, ||x|| = 1\}$ is a 2-manifold without boundary, since it is everywhere locally homeomorphic to $\mathbb{R}^2$.

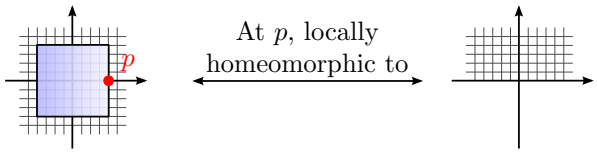


Figure 2: The surface $[-1,1] \times [-1,1]$ is a 2-manifold with boundary, since it is locally homeomorphic either to $\mathbb{R}^2$ or to $\mathbb{R} \times [0, +\infty)$.

2 Prerequisites of algebraic topology

This report makes use of many concepts from algebraic topology, such as topological spaces, homeomorphisms, manifolds with boundary, compact manifolds, homeomorphisms, the classification theorems for curves and surfaces, polygonal presentations, simplicial complexes, CW complexes, quotient spaces, as well as the differences between immersions and embeddings. In this section, we recall semi-informally all these concepts so that the unfamiliar reader can still follow the report. A formal but well illustrated introduction to these concepts can be found in [Lee, 2011].

2.1 Topological spaces and homeomorphisms

In this report, whenever we say ***topological space***, we mean a Hausdorff topological space. We do not recall here the formal definition since it is not essential for the understanding of this report, but the interested reader can find it in [Lee, 2011, Chapter 2], or more simply by looking it up on Wikipedia. Intuitively, a Hausdorff topological space is a set — for instance $X = \{x \in \mathbb{R} \mid 0 < x < 5\}$ — together with a definition of "open subsets" satisfying reasonable properties. For instance, $I = (1,2) = \{x \in \mathbb{R} \mid 1 < x < 2\}$ is an open subset of X, while $J = [1,2) = \{x \in \mathbb{R} \mid 1 \leq x < 2\}$ is not an open subset of X. In practice, most sets that you would think of are indeed Hausdorff topological spaces. For instance, $\mathbb{R}^n$ is a Hausdorff topological space, and any subset of $\mathbb{R}^n$ is a Hausdorff topological space as well.

Two topological spaces X and Y are said to be ***homeomorphic***, denoted $X \cong Y$, if and only if there exists an ***homeomorphism*** between X and Y, i.e. an invertible continuous function $\phi : X \to Y$ whose inverse ϕ^{-1} is also continuous. $X \cong Y$ captures the intuitive concept of "X and Y are essentially the same topological space", meaning that even though they are not necessarily "equal", they behave similarly and have a similar "shape". For instance, the two closed intervals $I_1 = [0,1]$ and $I_2 = [1,2]$ are not equal but they are homeomorphic. The two open intervals $J_1 = (0,1)$ and $J_2 = (1,2)$ are not equal but they are homeomorphic. Also, the two circles $S_1 = \{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 = 1\}$ and $S_1 = \{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 = 2\}$ are not equal but they are homeomorphic. However, none of I_1, J_1 or S_1 are homeomorphic to each other, which is the formal way of saying: "they do not look alike". This informal statement is so intuitive that we have different names for these objects (closed intervals, open intervals, circles), while there is no terminology to differentiate, say, I_1 and I_2. Identifying that a topological space X is homeomorphic to a known topological space Y is of primary importance because it makes possible to infer properties of X from the known properties of Y.

2.2 Manifolds with boundary and compact manifolds

An ***n-manifold without boundary*** $\mathbb{M}$ is a topological space that is everywhere locally homeomorphic to $\mathbb{R}^n$. More formally: if X is a topological space and $p \in X$, then an open subset of X containing p is called a ***neighbourhood*** of p and denoted N_p. $\mathbb{M}$ is an n-manifold without boundary if and only if for each $p \in \mathbb{M}$, there exists a neighbourhood N_p homeomorphic to $\mathbb{R}^n$. For instance, the sphere $\mathbb{S}^2$ is a 2-manifold without boundary because for each point p on the sphere, it "looks locally like" the plane, as illustrated in Figure 1. However, the square $[-1,1] \times [-1,1] \subset \mathbb{R}^2$ is *not* a 2-manifold without boundary because at $p = (1,0)$, it is locally homeomorphic to $\mathbb{R} \times [0, +\infty)$, as illustrated in Figure 2. Since we want this last example to be included in our definition of manifold, we use a more general definition: an ***n-manifold with boundary*** is defined as a topological space that is everywhere locally homeomorphic either to $\mathbb{R}^n$ or to $\mathbb{H}^n = \mathbb{R}^{n-1} \times [0, +\infty)$. In this

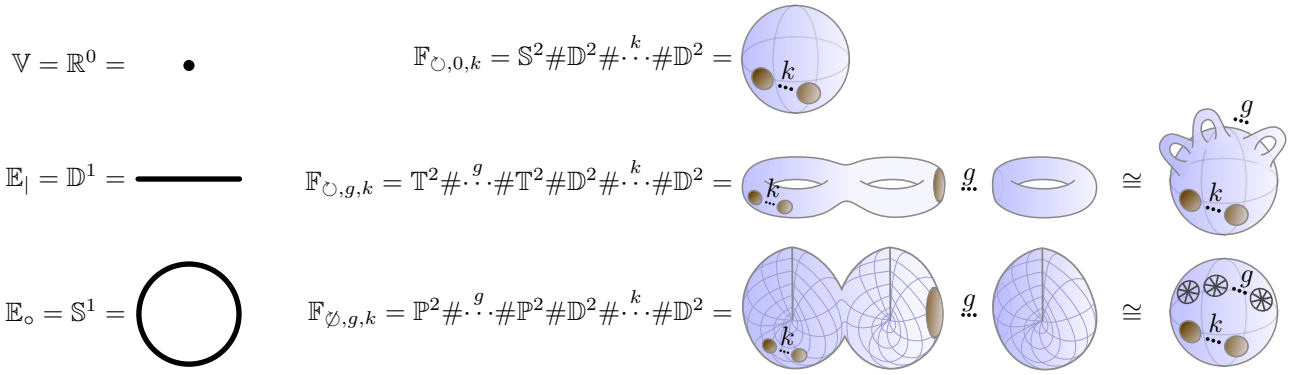


Figure 3: The classification of points, curves and surfaces. Any connected compact n-manifold with $n \leq 2$ is homeomorphic to one and only one of these known compact manifolds. The notations $\mathbb{V}$, $\mathbb{E}_|$, $\mathbb{E}_\circ$, $\mathbb{F}_{\circlearrowleft,g,k}$, and $\mathbb{F}_{\emptyset,g,k}$ are non-standard and introduced for conciseness and clarity. They are the characteristic manifolds for, respectively: vertices, open edges, closed edges, orientable faces, and non-orientable faces.

report, whenever we say ***manifold***, we mean manifold with boundary, unless *without boundary* is explicitly stated. If $\mathbb{M}$ is an n-manifold, then the ***interior*** of $\mathbb{M}$, denoted $\text{int}(\mathbb{M})$, are the points of $\mathbb{M}$ that have a neighbourhood homeomorphic to $\mathbb{R}^n$. Conversely, the ***boundary*** of $\mathbb{M}$, denoted $\partial\mathbb{M}$, are the points of $\mathbb{M}$ that have a neighbourhood homeomorphic to $\mathbb{H}^n$, i.e. $\partial\mathbb{M} = \mathbb{M} \setminus \text{int}(\mathbb{M})$.

A ***compact manifold*** $\mathbb{M}$ is a manifold that is compact as a topological space. A formal definition can be found in [Lee, 2011, Chapter 4] but is not necessary for the understanding of this report. If $\mathbb{M}$ is a subset of $\mathbb{R}^n$, then $\mathbb{M}$ is compact if and only if it is bounded and topologically closed in $\mathbb{R}^n$ (i.e., $\mathbb{R}^n \setminus \mathbb{M}$ is open). For instance, the closed interval $[0, 1]$ is a compact manifold, but the real line $\mathbb{R}$ is not a compact manifold because it is not bounded, and the open interval $(0, 1)$ is not a compact manifold because it is not closed in $\mathbb{R}$. If $\mathbb{M}$ is a compact n-manifold, then $\partial\mathbb{M}$ is a compact $(n-1)$-manifold but $\text{int}(\mathbb{M})$ is an n-manifold generally *not* compact. More specifically, $\text{int}(\mathbb{M})$ is compact iff $\partial\mathbb{M} = \varnothing$ (i.e., iff $\mathbb{M}$ is a compact n-manifold without boundary).

2.3 Classification of compact n-manifolds for $n \leq 2$

Compact manifolds of dimension two and lower, i.e. points, curves, and surfaces, have been completely "classified". This means that we know a very concise list of compact manifolds, illustrated in Figure 3, such as any point, curve or surface is necessarily homeomorphic to one and only one of the manifolds in the list. In other words, any surface "looks like" one and only one of the surfaces in the list. In this section, we recall this classification. Following common practice, we only consider here *connected* compact manifolds, but it is trivial to generalize to all compact manifolds since any compact n-manifold can be decomposed as a finite disjoint union of connected compact n-manifolds.

Dimension 0 A connected compact 0-manifold is homeomorphic to the simple point $\mathbb{R}^0 = \{0\}$.

Dimension 1 A connected compact 1-manifold is homeomorphic either to the unit circle $\mathbb{S}^1$ or to the closed interval $\mathbb{D}^1 = [-1, 1]$. A proof can be found in [Gale, 1987] or in [Lee, 2011, Ch. 5, p. 143-147].

Dimension 2 A connected compact 2-manifold **without boundary** is homeomorphic to either:

- The sphere $\mathbb{S}^2$, called the ***surface of genus 0***.
- The connected sum of $g \geq 1$ tori $(\mathbb{T}^2)^g = \mathbb{T}^2\#\overset{g}{.}\#\mathbb{T}^2$, called the ***orientable surface of genus g***.
- The connected sum of $g \geq 1$ projective planes $(\mathbb{P}^2)^g = \mathbb{P}^2\#\overset{g}{.}\#\mathbb{P}^2$, called the ***nonorientable surface of genus g***.

We clarify here the terminology. The ***torus*** $\mathbb{T}^2$ is the topological space obtained by "gluing" together (or "sewing") the opposite boundaries of a cylinder as depicted in Figure 4. Note that orientation matters: if you choose to glue using the reverse orientation of one of the boundaries, as depicted in Figure 5, you get the ***Klein***

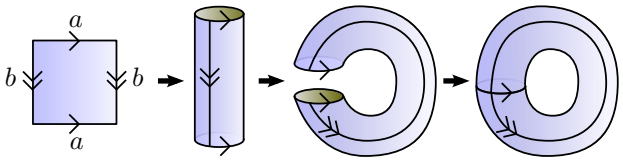


Figure 4: A polygonal presentation of the torus is the single word $W = aba^{-1}b^{-1}$.

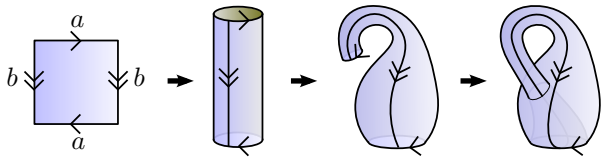


Figure 5: A polygonal presentation of the Klein bottle is the single word $W = abab^{-1}$.

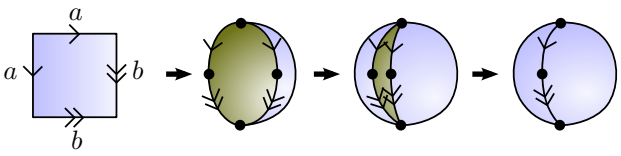


Figure 6: A polygonal presentation of the sphere is the single word $W = abb^{-1}a^{-1}$.

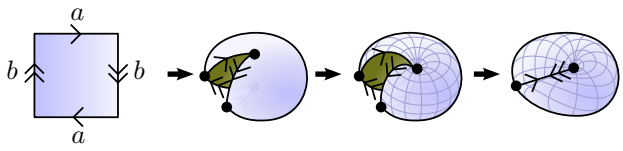


Figure 7: A polygonal presentation of the projective plane is the single word $W = abab$.

bottle $\mathbb{K}^2$ instead, which is not homeomorphic to the torus but to $\mathbb{P}^2\#\mathbb{P}^2$. The ***projective plane*** $\mathbb{P}^2$ is the topological space obtained by "gluing" the unique boundary of a disk to the unique boundary of a Möbius strip. Alternatively, as illustrated in Figure 7, it can be obtained by gluing together one half of the boundary of a disk to the other half, using the appropriate orientation. The ***connected sum*** of two surfaces consists in removing one disk from each surface, and gluing together the two obtained boundaries.

The classification of surfaces given above has been first proven in [Dehn and Heegaard, 1907], and is nicely covered and illustrated in [Lee, 2011, Ch. 6]. We recall below the high-level steps of this proof, which involves the concept of *polygonal presentations*, related to the concept of abstract PCS complex introduced in this report.

- A ***word*** W is defined, given a set S, as a finite sequence of $k \geq 1$ symbols, each of the form a or a^{-1} with $a \in S$. It represents a regular polygon with k edges, where some edges are identified in pairs with a chosen orientation. For instance, the word $W = abab^{-1}$ represents a square, where the first and third edges are identified with the same orientation, and the second and fourth edges are identified with opposite orientations (cf. Figure 5, left). A ***polygonal presentation*** $\mathcal{P}$ is defined as a set of words.
- The ***geometric realization*** of $\mathcal{P}$, denoted $|\mathcal{P}|$, is the topological space obtained by gluing together the paired edges of the polygons described by its words. For instance, the geometric realization of $\mathcal{P} = \{abab\}$ is a torus (cf. Figure 4). Other examples are given in Figure 5, 6, and 7. Two polygonal presentations are said to be ***topologically equivalent*** if and only if their geometric realizations are homeomorphic.
- A ***surface presentation*** is defined as a polygonal presentation $\mathcal{P}$ where each element $a \in S$ occurs exactly twice. In this case, we can prove than $|\mathcal{P}|$ is a compact 2-manifold without boundary. Conversely, it can be shown that any compact 2-manifold without boundary admits a surface presentation $\mathcal{P}$.
- Finally, combinatorial operations on surface presentations prove that any surface presentation is topologically equivalent to either:
  - the canonical surface presentation of the sphere:
$$\mathcal{P} = \{aa^{-1}\} \tag{1}$$
  - the canonical surface presentation of the connected sum of $g \geq 1$ tori:
$$\mathcal{P} = \{a_1b_1a_1^{-1}b_1^{-1} \dots a_gb_ga_g^{-1}b_g^{-1}\} \tag{2}$$
  - the canonical surface presentation of the connected sum of $g \geq 1$ projective planes:
$$\mathcal{P} = \{a_1a_1 \dots a_ga_g\} \tag{3}$$

The classification above was for compact 0-manifolds with boundary, compact 1-manifolds with boundary, but only for compact 2-manifolds *without boundary*. The only missing piece is the classification of compact 2-manifolds *with boundary*. It turns out that these are simply compact 2-manifolds without boundary from which

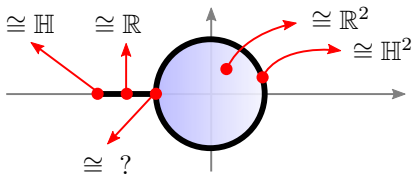


Figure 8: $X = \{(x, y) \mid x^2 + y^2 \leq 1\} \cup \{(x, 0) \mid x \in [-2, -1]\}$ is a non-manifold space: there exists no n such that X is everywhere locally homeomorphic to either $\mathbb{R}^n$ or $\mathbb{H}^n$.

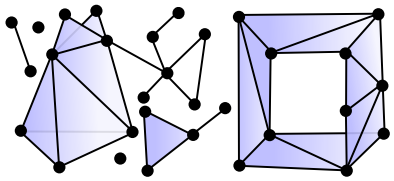


Figure 9: A two-dimensional simplicial complex. The union of the simplices is in general a non-manifold space. Though, some connected components may be manifolds.

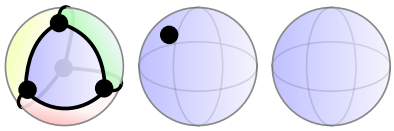


Figure 10: Left: minimal simplicial decomposition of $\mathbb{S}^2$ (14 simplices). Middle: minimal CW decomposition (2 CW-cells). Right: minimal PCS decomposition (1 PCS-cell).

we remove the interior of $k \geq 0$ disjoint closed disks, which can be formally achieved by taking the connected sum with $k \geq 0$ disks. We illustrate this classification in Figure 3. Finally, we recall an interesting theorem that is useful to analyse the cut and uncut topological operators. The proof of this theorem is an intermediate step in the proof of the classification of surfaces given above.

Theorem 1. *The connected sum of a projective plane and a torus is homeomorphic to the connected sum of three projective planes, i.e.:*

$$\mathbb{P}^2 \# \mathbb{T}^2 \cong \mathbb{P}^2 \# \mathbb{P}^2 \# \mathbb{P}^2 \tag{4}$$

2.4 Non-manifold topological spaces

Compact manifolds are very convenient to study, but unfortunately not all compact topological spaces are compact manifolds, as for instance the one illustrated in Figure 8. To include this example and many others (but not *all* compact topological spaces, which would be too general), a wider class of topological spaces has been defined: those that can be obtained by "gluing together" simple manifold pieces. For instance, the topological space in Figure 8 can be obtained by gluing a segment with a disk. Topological spaces defined within this general framework are commonly refered to as *complexes*, such as simplicial complexes and CW complexes, which we recall in this section. As noted in [Edelsbrunner and Harer, 2010, Ch. III, p. 51], one of the most important characteristic that makes each kind of complex different from one another is how "simple" the glued pieces are. The simpler the pieces, the more pieces you need to decompose a given space (cf. Figure 10). Therefore, choosing the right formalism to tackle a given topological problem is a trade-off between the complexity of each piece, and the number of pieces you need to decompose a given space. For instance, the pieces of a simplicial complex are called *simplices* and are n-dimensional triangles, while the pieces of a CW complex are called *cells* and are homeomorphic to an n-dimensional open disk. Thus, a simplex is a special case of a cell, meaning that the pieces of CW complexes are "more complex" than the pieces of simplicial complexes. As a consequence, the sphere can be decomposed with only two cells, while it requires 14 simplices. In this report, we introduce the PCS complex, whose pieces, also called cells for lack of a better name, are only required to be homeomorphic to the interior of a compact manifold. Thus, a "CW-cell" is a special case of a "PCS-cell", meaning that the pieces of PCS complexes are "even more complex" than the pieces of CW complexes, and can for instance decompose the sphere as a single PCS-cell.

Abstract simplicial complexes An abstract simplicial complex [Edelsbrunner and Harer, 2010, p. 53] is a finite collection of sets A such that:

$$(\alpha \in A \quad \text{and} \quad \beta \subseteq \alpha) \quad \Rightarrow \quad \beta \in A \tag{5}$$

The elements α in A are called ***simplices***, and each simplex is given as the set of its ***vertices***. The ***dimension*** of a simplex is $\dim \alpha = \operatorname{card} \alpha - 1$, and the dimension of the complex is the maximum dimension of any of its simplices. Intuitively, a two-dimensional abstract simplicial complex is a triangle mesh made of vertices, edges and triangles (possibly non-manifold, with dangling edges or isolated vertices), as illustrated in Figure 9. The formal definition ensures that if a triangle defined by the vertices $\{0, 1, 2\}$ is part of the complex, then all the vertices $\{0\}$,$\{1\}$, and $\{2\}$, and all the edges $\{0, 1\}$,$\{0, 2\}$, and $\{1, 2\}$ are also part of the complex, and they are called the ***boundary simplices*** of the triangle.

CW complexes Let us first formally define this concept, then right after provide the intuition behind the formalism. First, we define the ***n-disk***, its interior the ***open n-disk***, and its boundary the ***(n-1)-sphere*** as:

$$\mathbb{D}^n = \{x \in \mathbb{R}^n \mid \|x\| \leq 1\}, \tag{6}$$

$$\mathring{\mathbb{D}}^n = \text{int}(\mathbb{D}^n) = \{x \in \mathbb{R}^n \mid \|x\| < 1\}, \tag{7}$$

$$\mathbb{S}^{n-1} = \partial\mathbb{D}^n = \{x \in \mathbb{R}^n \mid \|x\| = 1\}. \tag{8}$$

An ***n-cell*** c is defined as a topological space homeomorphic to $\text{int}(\mathbb{D}^n)$. A ***cell decomposition*** $\mathcal{C}$ of a topological space X is a collection of disjoints cells c_i such that $X = \bigcup_i c_i$. The ***n-skeleton*** X^n of X is the union of k-cells of $\mathcal{C}$ such that $k \leq n$. Finally, $\mathcal{K} = (X, \mathcal{C})$ is called a ***CW complex*** if it satisfies:

- **Axiom 1**: ('Characteristic maps')

  For each n-cell $c \in \mathcal{C}$, there exists a continuous function $\Phi_c : \mathbb{D}^n \to X$ such that the restriction of Φ_c to $\text{int}(\mathbb{D}^n)$ is an homeomorphism from $\text{int}(\mathbb{D}^n)$ to c, and such that $\Phi_c(\partial\mathbb{D}^n) \subseteq X^{n-1}$.

- **Axiom 2:** ('**C**losure finiteness')

  The closure $\overline{c}$ intersects only a finite number of other cells.

- **Axiom 3:** ('**W**eak topology')

  $A \subseteq X$ is closed iff $A \cap \overline{c}$ is closed for each $c \in \mathcal{C}$.

Despite the fact that the last two axioms are those responsible for the acronym "CW", you can safely ignore them in this report, since they are automatically true if the number of cells is finite, which should always be the case for computer graphics applications. Therefore, let us simply clarify this obscure CW complex definition by focusing on the preliminary definitions and the first axiom. Since an n-cell is a pointset homeomorphic to $\text{int}(\mathbb{D}^n)$, this means that a 0-cell (called ***vertex***) is a single point in space, a 1-cell (called ***edge***) is a pointset homeomorphic to the open interval $(-1, 1)$, and a 2-cell (called ***face***) is a pointset homeomorphic to the open 2-disk $\mathring{\mathbb{D}}^2$. A two-dimensional cell decomposition of a topological space X is therefore a partition of X into vertices $v \cong \{0\}$, edges $e \cong (-1, 1)$ and faces $f \cong \mathring{\mathbb{D}}^2$.

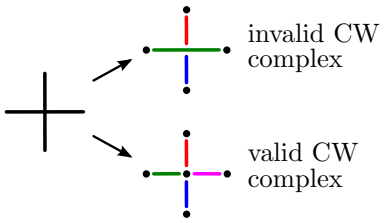


Figure 11: Two valid *cell decompositions* of a cross, but only one is a valid *CW complex*.

However, this allows spaces like $X = \mathbb{R}$ to be decomposed as a single cell $e = \mathbb{R}$, since $\mathbb{R} \cong (-1, 1)$. Also, this allows a cross to be decomposed as four vertices and three edges (cf. Figure 11, top). Because we do not want these decompositions to be valid CW complexes, Axiom 1 adds some restrictions. In the case of edges, instead of "simply" requiring $e \cong (-1, 1)$, we require the existence of a continous function $\Phi_e : [-1, 1] \to X$ such that $\Phi_e((-1, 1)) = e$. This way, even though edges are "open intervals", they are forced to "look like interior of closed intervals", thus the edge $e = \mathbb{R}$ is not allowed as part of a CW complex. Finally, Axiom 1 also requires that $\Phi_e(-1)$ and $\Phi_e(1)$ be included in the 0-skeleton of $\mathcal{C}$. In other words, it requires that the "edge boundary" $\partial e = \overline{e} \setminus e$ is made of vertices that are part of the decomposition. This additional requirement enforces the existence of a vertex at the intersection of the cross (cf. Figure 11). We note that Φ_e restricted to $(-1, 1)$ must be an homeomorphism (in particular, must be invertible), but it is not required that Φ_e be invertible on the whole domain of definition $[-1, 1]$. This prevents self-intersections in the interior of the edge, but allows $\Phi_e(-1)$ to be equal to $\Phi_e(1)$. In other words, it is allowed that the start vertex of the edge is equal to the end vertex. All these considerations scale for faces: the closure of a face must be compact, the boundary of a face must be included in a union of vertices and edges, and faces cannot self-intersect in their interior, but their boundary can "use" the same vertex or edge several times. For instance, the boundary of a face can be a single vertex (cf. Figure 10, middle), or even a single point in the interior of an edge (cf. Figure 18), or can do "switch-backs" in the interior of an edge (cf. Figure 19).

2.5 Geometric realizations and quotient spaces

The reader may have noticed that the definition of CW complexes that we have given differs greatly from the definitions of polygonal presentations and abstract simplicial complexes because it relies on the existence

of a topological space X, that we decompose into cells. On the contrary, a polygonal presentation or an abstract simplicial complex is not a topological space *per se*, but a combinatorial description of one. From such combinatorial description, one can *build* the corresponding topological space, called its *geometric realization*, by "gluing" together known topological spaces, which is formally done using the concept of *quotient space* that we recall in the following paragraph.

Let X be a set, and let $\sim$ be an equivalence relation on X. For instance, let us take as a very simple example $X = \{1, 2, 3\}$ and $\sim$ defined such that $1 \sim 2$, $1 \not\sim 2$, and $2 \not\sim 3$. The ***equivalence classes*** of $\sim$ are defined as a partition of X into subsets regrouping elements that are equivalent to each others. In our example, there are two equivalence classes: $E = \{1, 2\}$ (since 1 and 2 are equivalent) and $F = \{3\}$ (since 3 is not equivalent to any other elements). The ***quotient set*** of X by $\sim$, denoted $X/\sim$, is defined as the set of equivalence classes of $\sim$. Therefore, in our example, we have $X/\sim = \{E, F\} = \{\{1, 2\}, \{3\}\}$. In other words, *quotienting* a set by an equivalence relation can be understood as transforming elements that were equivalent into a single element. The concept of ***quotient space*** is very similar, except that it acts on topological spaces instead of sets. This means that in addition to define $Y = X/\sim$ as the set of equivalence classes of $\sim$, it also makes Y a topological space by defining which subsets of Y are "open". More specifically, the open subsets of Y are defined as the sets of equivalent classes whose unions are open sets in X. This means that two equivalent classes E and F are "close-by" in Y if and only if there exist $x_E \in E$ and $x_F \in F$ that were originally "close-by" in X. For instance, consider $X = [0, 1] \cup [2, 3] \cup [4, 5]$, i.e. $X \subset \mathbb{R}$ is a disjoint union of three closed intervals (cf. Figure 12, top). Then let us consider the equivalence relation $\sim$, defined by $0 \sim 2$, $0 \sim 4$ and $2 \sim 4$, all other pairs of real number in X not being equivalent. This means that the set $P = \{0, 2, 4\}$ is one equivalence class, and every other element $x \in X$ is its own equivalence class $\{x\}$. By using the convenient notation $(\overline{a}, \overline{b}] = \{\{x\} \mid x \in (a, b]\}$, then we have $Y = (\overline{0}, \overline{1}] \cup (\overline{2}, \overline{3}] \cup (\overline{4}, \overline{5}] \cup \{P\}$. Because 0 was in the closure of $(0, 1]$ in X, and $0 \in P$, it can be shown that P is in the closure of $(\overline{0}, \overline{1}]$ in Y. Similarly, it can be shown that P is in the closure of $(\overline{2}, \overline{3}]$ and $(\overline{4}, \overline{5}]$. Therefore, the closures of $(\overline{0}, \overline{1}]$, $(\overline{2}, \overline{3}]$, and $(\overline{4}, \overline{5}]$ intersect in Y (at P), while the closures of $(0, 1]$, $(2, 3]$, and $(4, 5]$ did not intersect originally in X. This is why it is said that using this operation, the three closed intervals $[0, 1]$, $[2, 3]$, and $[4, 5]$ have been *glued*, by *identifying* the three real numbers 0, 2 and 4 as a single element. Quotienting X by $\sim$ has transformed a disjoint union of three closed intervals (a 1-manifold with boundary) into a star-like shape with three branches (a non-manifold space), as illustrated in Figure 12.

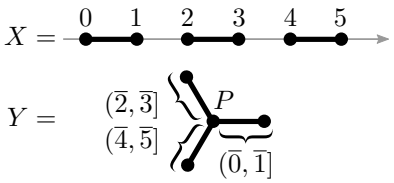


Figure 12: Ilustration of $X = [0, 1] \cup [2, 3] \cup [4, 5]$, and the quotient space $Y = X/\sim$ defined by $0 \sim 2 \sim 4$.

With this formalism, the ***geometric realization*** of an abstract simplicial complex, called a ***simplicial complex*** (i.e., not abstract), can be easily defined as a disjoint union of points, segments, triangles, and n-dimensional triangles that are glued together by identifying their common boundaries with a well-chosen equivalence relation. The reverse viewpoint can also be taken: given a possibly non-manifold topological space X (in a sense, "already glued"), and a decomposition of X into subsets homeomorphic to points, interior of segments, interior of triangles, and interior of n-dimensional satisfying a few properties on their boundaries, then it is called a simplicial complex, and its corresponding abstract simplicial complex can be defined.

Similarly, CW complexes can be defined either as we did (i.e., "Let X be a topological space. If there exists a cell-decomposition $\mathcal{C}$ such that there exists functions Φ_c satisfying [...], then $(X, \mathcal{C})$ is called a CW complex"), or by *building* them, via the explicit definition of characteric maps Φ_c gluing disjoint n-disks together. For instance, the space in Figure 8 is homeomorphic to a CW complex that can be built explicitly as follows. We first define X as the disjoint union of two points v_1 and v_2 (0-cells), two closed intervals e_1 and e_2 (1-cells, parameterized as $[-1, 1]$), and one disk f (2-cell, whose boundary is parameterized $[0, 2\pi)$). We then define $\Phi_{e_1}(-1) = v_1$, $\Phi_{e_1}(1) = v_2$, $\Phi_{e_2}(-1) = v_2$, $\Phi_{e_2}(1) = v_2$, $\Phi_f(\theta = 0) = v_2$, and $\Phi_f(\theta \in (0, 2\pi)) = \frac{\theta}{\pi} - 1 \in e_2$. From these characteretic maps, an equivalence relation $\sim$ can be defined, identifying each point in the boundary of each n-cell to a point of a k-cell, $k < n$, as illustrated in Figure 13. Quotienting X by this equivalence relation gives the final CW complex $Y = X/\sim$.

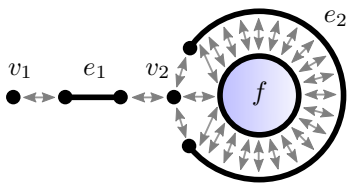


Figure 13: The space Figure 8 can be seen as a CW complex obtained by gluing together two points, two closed intervals and one disk. The double-arrows represent the equivalence relation for the glue operation.

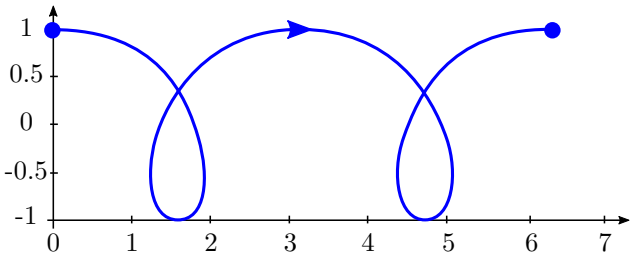


Figure 14: The continuous mapping $\Phi : [0, 4\pi] \to \mathbb{R}^2$ defined as $\Phi(t) = (\sin(t) + 0.5t, \cos(t))$ is an immersion. However, since the curve is self-intersecting, the mapping is not injective and hence Φ is not an embedding.

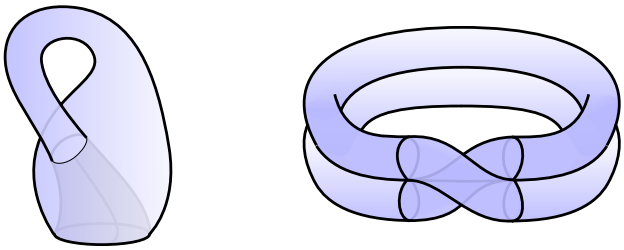


Figure 15: Two immersions of the Klein bottle in $\mathbb{R}^3$. Both immersions intersect themselves in a closed curve whose preimage consists of two loops. Image and caption inspired from [Edelsbrunner and Harer, 2010].

The geometric realization of a polygonal presentation, informally described in Section 2.3, is also formally defined via quotient spaces. Each word of size k defines a regular polygon with k edges, and points in the boundary of each polygon are identified to each other via the symbols in the words. This defines an equivalence relation, which subsequently defines the geometric realization as a quotient space.

Let us recall what we have learnt. We have a combinatorial structure called polygonal presentation, from which we can define a geometric realization using quotient spaces. We have a combinatorial structure called abstract simplicial complex, from which we can define a geometric realization using quotient spaces. Finally, we have a structure called CW complex, that can be defined using quotient spaces. However, CW do not (and cannot) have a "combinatorial description", because the characteristic maps still lie in the continuous world. Therefore, the notion of geometric realization does not apply for CW complexes. This makes them very inconvenient to implement on a computer. At the contrary, the notion of PCS complex that we introduce in this report does have a combinatorial description (called abstract PCS complex), even though it borrows most ideas from CW complexes.

2.6 Immersions vs. embeddings

A map Φ is defined as a continuous function between two topological spaces X and Y. Alternatively, we can call it an *immersion* of X into Y, and we say that X is *immersed* in Y. In addition, if Φ is injective then it is called an *embedding*, and we say that X is *embedded* in Y. Intuitively, an embedding is an immersion that does not produce self-intersections, as illustrated in Figure 14.

If the space Y is too low dimensional, there may not exist an embedding of X into Y. A classical example are non-orientable compact 2-manifolds without boundary, such as the Klein bottle, that can be embedded in $\mathbb{R}^4$ but not in $\mathbb{R}^3$. Hence, if $\mathcal{S}$ is an abstract simplicial complex representing a Klein bottle, and $X = |\mathcal{S}|$ is a geometric realization of $\mathcal{S}$, then every mapping from X to $Y = \mathbb{R}^3$ will produce self-intersections, as illustrated in Figure 15.

Combinatorial structures have the advantage to enable defining an immersion of their geometric realization (and hence making possible to visualize it and manipulate it in 2D or 3D), without actually constructing the geometric realization itself that would require additional dimensions. This is actually very standard in 3D polygon modeling: a Klein bottle can be easily modeled with any triangle mesh structure supporting non-orientable meshes. This will result in intersections of some triangles, but these intersections are not tracked and the intersecting triangles just ignore each others, which is the behaviour all polygon modeling artists expect. We use exactely the same principle with the VGC: instead of working with an embedding, as planar maps do, we work with an abstract combinatorial structure that is *immersed* in $\mathbb{R}^2$. The actual geometric realization of the VGC does not have to be explicitly constructed, very fortunately (cf. Section 8.3).

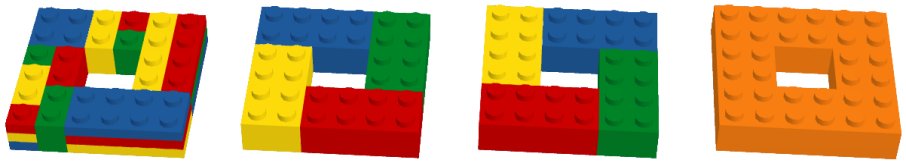


Figure 16: Left: Using Lego blocks, the above shape can be built in billions different ways. Middle: A decomposition involves at least four blocks, but there is no uniqueness of such a minimal decomposition. Right: If more building blocks were available, we can imagine that it could be decomposed with a single block, in which case we would have uniqueness of a minimal decomposition.

3 Motivations and overview

The main idea behind the PCS complex is to represent a two-dimensional non-manifold topological space as a decomposition into "as generic as possible" manifold pieces that are called vertices, edges and faces. This genericity comes from the goal to allow decompositions involving as few pieces as possible, for more flexibility for the artist, and to provide uniqueness of a minimal decomposition. For instance, we want to be able to represent a circle with a single closed edge, without the need to arbitrarily break it by inserting a vertex when no such vertex is relevant for the artist. Similarly, we want to represent a sphere as a single cell, as well as a torus, a Klein bottle, or any surface without boundary.

The choice of the available pieces to decompose topological spaces is a balance between simplicity and flexibility, depending on the application. The use of very simple pieces makes possible to have a lot of common properties for each of them, but on the other hand a lot of them might be necessary to decompose a given space. Conversely, very generic pieces make possible to have compact decompositions involving very few pieces, but then so little is known about each of them that it might not be useful to decompose the space in the first place (for instance, if pieces are so generic that every space can be decomposed into one piece, then it defeats the purpose of decomposition). Traditionally, decomposing a non-manifold topological space X into manifold pieces is done either via a simplicial complex, or a CW complex. These two approaches both have their benefits, and have been extensively popular notably because they scale in dimension. However, none of these complexes use pieces "generic enough" to ensure uniqueness of a minimal decomposition, a simple analogy using Lego blocks is illustrated in Figure 16. The Selective Geometric Complex (SGC) [Rossignac and O'Connor, 1989] provides uniqueness of a minimal decomposition, but relates to algebraic varieties while we are interested in topological manifolds. In addition, the CW complex does not admit a *presentation scheme*, i.e. a purely combinatorial structure whose geometric realization would be a CW complex, and hence are not suitable for computer implementation. The SGC admits a computer implementation, but it must include the definition of its *extents*, i.e. the coefficients of the polynomial equations defining the topological space. Hence, we cannot really say that it is a presentation scheme, but rather that the SGC itself, as an embedded topological space, is a combinatorial structure. In other words, a SGC cannot be defined separately from its geometric realization.

In our case, we are focused on *two-dimensional* topological spaces (possibly non-orientable or non-manifold), and the dimension two is very special in the sense that manifolds of dimension two and lower have all been classified. We use this at our advantage to propose a new notion of cell complex, similar to CW complexes, but specifically tailored to behave well in the two-dimensional case. Its cells are more flexible than CW cells, but we impose much stricter constraints on how cells can be glued together. These ad hoc constraints may seem rather arbitrary, but *in the specific case of the dimension two or less*, they make possible to define a representation scheme similar to the Vector Graphics Complex [Dalstein et al., 2014], and hopefully (not yet proven) ensure uniqueness of a minimal decomposition, while still representing the same class of topological spaces as 2D simplicial complexes and 2D regular CW complexes.

4 PCS complex

In this section, we introduce the concept of *PCS complex*, inspired by the concept of CW complex. It redefines the notion of cell to make them more generic (homeomorphic to the interior of any compact manifold), but is limited to two-dimensional spaces and introduce ad hoc gluing constraints (to make it combinatorially representable).

The section is organized as follows. First, in Section 4.1, we define the notion of *cell complex* for arbitrary dimension (though, designed specifically to work well in the dimension two or less, and likely useless for any higher dimension), then in Section 4.2, we prove a few properties true for arbitrary dimension. In Section 4.3, we compare our cell complexes to CW complexes. Finally, in Section 4.4, we define a PCS complex as a cell complex of dimension at most two, and exhaustively characterize all possible types of vertices, edges, and faces, and how they are allowed to be glued together. This characterization is much less compact than the original definition, but provides insight on what exactly does the definition allow, and is of primary importance for the link with abstract PCS complexes. The first three subsections are highly theoretical and with very few pictures, but the last one is more practical and with more examples. Therefore, we recommend the reader unfamiliar with CW complexes to directly jump to Section 4.4, and go back to the formal definitions later.

4.1 Cell complex

Throughout this report, a topological space means a Hausdorff topological space, and a n-manifold means a topological n-manifold with boundary.

Cell An ***n-cell*** c is a topological space homeomorphic to the interior of a connected compact n-manifold. The ***dimension*** of c is $\dim c = n$. A ***cell*** is an n-cell for some n. For $n = 0$, 1, and 2, we call them vertices, edges and faces.

Cell decomposition Let X be a topological space. A ***cell decomposition*** $\mathcal{C}$ of X is a finite collection of disjoints cells c_i such that $X = \bigcup_i c_i$. The ***n-skeleton*** X^n of X (resp. the ***n-skeletonset*** $\mathcal{C}^n$ of $\mathcal{C}$) is the union (resp. the set) of all the k-cells of $\mathcal{C}$ such that $k \leq n$. The smallest n such that $X^n = X$ is called the ***dimension*** of $(X, \mathcal{C})$.

Pointsets and set of pointsets We will often refer to objects that are either pointsets or sets of pointsets, and it is important not to confuse them. For instance, X is a pointset, a cell c is a subset of X and hence is also a pointset (a set of points $p \in X$). A *union* of cells is also a pointset. However, a *set* of cells, such as $\mathcal{C}$, is not a pointset but a set of pointsets. A subset $\mathcal{C}' \subset \mathcal{C}$ is also a set of pointset. If c_i are pointsets, and $\mathcal{C}_j$ are sets of pointsets, we introduce the notation $c' =< c_1, \ldots, c_k, \mathcal{C}_1, \ldots, \mathcal{C}_m >$ to conveniently define c' as the pointset obtained by the union of the pointsets c_i and of the pointsets in the sets $\mathcal{C}_j$. For instance, $< c_1, c_2 >= c_1 \cup c_2$, $< \mathcal{C} >= X$, and $< \mathcal{C}^n >= X^n$.

Closure and boundary The ***closure*** of a cell $c \in \mathcal{C}$, denoted $\bar{c}$, is the closure of c in X. The ***boundary*** of a cell $c \in \mathcal{C}$, denoted ∂c, is defined as the set difference $\bar{c} \setminus c$. An edge whose boundary is empty is called a ***closed edge***, otherwise it is called an ***open edge***.

Cell complex Let X be a topological space and $\mathcal{C}$ be a cell decomposition of X. The pair $\mathcal{K} = (X, \mathcal{C})$ is called a ***cell complex*** if and only if, for each n-cell $c \in \mathcal{C}$, there exist a connected compact n-manifold $\mathbb{M}_c$ and a map $\Phi_c : \mathbb{M}_c \to X$ satisfying the following ***cell complex constraints***:

- The restriction of Φ_c to $\text{int}(\mathbb{M}_c)$ is an homeomorphism from $\text{int}(\mathbb{M}_c)$ to c.
- For each connected component $\mathbb{B}_{c,i}$ of $\partial\mathbb{M}_c$, either:
  1. there exists a cell decomposition $\mathcal{D}_{c,i}$ of $\mathbb{B}_{c,i}$ such that for all $d_{c,i,j} \in \mathcal{D}_{c,i}$, the restriction of Φ_c to $d_{c,i,j}$ is an homeomorphism from $d_{c,i,j}$ to a cell $e_{c,i,j} \in \mathcal{C}$, or
  2. the image of $\mathbb{B}_{c,i}$ by Φ_c is a single vertex $v_{c,i} \in \mathcal{C}$, or
  3. the boundary component $\mathbb{B}_{c,i}$ is homeomorphic to $\mathbb{S}^1$, it is mapped by Φ_c to a single closed edge $e_{c,i} \in \mathcal{C}$, and the restriction of Φ_c to $\mathbb{B}_{c,i}$ "wraps $N_{f,i}$ times around $e_{c,i}$", for some $N_{f,i} \in \mathbb{N}^+$.

In other words, Φ_c must be an "homeomorphism by part" from cells decomposing $\mathbb{M}_c$ to cells of C (case 1.), with the first exception that a connected component of $\partial\mathbb{M}_c$ is allowed to shrink to a single vertex (case 2.), and the second exception that a connected component of $\partial\mathbb{M}_c$ homeomorphic to $\mathbb{S}^1$ is allowed to be mapped to a closed edge by wrapping around it several times (case 3.). We illustrate these cases in Figure 17, and formalize below what we mean by "wraps $N_{f,i}$ times around $e_{c,i}$".

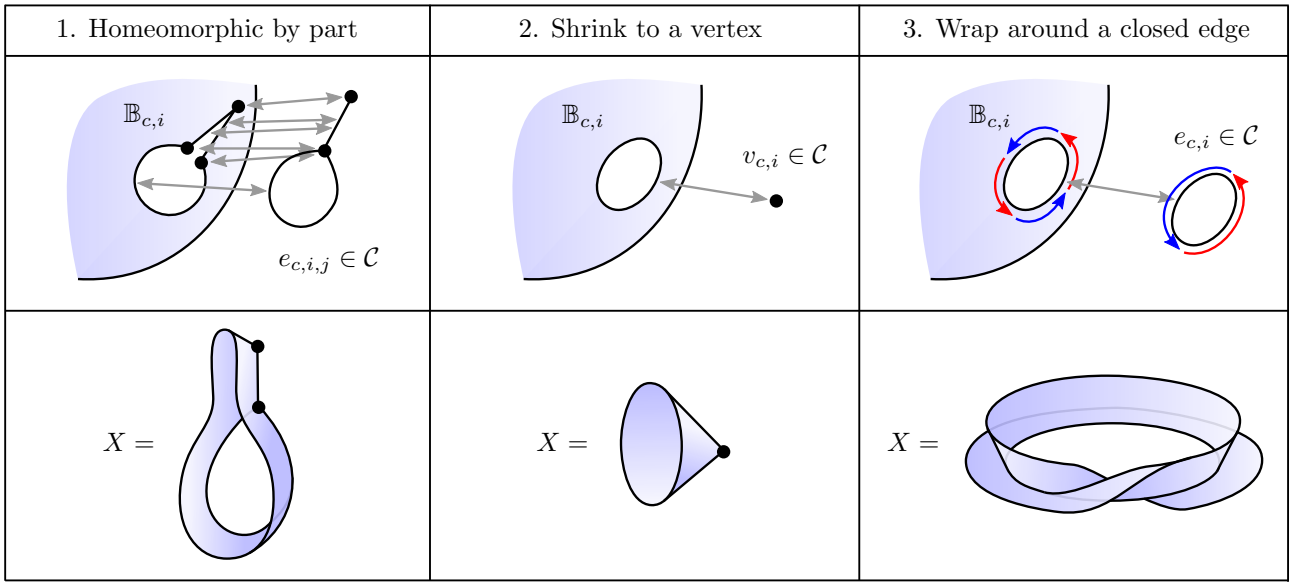


Figure 17: The three possible "gluing conditions" that each connected component $\mathbb{B}_{c,i}$ of the boundary of each cell c must satisfy. We illustrate them for $\dim c = 2$, since it is the dimension for which they have been designed. Top: How each connected component of the boundary of the characteristic manifold is glued to cells of lower dimension. Bottom: The actual, glued topological space X. In terms of abstract PCS complex, these three cases correspond to the three types of cycle. From left to right: non-simple cycle, Steiner cycle, and simple cycle.

Wrapping circles around circles Let A and B be two spaces homeomorphic to the circle $\mathbb{S}^1$. We say that a map $\Phi : A \to B$ ***wraps*** $N > 0$ ***times around*** B if and only if there exist two homeomorphisms $\Phi_A : A \to \mathbb{S}^1$ and $\Phi_B : B \to \mathbb{S}^1$ such that:

$$\Phi = \Phi_B^{-1} \circ W_N \circ \Phi_A \tag{9}$$

where, by using the usual parameterization $\theta \in [0, 2\pi)$ of $\mathbb{S}^1$, $W_N : \mathbb{S}^1 \to \mathbb{S}^1$ is the continuous map defined by:

$$W_N(\theta) = N\theta \tag{10}$$

Characteristic objects The connected compact n-manifold $\mathbb{M}_c$ is called the ***characteristic manifold*** of c, and the map Φ_c is called the ***characteristic map*** of c.

Cell neighbourhood We define the ***boundary cells*** of c as the set $\mathcal{B}_c$ of all $e_{c,i,j}$, $v_{c,i}$, and $e_{c,i}$. The ***star*** of a cell $c \in \mathcal{C}$ is defined as the set of cells

$$\mathcal{S}_c = \{c' \in \mathcal{C} \,|\, c \in \mathcal{B}_{c'}\}. \tag{11}$$

Dimension The ***dimension*** of a cell complex is defined as the dimension of its cell decomposition. A cell complex of dimension n is also called an ***n-complex*** for conciseness.

4.2 Relation between ∂c and $\mathcal{B}_c$, compactness, and subcomplexes

For the sake of completeness and comparison with CW complexes, we formally prove in this section a few immediate properties that cell complexes (in a PCS sense) satisfy, for arbitrary dimension. The reader not familiar with CW complexes may safely skip this section. Let $(X, \mathcal{C})$ be a cell complex. Then we have:

Lemma 1. $\forall c \in \mathcal{C},\ \ \mathcal{B}_c \subseteq \mathcal{C}^{n-1}$, *where* $n = \dim c$.

Proof. If $n = 0$, then $\mathbb{M}_c$ is a singleton and $\partial\mathbb{M}_c = \varnothing$ so there are no $\mathbb{B}_{c,i}$ hence no $v_{c,i}$, $e_{c,i}$, or $e_{c,i,j}$ and $\mathcal{B}_c = \varnothing$. Let $n \geq 1$. Since $\dim v_{c,i} = 0$, then $\dim v_{c,i} \leq n - 1$ and $v_{c,i} \in \mathcal{C}^{n-1}$. The case where $\mathcal{B}_c$ contains a cell of

type $e_{c,i}$ can only occur when $n \geq 2$, so we also have $e_{c,i} \in \mathcal{C}^{n-1}$. Since $d_{c,i,j}$ is a cell of $\mathbb{B}_{c,i}$ and that $\mathbb{B}_{c,i}$ is a compact $(n-1)$-manifold, we have $\dim d_{c,i,j} \leq n-1$. In addition, $\dim e_{c,i,j} = \dim d_{c,i,j}$ since Φ_c restricts to an homeomorphism from $d_{c,i,j}$ to $e_{c,i,j}$, hence $\dim e_{c,i,j} \leq n-1$ and $e_{c,i,j} \in \mathcal{C}^{n-1}$. □

Lemma 2. $\forall c \in \mathcal{C},\ \Phi_c(\partial \mathbb{M}_c) =< \mathcal{B}_c >$.

Proof. We have $\partial \mathbb{M}_c = \bigcup_i \mathbb{B}_{c,i}$, hence $\Phi_c(\partial \mathbb{M}_c) = \bigcup_i \Phi_c(\mathbb{B}_{c,i})$. The image of $\mathbb{B}_{c,i}$ is either a single vertex $v_{c,i}$ (case 2.), a closed edge $e_{c,i}$ (case 3.), or $\mathbb{B}_{c,i} = \bigcup_j d_{c,i,j}$ (case 1.) in which case $\Phi_c(\mathbb{B}_{c,i}) = \bigcup_j \Phi_c(d_{c,i,j}) = \bigcup_j e_{c,i,j}$. Hence, $\Phi_c(\partial \mathbb{M}_c) =< \ \ldots, v_{c,i}, \ldots, e_{c,i,j}, \ldots, e_{c,i}, \cdots >=< \mathcal{B}_c >$. □

Lemma 3. $\forall c \in \mathcal{C},\ \overline{c} = \Phi_c(\mathbb{M}_c)$.

Proof. If $\Phi : X \to Y$ is a map and $X' \subseteq X$, then $\Phi(\overline{X'}) \subseteq \overline{\Phi(X')}$. Thus in our case:

$$\Phi_c(\mathbb{M}_c) = \Phi_c(\overline{\text{int}(\mathbb{M}_c)}) \subseteq \overline{\Phi_c(\text{int}(\mathbb{M}_c))} = \overline{c}.$$

In addition, $\Phi_c(\mathbb{M}_c)$ is compact as a continuous image of a compact, thus $\Phi_c(\mathbb{M}_c)$ is closed in X since X is Hausdorff. Considering that $\overline{c}$ is defined as the intersection of all closed set in X containing c, that $\Phi_c(\mathbb{M}_c)$ contains c, and that $\Phi_c(\mathbb{M}_c)$ is closed in X, it proves that $\overline{c} \subseteq \Phi_c(\mathbb{M}_c)$. Hence, we proved that $\Phi_c(\mathbb{M}_c) \subseteq \overline{c} \subseteq \Phi_c(\mathbb{M}_c)$ thus $\overline{c} = \Phi_c(\mathbb{M}_c)$. □

Lemma 4. $\forall c \in \mathcal{C},\ \overline{c} =< c, \mathcal{B}_c >$.

Proof. $\overline{c} = \Phi_c(\mathbb{M}_c) = \Phi_c(\text{int}(\mathbb{M}_c) \cup \partial \mathbb{M}_c) = \Phi_c(\text{int}(\mathbb{M}_c)) \cup \Phi_c(\partial \mathbb{M}_c) = c \cup < \mathcal{B}_c >$. □

Proposition 1. $\forall c \in \mathcal{C},\ \partial c =< \mathcal{B}_c >$.

Proof. $\partial c = \overline{c} \backslash c =< c, \mathcal{B}_c > \backslash c =< \mathcal{B}_c >$ since $\mathcal{B}_c \subseteq \mathcal{C}^{n-1}$ and $\dim c = n$ thus $c \notin \mathcal{B}_c$. □

Proposition 2. *X is compact.*

Proof. $\forall c \in \mathcal{C}, c \subseteq \overline{c}$ and $\overline{c} \subseteq X$ thus $X = \left(\bigcup_{c \in \mathcal{C}} c\right) \subseteq \left(\bigcup_{c \in \mathcal{C}} \overline{c}\right) \subseteq X$. Hence, all inclusions are equalities, and X is compact as a finite union of compacts. □

Proposition 3. *$\forall c \in \mathcal{C}$, ∂c is compact and closed in X.*

Proof. The boundary of a compact manifold is compact, hence $\partial \mathbb{M}_c$ is compact, and then $\Phi_c(\partial \mathbb{M}_c) =< \mathcal{B}_c >= \partial c$ is compact. Thus, it is closed in X since X is Hausdorff. □

Proposition 4. *$\forall c \in \mathcal{C},\ \ (\partial c, \mathcal{B}_c)$ is a cell complex.*

Proof. $\partial c =< \mathcal{B}_c >$ hence $\mathcal{B}_c$ is a cell decomposition of ∂c. Let $c' \in \mathcal{B}_c$, and $n' = \dim c'$. The existence of a manifold $\mathbb{M}_{c'}$, a map $\Phi_{c'} : \mathbb{M}_{c'} \to X$, decompositions $\mathcal{D}_{c',i}$ of $\mathbb{M}_{c'}$ and cells $v_{c',i} \in \mathcal{C}$, $e_{c',i} \in \mathcal{C}$, and $e_{c',i,j} \in \mathcal{C}$ comes directly from the fact that $(X, \mathcal{C})$ is a cell complex. We only need to verify that $\Phi_{c'} : \mathbb{M}_{c'} \to \partial c$ (instead of X) and that $v_{c',i} \in \mathcal{B}_c$, $e_{c',i} \in \mathcal{B}_c$ and $e_{c',i,j} \in \mathcal{B}_c$ (instead of $\mathcal{C}$).

We know that $c' \in \mathcal{B}_c$, thus $c' \subseteq \partial c$, thus $\overline{c'} \subseteq \partial c$ (because ∂c is closed in X), thus $\Phi_{c'} : \mathbb{M}_{c'} \to \partial c$ (because $\Phi_{c'}(\mathbb{M}_{c'}) = \overline{c'}$). In addition, the cells $v_{c',i}$, $e_{c',i}$ and $e_{c',i,j}$ are images of restrictions of $\Phi_{c'}$ thus are subsets of $\Phi_{c'}(\mathbb{M}_{c'})$, thus are subset of ∂c, hence are elements of $\mathcal{B}_c$. □

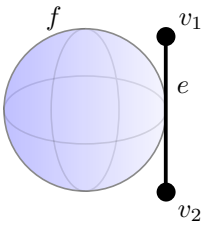
Corollary 1. *$\forall c \in \mathcal{C},\ \ (\overline{c}, \{c\} \cup \mathcal{B}_c)$ is a cell complex.*

Proof. We are just adding c to the complex above, and we know that $\Phi_c : \mathbb{M}_c \to \overline{c}$, as well as the cells $v_{c',i}$, $e_{c',i,j}$ and $e_{c',i}$ are in $\{c\} \cup \mathcal{B}_c$ since they are by definition in $\mathcal{B}_c$. □

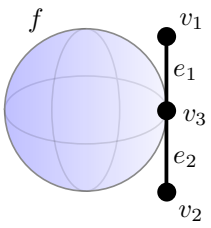
Proposition 5. *$\forall c \in \mathcal{C}$, if $c' \in \mathcal{B}_c$ then $\mathcal{B}_{c'} \subseteq \mathcal{B}_c$. In other words, the relation "c' is in the boundary of c" is transitive:*

$$(\ c'' \in \mathcal{B}_{c'} \ \wedge\ c' \in \mathcal{B}_c\ ) \ \Rightarrow\ c'' \in \mathcal{B}_c$$

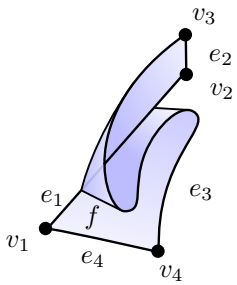
Proof. If $c' \in \mathcal{B}_c$ then $c' \subseteq \partial c$. Hence, $\overline{c'} \subseteq \partial c$ since ∂c is closed, from which it follows that $\partial c' \subseteq \partial c$ since $\partial c' = \overline{c'} \backslash c'$. Thus $< \mathcal{B}_{c'} > \subseteq < \mathcal{B}_c >$, thus $\mathcal{B}_{c'} \subseteq \mathcal{B}_c$. □



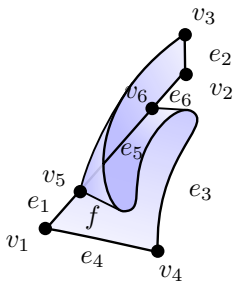
valid CW complex
invalid PCS complex



valid CW complex
valid PCS complex



valid CW complex
invalid PCS complex



valid CW complex
valid PCS complex

$$X = \mathbb{S}^2 \cup \{(1,t,0) \mid t \in [-1,1]\}$$
$$v_1 = \{(1,1,0)\}$$
$$v_2 = \{(1,-1,0)\}$$
$$v_3 = \{(1,0,0)\}$$
$$e = \{(1,t,0) \mid t \in (-1,1)\}$$
$$e_1 = \{(1,t,0) \mid t \in (0,1)\}$$
$$e_2 = \{(1,t,0) \mid t \in (-1,0)\}$$
$$f = \mathbb{S}^2 \setminus \{(1,0,0)\}$$

Figure 18: Left: Counter-example showing that Proposition 1 is not true for CW complexes: ∂f is not equal to any union of other cells, but only "included" in such a union (e.g., $\partial f = \{(1,0,0)\} \subset e$). Right: A valid PCS decomposition of X requires adding the additional vertex v_3 splitting e into e_1 and e_2.

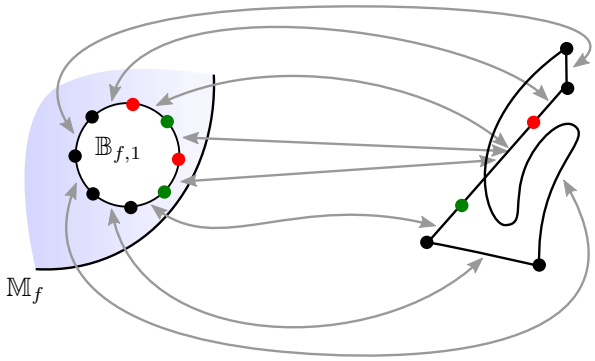


Figure 19: Top-left: The boundary of a CW face is allowed to do "switch-backs" within an edge. Top-right and bottom: A valid PCS decomposition of X requires the additional vertices v_5 and v_6 so that $\Phi_f(\mathbb{B}_{f,1})$ is homeomorphic by part from vertices and edges decomposing $\mathbb{B}_{f,1}$ to vertices and edges of $\mathcal{C}$.

4.3 Comparison with CW complexes

Our definitions of cell, cell decomposition and cell complex differ from the ones of CW complexes, thus there are a few differences that are worth noting. In this section, we use the terms PCS-cell, PCS-cell decomposition and PCS-cell complex to refer to our definition (for arbitrary dimension), while we use the terms CW-cell, CW-cell decomposition and CW-cell complex for the classical definition.

A n-CW-cell is a specific case of n-PCS-cell, since a n-CW-cell is homeomorphic to $\text{int}(\mathbb{D}^n)$, and that $\mathbb{D}^n$ is a connected compact n-manifold. Likewise, a finite CW-cell decomposition is a specific case of a PCS-cell decomposition. Note that PCS-cell decompositions are enforced to be finite, while CW-cell decompositions can be infinite.

However, a finite CW-cell complex is not necessarily a PCS-cell complex. Indeed, even though the definition of PCS-cells is more general than CW-cells, the "gluing conditions" that PCS-cells must satisfy to define a PCS-cell complex are stricter than those for CW-cell complexes. Indeed, we replaced $\Phi_c(\partial\mathbb{M}_c) \subseteq X^{n-1}$ by a stricter version imposing, for each connected component $\mathbb{B}_{c,i}$ of $\partial\mathbb{M}_c$, that $\Phi_c(\mathbb{B}_{c,i})$ is either homeomorphic by part from PCS-cells decomposing $\mathbb{B}_{c,i}$ to PCS-cells of $\mathcal{C}$ (case 1.), or maps $\mathbb{B}_{c,i}$ to a single vertex (case 2.), or, if $\mathbb{B}_{c,i} \cong \mathbb{S}^1$, wraps it around a closed edge (case 3.). In other words, a CW-cell has a lot of freedom on how to be glued to lower dimensional CW-cells, but a PCS-cell must be glued to lower dimensional PCS-cells in a very controlled and non-pathological way. For instance, the boundary of a PCS-face cannot be mapped into a strict subset of an edge (cf. Figure 18), or do "switch-backs" in the interior of an edge (cf. Figure 19). This imposes a cleaner incidence structure as illustrated by Proposition 1, that is not true with CW-cell complexes: there may not exist $\mathcal{B}_c \subseteq \mathcal{C}^{n-1}$ such that $\partial c = < \mathcal{B}_c >$, Figure 18 being a counter-example. This regularity makes possible to describe combinatorially a PCS-cell complex. Note that this regularity is still less strict than the notion of *regular CW complex* [Hatcher, 2001, p. 534], which is too strict to provide uniqueness of a minimal complex.

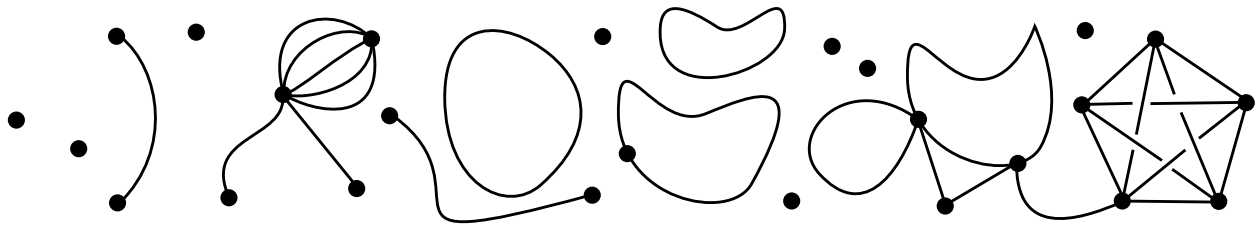


Figure 20: A valid 1-complex. It can be seen as an extension of multigraph to support closed edges.

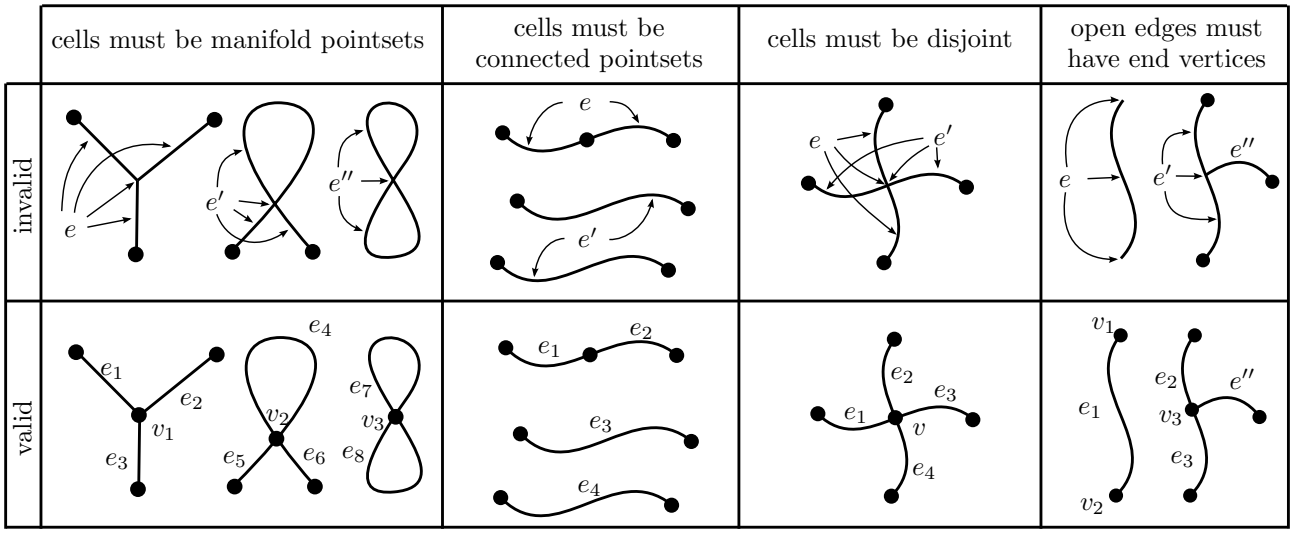


Figure 21: Invalid 1-complexes (top), and how to make them valid (bottom).

4.4 PCS complex

A ***PCS complex*** is defined as a cell complex of dimension at most two. Hence, its cells are either vertices, edges, or faces. In this section, we analyze in depth what the general definition of cell complex means for each type of cells in a PCS complex, which allows us to provide a detailed characterization of them. This characterization can be seen as an alternate, less compact definition of PCS complex, which provides the link between PCS complexes and abstract PCS complexes.

Vertices Vertices are 0-cells, i.e. pointsets homeomorphic to the interior of a connected compact 0-manifold. Up to homeomorphism, there exists only one connected compact 0-manifold:

- $\mathbb{V}$: the singleton $\mathbb{R}^0 = \{0\}$ whose interior is $\mathbb{R}^0$ and boundary is $\varnothing$.

Hence, a pointset v is a vertex if and only if it is a singleton, in which case its characteristic manifold is $\mathbb{M}_v = \mathbb{V}$.

Since $\partial\mathbb{V} = \varnothing$, a vertex automatically satisfies the cell complex constraints. Therefore, cell complexes of dimension 0 *are* finite sets X. They admit a unique cell decomposition $\mathcal{C} = \{\{x\},\ x \in X\}$.

Edges Edges are 1-cells, i.e. pointsets homeomorphic to the interior of a connected compact 1-manifold. Up to homeomorphism, there exist only two connected compact 1-manifolds:

- $\mathbb{E}_|$: the segment $\mathbb{D}^1 = [-1, 1]$ whose interior is $(-1, 1)$ and boundary is $\{-1, 1\}$.
- $\mathbb{E}_\circ$: the circle $\mathbb{S}^1$ whose interior is $\mathbb{S}^1$ and boundary is $\varnothing$.

Hence, a pointset e is an edge if and only if it is homeomorphic to $(-1, 1)$ or $\mathbb{S}^1$. In the first case, it is called an open edge and its characteristic manifold is $\mathbb{M}_e = \mathbb{E}_|$. In the second case, it is called a closed edge and its characteristic manifold is $\mathbb{M}_e = \mathbb{E}_\circ$.

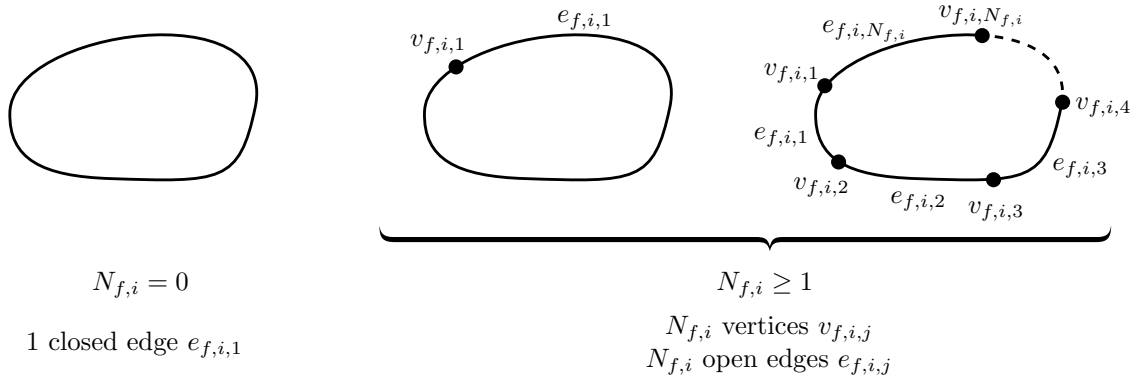


Figure 22: The only possible cell decompositions $\mathcal{D}_{f,i}$ of a boundary component $\mathbb{B}_{f,i}$ of $\mathbb{F}_{\epsilon,g,k}$.

Since $\partial\mathbb{E}_\circ = \varnothing$, a closed edge e automatically satisfies the cell complex constraints. Let e be an open edge. Its characteristic manifold $\mathbb{E}_| = [-1,1]$ has a non-empty boundary $\partial\mathbb{E}_| = \{-1,1\}$ made of two connected components $\mathbb{B}_{e,\text{start}} = \{-1\}$ and $\mathbb{B}_{e,\text{end}} = \{1\}$. For each $\mathbb{B}_{e,i}$, the two cases 1. and 2. of the cell complex constraints are equivalent to:

$$\exists v_{e,i} \in \mathcal{C}, \quad \Phi_e(\mathbb{B}_{e,i}) = v_{e,i}. \tag{12}$$

The case 3. does not apply since $\mathbb{B}_{e,i}$ is not homeomorphic to $\mathbb{S}^1$. Therefore, a decomposition of X into a finite disjoint union of vertices and edges is a cell complex of dimension 1 if and only if:

$$\begin{aligned} &\text{for all open edge } e \in \mathcal{C}, \\ &\text{there exist} \begin{cases} \Phi_e : [-1,1] \to X \text{ continuous} \\ v_{e,\text{start}} \in \mathcal{C} \\ v_{e,\text{end}} \in \mathcal{C} \end{cases} \\ &\text{such that} \begin{cases} \Phi_e : (-1,1) \to e \text{ homeomorphism} \\ \Phi_e(-1) = v_{e,\text{start}} \\ \Phi_e(1) = v_{e,\text{end}} \end{cases} \end{aligned} \tag{13}$$

This is illustrated in Figure 20. Invalid cell complexes of dimension 1 are illustrated in Figure 21.

Faces Faces are 2-cells, i.e. pointsets homeomorphic to the interior of a connected compact 2-manifold. Up to homeomorphism, there exist only three "kinds" of connected compact 2-manifolds (cf. Section 2.3):

- $\mathbb{F}_{\circlearrowleft,0,k}$: the sphere with k holes.
- $\mathbb{F}_{\circlearrowleft,g,k}$: the connected sum of $g \geq 1$ tori with k holes.
- $\mathbb{F}_{\not\circlearrowleft,g,k}$: the connected sum of $g \geq 1$ projective planes with k holes.

Hence, a pointset f is a face if and only if it is homeomorphic to the interior of one of the above manifolds. More formally, f is a face if and only if there exist $\epsilon \in \{\circlearrowleft, \not\circlearrowleft\}$, $g \in \mathbb{N}$ and $k \in \mathbb{N}$ such that $f \cong \text{int}(\mathbb{F}_{\epsilon,g,k})$, in which case it is called an (ϵ,g,k)-face and its characteristic manifold is $\mathbb{M}_f = \mathbb{F}_{\epsilon,g,k}$. All these characteristic manifolds are illustrated in Figure 3 (right).

Now, let us characterize how the boundary components $\mathbb{B}_{f,i}$ of a face f are allowed to be glued to vertices and edges. First, since $\mathbb{M}_f$ is a compact 2-manifold, we know that $\mathbb{B}_{f,i}$ is a compact 1-manifold without boundary, i.e. $\mathbb{B}_{f,i} \cong \mathbb{S}^1$. This means that the three cases of the gluing constraints (cf. Figure 17) have to be considered, and are not equivalent. The cases 2. and 3. do not need further analysis. However, let us explicit what the case 1. means for faces, i.e. let us expand the definition in the specific case where $\mathbb{B}_{f,i} \cong \mathbb{S}^1$. Let us start by the following lemma, illustrated in Figure 22:

Lemma 5. *Let $\mathcal{D}_{f,i}$ be a cell decomposition of $\mathbb{B}_{f,i}$, and let $N_{f,i}$ be the number of vertices in $\mathcal{D}_{f,i}$.*

- *If $N_{f,i} = 0$, then $\mathcal{D}_{f,i} = \{\mathbb{B}_{f,i}\}$, i.e. the decomposition is a single closed edge.*
- *If $N_{f,i} \geq 1$, then $\mathcal{D}_{f,i}$ is decomposed into $N_{f,i}$ vertices and $N_{f,i}$ open edges.*

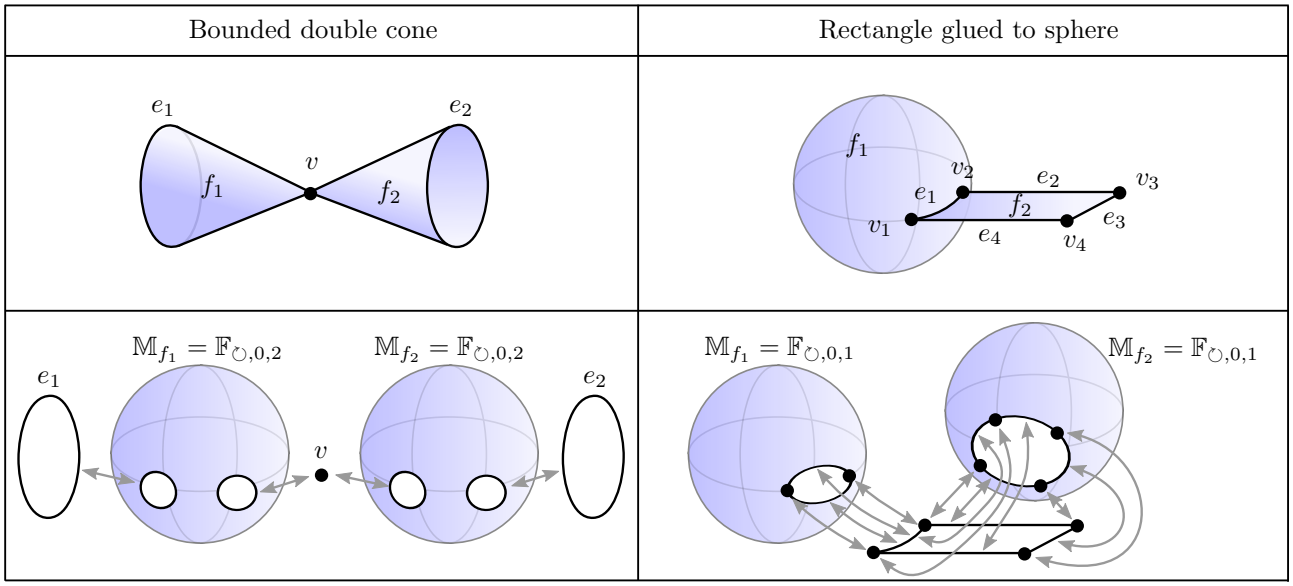


Figure 23: Two examples of valid PCS complexes. Top: The topological space X and its PCS decomposition. Bottom: Characteristic manifolds of the faces, and how their boundary components are glued to the 1-skeleton.

Proof. By definitition, $\mathcal{D}_{f,i}$ is a finite collection of disjoint cells $d_{f,i,j}$ such that $\mathbb{B}_{f,i} = \bigcup_j d_{f,i,j}$. Since $\mathbb{B}_{f,i} \cong \mathbb{S}^1$, it can only involve vertices and/or edges. Let $N_{f,i}^{(v)}$ be the number of vertices and $N_{f,i}^{(e)}$ the number of edges.

- Let assume $N_{f,i}^{(v)} = 0$. Hence, $\mathbb{B}_{f,i}$ is a finite disjoint union of edges. Since $\mathbb{B}_{f,i}$ is compact, but open edges are not compacts (thus a finite union of open edges is not compact either), there exists at least an edge e in $\mathcal{D}_{f,i}$ that is a closed edge. However, the only closed 1-manifold included in $\mathbb{S}^1$ is $\mathbb{S}^1$ itself, thus $N_{f,i}^{(e)} = 1$ and $\mathcal{D}_{f,i} = \{e\}$.
- Let assume $N_{f,i}^{(v)} \geq 1$, and let $v_{f,i,j}$ be the $N_{f,i}^{(v)}$ vertices of $\mathcal{D}_{f,i}$. By fixing $\theta \in [0, 2\pi)$ a parameterization of $\mathbb{B}_{f,i}$, each vertex $v_{f,i,j}$ corresponds to a unique $\theta_{f,i,j}$, and we assume $\theta_{f,i,1} < \theta_{f,i,2} < \cdots < \theta_{f,i,N_{f,i}^{(v)}}$, without loss of generality. Let $e_{f,i,j}$ be the pointset $(\theta_{f,i,j}, \theta_{f,i,j+1}) \subset \mathbb{B}_{f,i}$. Since $e_{f,i,j}$ contains no vertices of $\mathcal{D}_{f,i}$, then $e_{f,i,j}$ is included in a disjoint union of edges in $\mathcal{D}_{f,i}$. None of them can be a closed edge, because it would contradict $N_{f,i}^{(v)} \geq 1$ (see bullet above: if $\mathcal{D}_{f,i}$ contains a closed edge e then $\mathcal{D}_{f,i} = \{e\}$, thus $N_{f,i}^{(v)} = 0$). Hence, $e_{f,i,j}$ is included in a disjoint union of m open edges in $\mathcal{D}_{f,i}$. Besides, it can be shown that a disjoint union of $m \geq 2$ open edges in $\mathbb{S}^1$ is a disconnected set, the connected components being the m open edges. Therefore, since $e_{f,i,j}$ is a connected set, $e_{f,i,j}$ is actually included in a single open edge $e \in \mathcal{D}_{f,i}$. In addition, we know that e contains neither $\theta_{f,i,j}$ nor $\theta_{f,i,j+1}$ (since the cells of $\mathcal{D}_{f,i}$ are disjoint). In conclusion, $e_{f,i,j} = (\theta_{f,i,j}, \theta_{f,i,j+1})$ is included in e, e is connected, and e contains neither $\theta_{f,i,j}$ nor $\theta_{f,i,j+1}$, therefore $e = e_{f,i,j} = (\theta_{f,i,j}, \theta_{f,i,j+1})$. This proves that the $e_{f,i,j}$ are actually open edges of $\mathcal{D}_{f,i}$. In addition, since the union of the $v_{f,i,j}$ and $e_{f,i,j}$ is equal to $\mathbb{B}_{f,i}$, it proves that there are no other cells in $\mathcal{D}_{f,i}$. In conclusion, $N_{f,i}^{(e)} = N_{f,i}^{(v)}$, and $\mathcal{D}_{f,i}$ is a disjoint union of $N_{f,i}^{(v)}$ vertices and $N_{f,i}^{(v)}$ open edges.

□

Using the above lemma, the cell complex constraints for a face $f \in \mathcal{C}$ can be rewritten to:

- Case 1a ($N_{f,i} = 0$): $\mathbb{B}_{f,i}$ is mapped homeomorphically by Φ_f to a single closed edge $e_{f,i} \in \mathcal{C}$, or
- Case 1b ($N_{f,i} \geq 1$): $\mathbb{B}_{f,i}$ is decomposed into $N_{f,i} \geq 1$ vertices, each mapped by Φ_f to a vertex $v_{f,i,j} \in \mathcal{C}$, and $N_{f,i}$ open edges, each mapped homeomorphically by Φ_f to an open edge $e_{f,i,j} \in \mathcal{C}$, or
- Case 2: $\mathbb{B}_{f,i}$ is mapped by Φ_f to a single vertex $v_{f,i} \in \mathcal{C}$, or
- Case 3: $\mathbb{B}_{f,i}$ is mapped by Φ_f by being wrapped $N_{f,i}$ times around a closed edge $e_{f,i} \in \mathcal{C}$.

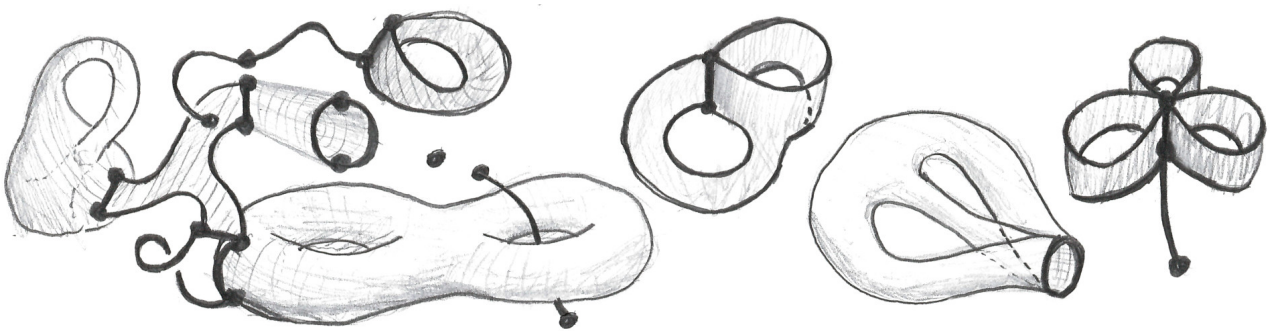


Figure 24: More examples of valid PCS complexes. Is has to be imagined embedded in $\mathbb{R}^4$, i.e. with no "self-intersection" of the Klein bottle or the sphere with three holes glued together.

Finally, we can observe that Case 1a. is already taken into account by Case 3. (wrapping one time around a closed edge), and therefore can be ignored. These three cases are illustrated in Figure 17.

By combining all the information that we have shown, we are finally able to provide a characterization of PCS complexes: a decomposition of X into a finite disjoint union of vertices, edges and faces is a PCS complex if and only if:

$$
\begin{array}{l}
\text{For all open edge } e \in \mathcal{C}, \\
\text{there exist } \left\{ \begin{array}{l} \Phi_e : [-1,1] \to X \text{ continuous} \\ v_{e,\text{start}} \in \mathcal{C} \\ v_{e,\text{end}} \in \mathcal{C} \end{array} \right. \\
\text{such that } \left\{ \begin{array}{l} \Phi_e : (-1,1) \to e \text{ homeomorphism} \\ \Phi_e(-1) = v_{e,\text{start}} \\ \Phi_e(1) = v_{e,\text{end}} \end{array} \right.
\end{array}
$$

$$
\begin{array}{l}
\text{And for all } (\epsilon,g,k)\text{-face } f \in \mathcal{C}, \\
\text{there exist } \left\{ \begin{array}{l} \Phi_f : \mathbb{F}_{\epsilon,g,k} \to X \text{ continuous} \\ \text{a partition of } [1..k] \text{ into } I_{f,1}, I_{f,2} \text{ and } I_{f,3} \\ \forall i \in I_{f,1}, \left\{ \begin{array}{l} N_{f,i} \in \mathbb{N},\ N_{f,i} \geq 1 \\ \forall j \in [1..N_{f,i}],\ v_{f,i,j} \text{ vertex of } \mathcal{C} \\ \forall j \in [1..N_{f,i}],\ e_{f,i,j} \text{ open edge of } \mathcal{C} \end{array} \right. \\ \forall i \in I_{f,2}, \left\{ \begin{array}{l} v_{f,i} \text{ vertex of } \mathcal{C} \end{array} \right. \\ \forall i \in I_{f,3}, \left\{ \begin{array}{l} N_{f,i} \in \mathbb{N},\ N_{f,i} \geq 1 \\ e_{f,i} \text{ closed edge of } \mathcal{C} \end{array} \right. \end{array} \right. \\
\text{such that } \left\{ \begin{array}{l} \Phi_f : \text{int}(\mathbb{F}_{\epsilon,g,k}) \to f \text{ homeomorphism} \\ \forall i \in I_{f,1},\ \forall j \in [1..N_{f,i}], \\ \quad \left\{ \begin{array}{l} \Phi_f(\mathbf{v}_{f,i,j}) = v_{f,i,j} \\ \Phi_f : \mathbf{e}_{f,i,j} \to e_{f,i,j} \text{ homeomorphism} \end{array} \right. \\ \forall i \in I_{f,2},\ \Phi_f(\mathbb{B}_{f,i}) = v_{i,f} \\ \forall i \in I_{f,3},\ \Phi_f : \mathbb{B}_{f,i} \to e_{f,i} \text{ wraps } N_{f,i} \text{ times around } e_{f,i} \end{array} \right. \\
\text{where } \left\{ \begin{array}{l} \mathbb{B}_{f,i} \text{ is the i-th boundary component of } \mathbb{F}_{\epsilon,g,k} \\ \text{and } \{\dots, \mathbf{v}_{f,i,j}, \dots, \mathbf{e}_{f,i,j}, \dots\} \text{ is a} \\ \text{decomposition of } \mathbb{B}_{f,i} \text{ into } N_{f,i} \text{ vertices and} \\ N_{f,i} \text{ open edges, ordered clockwise or counter-clockwise.} \end{array} \right.
\end{array}
\tag{14}
$$

This is, for the case of the dimension 2, an equivalent formulation of the cell complex constraints described in Section 4.1. It is much less compact and does not scale in dimension, but exhaustively describes the different types of cells involved and how they can be glued together. Examples of valid PCS complex are given Figure 23 and 24.

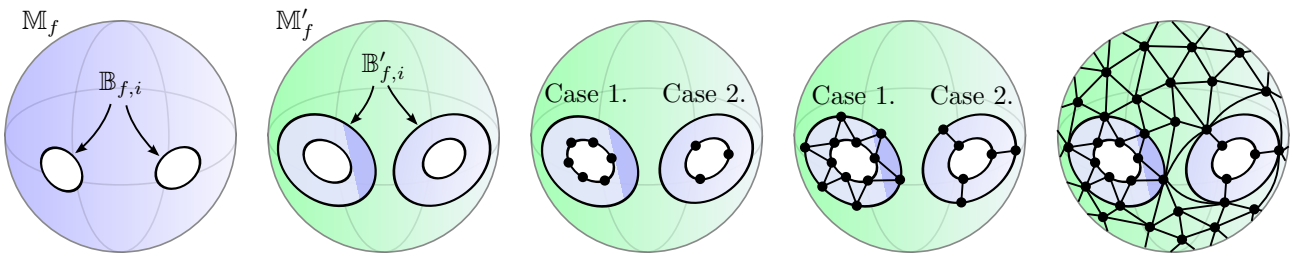


Figure 25: Steps in the construction of the mixed triangulation-quadrangulation of $\mathbb{M}_f$, from the proof of Proposition 7.

4.5 Equivalence between PCS-decomposable spaces and 2-triangulable spaces

In this section, we show that the class of topological spaces that can be decomposed as a PCS complex is the same as the class of topological spaces that admits a 2-triangulation. This shows that PCS complexes are able to represent any "reasonable" two-dimensional object.

Proposition 6. *A topological space that admits a 2-triangulation can be decomposed as a PCS complex.*

Proof. This proposition comes directly from the observation that a 2-triangulation of a space X is in fact also a valid PCS decomposition. Indeed, a 0-simplex is a PCS vertex, a 1-simplex is a PCS open edge, and a 2-simplex is a PCS face whose characteristic manifold is $\mathbb{F}_{\circlearrowleft,0,1}$ (the sphere with one hole), with its unique boundary component $\mathbb{B}_{f,1}$ decomposed into three vertices and three open edges, each mapped homeomorphically to vertices and open edges. □

Proposition 7. *A topological space that can be decomposed as a PCS complex admits a 2-triangulation.*

Proof. To prove this statement, we provide an explicit construction of the triangulation. Let $\mathcal{K} = (X, \mathcal{C})$ be a PCS complex. Without loss of generality, we assume that $\mathcal{C}$ does not contain any closed edge. Indeed, any valid PCS decomposition $(X, \mathcal{C})$ can be preliminary turned into a valid PCS decomposition $(X, \mathcal{C}')$ that does not contain closed edges, by partitioning every closed edge of $\mathcal{C}$ into an open edge and a vertex. Now, let us start the construction of the triangulation:

- The vertices in $\mathcal{C}$ are the 0-simplices of the triangulation.
- The open edges in $\mathcal{C}$ are all split into three 1-simplices and two 0-simplices.

These two first steps have constructed a *valid* 1-triangulation of the 1-skeleton of X. Splitting each edge in three is necessary to ensure that each 1-simplex has a different start and end 0-simplex, and that any given pair of 0-simplices is connected by *at most* one 1-simplex. Now, let us triangulate the faces:

- Let f be a face in $\mathcal{C}$.
- Let $\mathbb{M}_f$ be the characteristic manifold of f, Φ_f be the characteristic map, and $\mathbb{B}_{f,i} \cong \mathbb{S}^1$ be the boundary components of $\mathbb{M}_f$.
- Let $\mathbb{M}'_f$ be a compact submanifold of $\mathbb{M}_f$ obtained by "offsetting by a small amount" the holes of $\mathbb{M}_f$ (cf. Figure 25(b)). We call $\mathbb{B}'_{f,i}$ the boundary components of $\mathbb{M}'_f$.
- For each boundary component $\mathbb{B}_{f,i}$ mapped by Φ_f to vertices and open edges (Case 1., i.e. when $i \in I_{f,1}$ from the characterization Eq. 14), we 1-triangulate $\mathbb{B}_{f,i}$ by using the pre-image by Φ_f of the previously constructed 1-triangulation of ∂f (cf. Figure 25(c), left hole). This 1-triangulation has $3N_{f,i}$ 0-simplices, and the same number of 1-simplices.
- For each boundary component $\mathbb{B}_{f,i}$ mapped by Φ_f to a single vertex (Case 2.), we arbitrarily 1-triangulate $\mathbb{B}_{f,i}$ using three 0-simplices, and three 1-simplices (cf. Figure 25(c), right hole). We note that Case 3. can be ignored since $\mathcal{C}$ does not contain any closed edge.

- For each boundary component $\mathbb{B}_{f,i}$ in Case 1., we arbitrarily 1-triangulate $\mathbb{B}'_{f,i}$ using $3N_{f,i}$ 0-simplices and $3N_{f,i}$ 1-simplices. Then, we 2-triangulate the topological cyclinder between $\mathbb{B}_{f,i}$ and $\mathbb{B}'_{f,i}$ using the pattern illustrated in Figure 25(d), left hole.
- For each boundary component $\mathbb{B}_{f,i}$ in Case 2., we arbitrarily 1-triangulate $\mathbb{B}'_{f,i}$ using three 0-simplices and three 1-simplices. Then, we *quadrangulate* the topological cylinder between $\mathbb{B}_{f,i}$ and $\mathbb{B}'_{f,i}$ using the pattern illustrated in Figure 25(d), right hole.
- Finally, we 2-triangulate $\mathbb{M}'_f$ by preserving the existing 1-triangulation of its boundary, that we know is possible since $\mathbb{M}'_f$ is a compact 2-manifold (cf. Figure 25(e)).
- At this stage of the construction, we have obtained a mixed triangulation-quadrangulation $\mathcal{T}$ of $\mathbb{M}_f$. To conclude the construction, we define the triangulation of f to be the image of $\mathcal{T}$ by Φ_f.

The reader can verify that the quads of $\mathbb{M}_f$ become triangles of f since in Case 2., $\mathbb{B}_{f,i}$ shrinks to a single vertex. Also, due the homeomorphism properties of Φ_f, the triangles stay triangles, no 1-simplex of f has its start 0-simplex equal to its end 0-simplex, and no pair of 0-simplices are connected by 2 or more 1-simplices (this is guaranteed by the specific triangle pattern chosen around $\mathbb{B}_{f,i}$ for Case 1.). Therefore, by performing this process for all faces, we obtain a valid 2-triangulation of X. □

5 Abstract PCS complex

In this section, we define the notion of *abstract PCS complex*, independently from the notion of PCS complex. It is a purely combinatorial and finite structure, similar to the concept of *polygon presentation*. It is equivalent to the VGC [Dalstein et al., 2014] to which we add the information of orientability and genus of faces. Then, from an abstract PCS complex $\mathcal{P}$, we define a topological space $|\mathcal{P}|$, which we call the *geometric realization* of $\mathcal{P}$, and we show that it is a PCS complex. Finally, we show that for every PCS complex $\mathcal{K}$, there exists an abstract PCS complex $\mathcal{P}$ such that $|\mathcal{P}|$ is homeomorphic to $\mathcal{K}$. This means that PCS complexes can unambiguously (up to homeomorphism), be described purely combinatorially as abstract PCS complexes. Abstract PCS complexes are very similar to vector graphics complexes, but we adopt here a different formalism more adapted for subsequent analysis.

5.1 Definition

An
terminologyabstract PCS complex, is a tuple $\mathcal{P} = (C, \dim, \epsilon, g, \widehat{\partial})$ such that:

- C is a finite set. Its elements are called ***abstract cells***.
- $\dim : C \to \{0, 1, 2\}$ is a function specifying the ***dimension*** of an abstract cell. We use this function to define the sets of ***abstract vertices***, ***abstract edges***, and ***abstract faces*** by:

$$V = \{c \in C \mid \dim(c) = 0\} \tag{15}$$
$$E = \{c \in C \mid \dim(c) = 1\} \tag{16}$$
$$F = \{c \in C \mid \dim(c) = 2\} \tag{17}$$

  For conciseness, we omit the term "abstract" when it is clear from the context that we are referring to an abstract cell of an abstract PCS complex, as opposed to a cell of a PCS complex.
- $\epsilon : F \to \{\circlearrowleft, \not\circlearrowleft\}$ is a function specifying the ***orientability*** of a face.
- $g : F \to \mathbb{N}$ is a function specifying the ***genus*** of a face, satisfying the following constraint to ensure that non-orientable faces have strictly positive genuses:

$$\forall f \in F, \quad (\epsilon(f) = \not\circlearrowleft) \;\Rightarrow\; (g(f) \geq 1) \tag{18}$$

- $\widehat{\partial} : C \to \widehat{\mathbf{V}} \cup \widehat{\mathbf{E}} \cup \widehat{\mathbf{F}}$ is a function specifying the ***semantic boundary*** of a cell, satisfying:

$$\forall v \in V, \quad \widehat{\partial} v \in \widehat{\mathbf{V}} \tag{19}$$

$$\forall e \in E, \quad \widehat{\partial} e \in \widehat{\mathbf{E}} \tag{20}$$

$$\forall f \in F, \quad \widehat{\partial} f \in \widehat{\mathbf{F}} \tag{21}$$

  where the sets $\widehat{\mathbf{V}}$, $\widehat{\mathbf{E}}$, and $\widehat{\mathbf{F}}$ are defined in the remaining of this section.

A ***vertex semantic boundary*** is defined as the empty set. Hence, the set of all possible vertex semantic boundaries is

$$\widehat{\mathbf{V}} = \{\varnothing\}. \tag{22}$$

An ***edge semantic boundary*** is defined as either the empty set or an ordered pair of edges $(v_{\text{start}}, v_{\text{end}}) \in V \times V$. Hence, the set of all possible edge semantic boundaries is

$$\widehat{\mathbf{E}} = \{\varnothing\} \cup (V \times V). \tag{23}$$

$\widehat{\partial} e$ specifies not only the vertices in the boundary of e, but also, when $\widehat{\partial} e \neq \varnothing$, it specifies a *natural orientation* of the edge, from v_{start} to v_{end}. An edge such that $\widehat{\partial} e = \varnothing$ is called a ***closed edge***, otherwise it is called an ***open edge***. We denote by $E_\circ$ the set of closed edges and by $E_|$ the set of open edges:

$$E_\circ = \{e \in E \mid \widehat{\partial} e = \varnothing\} \tag{24}$$

$$E_| = \{e \in E \mid \widehat{\partial} e \in V \times V\} \tag{25}$$

A ***halfedge***, or ***oriented edge***, is defined by an edge and a chosen orientation, i.e. a pair $h = (e, \beta) \in E \times \{\top, \bot\}$. If $\beta = \top$, then the edge is considered in its natural orientation, otherwise it is considered in the opposite orientation. The set of all halfedges is denoted

$$H = E \times \{\top, \bot\}. \tag{26}$$

A halfedge $h = (e, \beta)$ is called a ***closed halfedge*** if e is a closed edge, and an ***open halfedge*** otherwise. We denote by $H_\circ$ the set of closed halfedges, and by $H_|$ the set of open halfedges and:

$$H_\circ = E_\circ \times \{\top, \bot\} \tag{27}$$

$$H_| = E_| \times \{\top, \bot\} \tag{28}$$

We define the functions

$$\begin{array}{rccc} \text{start}: & H_| & \to & V \\ & (e, \top) & \mapsto & v_{\text{start}} \\ & (e, \bot) & \mapsto & v_{\text{end}} \end{array} \tag{29}$$

and

$$\begin{array}{rccc} \text{end}: & H_| & \to & V \\ & (e, \top) & \mapsto & v_{\text{end}} \\ & (e, \bot) & \mapsto & v_{\text{start}} \end{array} \tag{30}$$

where

$$(v_{\text{start}}, v_{\text{end}}) = \widehat{\partial} e. \tag{31}$$

We denote the set of stricly positive integers as $\mathbb{N}^+ = \mathbb{N} \backslash \{0\}$. A ***cycle*** γ is defined as either:

1. a non-empty sequence of open halfedges $(h_j)_{j \in [1..N]} \in H_|^+$ such that

$$\forall j \in [1..N], \ \text{end}(h_j) = \text{start}(h_{(j+1) \bmod N}) \tag{32}$$

   where $H_|^+ = \cup_{N \geq 1} H_|^N$, and $j \bmod N$ refers to the integer $j' \in [1..N]$ such that $j \equiv j' \pmod N$, or
2. a vertex $v \in V$, or
3. a pair $(h, N) \in H_\circ \times \mathbb{N}^+$ representing a closed halfedge repeated N times.

In the first case, the cycle γ is called a ***non-simple cycle***, in the second case, it is called a ***Steiner cycle***, and in the third case, it is called a ***simple cycle***. Formally, the sets of all possible non-simple cycles, Steiner cycles, and simple cycles are respectively defined by

$$\Gamma_+ = \{(h_j) \in H_|^+ \mid (32)\} \tag{33}$$

$$\Gamma_\bullet = V \tag{34}$$

$$\Gamma_\circ = H_\circ \times \mathbb{N}^+ \tag{35}$$

and the set of all possible cycles is defined by

$$\Gamma = \Gamma_+ \cup \Gamma_\bullet \cup \Gamma_\circ. \tag{36}$$

Finally, a ***face semantic boundary*** is defined as a (possibly empty) sequence of cycles. Hence, with $\Gamma^* = \cup_{N \geq 0} \Gamma^N$, the set of all possible face semantic boundaries is defined by

$$\widehat{\mathbf{F}} = \Gamma^*. \tag{37}$$

5.2 Convenient notations

Vertex We use the notation v to refer to a vertex.

Edge We use the notation e to refer to an edge that can be either open or closed.

Open edge We use the notation $e^|$ to refer to an open edge, or simply e when it is clear from the context that the edge is open. To conveniently refer to or define an open edge e together with its semantic boundary $\widehat{\partial} e$, we use the following abuse of notation:

$$e = (v_{\text{start}}, v_{\text{end}}) \tag{38}$$

We conveniently refer to these objects by $v_{\text{start}}(e)$ and $v_{\text{end}}(e)$.

Closed edge We use the notation e° to refer to a closed edge, or simply e when it is clear from the context that the edge is closed.

Halfedge We use the notation $h = (e, \beta)$, with $\beta \in \{\top, \bot\}$, to refer to or define a halfedge that can be either open or closed. We conveniently use the notation $e(h)$ and $\beta(h)$ to refer to the components of the halfedge. We use the notation h° to refer specifically to a closed halfedge, and the notation $h^|$ to refer specifically to an open halfedge. Finally, we sometimes use the abuse of notation $v_{\text{start}}(h) = \text{start}(h)$ and $v_{\text{end}}(h) = \text{end}(h)$.

Cycle We use the notation γ to refer to a cycle that can be Steiner, simple or non-simple.

Steiner cycle We use the notation $\gamma^\bullet$ to refer to a Steiner cycle, or simply γ when it is clear from the context that we refer to a Steiner cycle. We use the notation $v(\gamma^\bullet)$ to refer to the vertex that defines it, and the following notation to refer to or define a Steiner cycle together with its vertex v:

$$\gamma^\bullet = [v] \tag{39}$$

Simple cycle We use the notation γ° to refer to a simple cycle, or simply γ when it is clear from the context that we refer to a Steiner cycle. We use the notations $h^\circ(\gamma^\circ)$ and $N(\gamma^\circ)$ to refer to the closed halfedge and integer that define it. We also use the convenient notations $e^\circ(\gamma^\circ) = e(h^\circ(\gamma^\circ))$ and $\beta(\gamma^\circ) = \beta(h^\circ(\gamma^\circ))$. Finally, we use the following notation to refer to or define a Steiner cycle together with its defining components:

$$\gamma^\circ = [h^{\circ N}] = [(e^\circ, \beta)^N] \tag{40}$$

Non-simple cycle We use the notation γ^+ to refer to a non-simple cycle, or simply γ when it is clear from the context that we refer to a non-simple cycle. We use the following notation to refer to or define a non-simple cycle together with its defining open halfedges:

$$\gamma^+ = [h_1, \dots, h_N] = [(e_1, \beta_1), \dots, (e_N, \beta_N)] \tag{41}$$

We conveniently refer to these objects by $N(\gamma)$, $h_j(\gamma)$, $e_j(\gamma)$ and $\beta_j(\gamma)$, where $j \in \mathbb{N}$ is considered modulo N (e.g., $h_0(\gamma)$ is well-defined and refers to h_N). We use the notation $v_i(\gamma) = v_{\text{end}}(h_i(\gamma))$. In particular, we have $v_0(\gamma) = v_N(\gamma)$, and to conveniently visualize all the cells involved in a non-simple cycle, we use the notation:

$$\gamma^+ = [\ \underset{v_0}{\bullet}\ (e_1, \beta_1)\ \underset{v_1}{\bullet}\ \cdots\ \underset{v_{N-1}}{\bullet}\ (e_N, \beta_N)\ \underset{v_N}{\bullet}\ ] \tag{42}$$

Face We use the notation f to refer to a face. To conveniently refer to or define a face f together with its semantic boundary $\widehat{\partial} f$, we use the following abuse of notation:

$$f = (\epsilon, g, [\gamma_1, \dots, \gamma_k]) \tag{43}$$

or simply

$$f = [\gamma_1, \dots, \gamma_k] \tag{44}$$

when ϵ and g are irrelevant or clear from context. We conveniently refer to these objects by $\epsilon(f)$, $g(f)$, $k(f)$, and $\gamma_i(f)$. It is possible that $f = [\ ]$, in which case f is a face without boundary (we do not use a special notation for faces without boundary).

5.3 Geometric realization

In this section, we define for each abstract PCS complex $\mathcal{P}$ a PCS complex $|\mathcal{P}|$ that we call the ***geometric realization*** of $\mathcal{P}$. Let $\mathcal{P} = (C, \dim, \epsilon, g, \widehat{\partial})$ be an abstract PCS complex. First, for each abstract cell $c \in C$, let $\mathbb{M}_c$ be the compact manifold defined by:

- If $c \in V$, then $\mathbb{M}_c = \mathbb{V}$.
- If $c \in E_\circ$, then $\mathbb{M}_c = \mathbb{E}_\circ$
- If $c \in E_|$, then $\mathbb{M}_c = \mathbb{E}_|$
- If $c \in F$, then $\mathbb{M}_c = \mathbb{F}_{\epsilon(c), g(c), k(c)}$.

Then, let Y be the topological space defined by the topological disjoint union of all $\mathbb{M}_c$, i.e. $Y = \amalg_{c \in C} \mathbb{M}_c$. By "topological disjoint union", we mean that even though the characteristic manifolds of different cells can be equal (e.g., $\mathbb{M}_v = \mathbb{M}_{v'} = \mathbb{V} = \{0\}$), they are "duplicated" to appear as many times in Y. Formally, this is achieved by pairing the objects to unite with an identifier (e.g., $\mathbb{M}_v \amalg \mathbb{M}_{v'} = \mathbb{V} \amalg \mathbb{V} = \{(x, 1) \mid x \in \mathbb{V}\} \cup \{(x, 2) \mid x \in \mathbb{V}\} = \{(0, 1), (0, 2)\}$), and is well covered in textbooks on algebraic topology. In our case, the abstract cells themselves can take the role of identifiers, therefore we have

$$\mathbb{M}'_c = \{(x, c) \mid x \in \mathbb{M}_c\} \tag{45}$$

$$Y = \bigcup_{c \in C} \mathbb{M}'_c \tag{46}$$

The last step is to define, for each cell $c \in C$, a map $\Phi_c : \mathbb{M}'_c \to Y$ that we call a ***quotient map***. We will use these maps to define the topological space X as the quotient space $X = Y/\sim$, where $\sim$ is the smallest equivalence relation containing $x \sim \Phi_c(x)$ for all $c \in C$ and all $x \in \mathbb{M}'_c$. This means that if $\Phi_c(x) = y$ in Y, then x and y become a single element in X. In other words, every point $x \in \mathbb{M}'_c$ is *glued* to its image by Φ_c. We now define these maps. Let $c \in C$ be an abstract cell.

- If $c \in V$, we recall that $\mathbb{M}_c = \mathbb{V} = \{0\}$. Therefore, $\mathbb{M}'_c$ has a unique element $\{(0, c)\}$, and we define:

$$\begin{array}{rccc} \Phi_c : & \mathbb{M}'_c & \to & Y \\ & (0, c) & \mapsto & (0, c) \end{array} \tag{47}$$

- If $c \in E_\circ$, we recall that $\mathbb{M}_c = \mathbb{S}^1$. With $\theta \in [0, 2\pi)$ referring to the usual parameterization of $\mathbb{S}^1$, we define:

$$\begin{array}{rccc} \Phi_c : & \mathbb{M}'_c & \to & Y \\ \forall \theta \in [0, 2\pi), & (\theta, c) & \mapsto & (\theta, c) \end{array} \tag{48}$$

- If $c \in E_|$ then there exist abstract vertices v_{start} and v_{end} such that $\widehat{\partial} c = (v_{\text{start}}, v_{\text{end}})$. Also, we recall that $\mathbb{M}_c = [-1, 1]$. We define:

$$\begin{array}{rccc} \Phi_c : & \mathbb{M}'_c & \to & Y \\ & (-1, c) & \mapsto & (0, v_{\text{start}}) \\ \forall x \in (-1, 1), & (x, c) & \mapsto & (x, c) \\ & (1, c) & \mapsto & (0, v_{\text{end}}) \end{array} \tag{49}$$

- If $c \in F$, then there exist g,ϵ, and γ_i such that $\widehat{\partial} c = (g, \epsilon, [\gamma_1, \ldots, \gamma_k])$, and we have $\mathbb{M}_c = \mathbb{F}_{g,\epsilon,k}$. Therefore, $\mathbb{M}'_c$ has k boundary components $\mathbb{B}'_{c,i}$ all homeomorphic to $\mathbb{S}^1$, that we parameterize by (θ, c, i) with $\theta \in [0, 2\pi)$. This means that by (θ, c, i), we refer to a point $(x, c) \in \mathbb{B}'_{c,i} \subset \mathbb{M}'_c \subset Y$. We define the map $\Phi_c : \mathbb{M}'_c \to Y$ by:
  - $\forall x \in \text{int}(\mathbb{M}_f)$:

$$(x, c) \mapsto (x, c) \tag{50}$$

  - $\forall i \in [1..k]$:
    - If $\gamma_i = [v]$:

$$\forall \theta \in [0, 2\pi), \quad (\theta, c, i) \mapsto (0, v) \tag{51}$$

    - If $\gamma_i^\circ = [(e^\circ, \top)^N]$:

$$\forall \theta \in [0, 2\pi), \quad (\theta, c, i) \mapsto (N\theta, e^\circ) \tag{52}$$

    - If $\gamma_i^\circ = [(e^\circ, \bot)^N]$:

$$\forall \theta \in [0, 2\pi), \quad (\theta, c, i) \mapsto (-N\theta, e^\circ) \tag{53}$$

    - If $\gamma_i = [\underset{v_0}{\bullet} (e_1, \beta_1) \underset{v_1}{\bullet} \cdots \underset{v_{N-1}}{\bullet} (e_N, \beta_N) \underset{v_N}{\bullet}]$, then $\forall j \in [1..N]$:

$$\text{If } \beta_i = \top : \quad \forall \theta \in \left[\frac{2(j-1)\pi}{N}, \frac{2j\pi}{N}\right), \quad (\theta, c, i) \mapsto (2u - 1, e_j) \tag{54}$$

$$\text{If } \beta_i = \bot : \quad \forall \theta \in \left[\frac{2(j-1)\pi}{N}, \frac{2j\pi}{N}\right), \quad (\theta, c, i) \mapsto (1 - 2u, e_j) \tag{55}$$

      Where u is the value in $[0, 1)$ such that $\theta = \frac{2(j-1+u)\pi}{N}$, i.e.:

$$u = 1 - j + \frac{N\theta}{2\pi} \tag{56}$$

As announced, we conclude the construction by defining $|\mathcal{P}| = (X, \mathcal{C})$, where $X = Y/\sim$, and $\mathcal{C} = \{|c| : c \in C\}$, where $|c|$ refers to the set of equivalence classes of $x \in X$ corresponding to the points $y \in \text{int}(\mathbb{M}'_c) \subset Y$. The cells $|c|$ in $\mathcal{C}$ are actual topological cells, unlike the abstract cells c in C which are only symbols/identifiers. The interested reader can verify that $|\mathcal{P}|$ is indeed a PCS complex. This is achieved by using the characterization of PCS complex from Equation 14. The existence of all required objects comes from the explicit construction of $|\mathcal{P}|$.

5.4 Presentation scheme

Let $\mathcal{K} = (X, \mathcal{C})$ be a PCS complex. In a largely symmetric fashion of the previous section, starting from the characterization in Equation 14, we can define an abstract PCS complex $\mathcal{K}^A$, called the ***presentation scheme*** of $\mathcal{K}$, such that $\mathcal{K}$ and $|\mathcal{K}^A|$ are homeomorphic. By "PCS complex homeomorphism", we mean that not only there is a homeomorphism between the two topological spaces, but also that this homeomorphism maps cells of $\mathcal{K}$ to cells of $|\mathcal{K}^A|$. We detail here how to construct $\mathcal{K}^A = (C, \dim, \epsilon, g, \widehat{\partial})$ from $\mathcal{K}$:

- Let N_c be the number of cells in $\mathcal{C}$. We define $C = [1..N_c]$. Since $\mathcal{C}$ and C have the same cardinal N_c, there exists a bijection σ from cells in $\mathcal{C}$ to abstract cells in C. For each cell $c \in \mathcal{C}$, we use the notation $c^A \in C$ to refer to the abstract cell $\sigma(c)$ corresponding to the cell $c \in \mathcal{C}$.
- For each vertex $v \in \mathcal{C}$, we define $\dim v^A = 0$, and we define $\widehat{\partial} v^A = \varnothing$.
- For each closed edge $e \in \mathcal{C}$, we define $\dim e^A = 1$, and we define $\widehat{\partial} e^A = \varnothing$.
- For each open edge $e \in \mathcal{C}$, we define $\dim e^A = 1$. Let $v_{e,\text{start}} \in \mathcal{C}$ and $v_{e,\text{end}} \in \mathcal{C}$ be the vertices provided by Eq. 14. We define $\widehat{\partial} e^A = (v^A_{e,\text{start}}, v^A_{e,\text{end}})$.
- For each face $f \in \mathcal{C}$, we define $\dim f^A = 2$. Let $\mathbb{F}_{\epsilon,g,k}$ be the characteristic manifold of f. We define $\epsilon(f^A) = \epsilon$ and $g(f^A) = g$. Let $I_{f,1}$, $I_{f,2}$, and $I_{f,3}$ be the partition of $[1..k]$ from Eq. 14. We define $\widehat{\partial} f^A = [\gamma_1, \ldots, \gamma_k]$, where the γ_i are defined by:
  - If $i \in I_{f,1}$, then let $e_{f,i,j}$ be the $N_{f,i}$ open edges from Eq. 14. Let $\theta \in [0, 2\pi)$ be a parameterization of $\mathbb{B}_{f,i}$ such that the $\mathbf{e}_{f,i,j}$ are ordered increasingly. If f is orientable, the chosen orientation must be consistent across all the $\mathbb{B}_{f,i}$ (i.e. all clockwise or all counter-clockwise). Let $x \in (-1, 1)$ be the parameterization of $e_{f,i,j}$ provided by $\Phi_{e_{f,i,j}}$. Using these parameterizations, then for all $j \in [1..N_{f,i}]$, Φ_f restricted to $\mathbf{e}_{f,i,j}$ is a bijective function from a subinterval of $[0, 2\pi)$ to $(-1, 1)$. Thus, it is either strictly increasing or strictly decreasing. If it is strictly increasing, we define $\beta_j = \top$, otherwise we define $\beta_j = \bot$. Finally, we define $\gamma_i = [(e^A_{f,i,1}, \beta_1), \ldots, (e^A_{f,i,N}, \beta_N)]$ (non-simple cycle).
  - If $i \in I_{f,2}$, then let $v_{f,i}$ be the vertex from Eq. 14. We define $\gamma_i = [v^A_{f,i}]$ (Steiner cycle).
  - If $i \in I_{f,3}$, then let $N_{f,i}$ and $e_{f,i}$ be the integer and the closed edge from Eq. 14. Let $\theta \in [0, 2\pi)$ be the parameterization of $\mathbb{B}_{f,i}$ already defined above, and let $\theta \in [0, 2\pi)$ be the parameterization of $e_{f,i}$ provided by $\Phi_{e_{f,i}}$. Using these parameterizations, if Φ_f restricted to $\mathbb{B}_{f,i}$ is strictly increasing, then we define $\beta = \top$, otherwise we define $\beta = \bot$. Finally, we define $\gamma_i = [(e^A_{f,i}, \beta)^{N_{f,i}}]$ (simple cycle).

Due to the fact that the construction of $\mathcal{K}^A$ from $\mathcal{K}$ is symmetric to the construction of $|\mathcal{P}|$ from $\mathcal{P}$, the reader can verify that $|\mathcal{K}^A|$ and $\mathcal{K}$ are homeomorphic. In other words, an abstract PCS complex is a combinatorial structure that describes a PCS complex up to homeomorphism, and conversely every PCS complex admits an abstract PCS complex that describes it.

6 Algebraic operations on halfedges, paths and cycles

In order to describe more conveniently the topological operators on abstract PCS complexes (cf. Section 7), we first introduce the notion of *paths*, and a few basic algebraic operations on halfedges, paths and cycles, which are: flipping a halfedge, a path, or a cycle; converting an open halfedge to a path and a path to a cycle; concatenating paths to create a longer path; rotating a non-simple cycle; and extracting a subpath from a path or a non-simple cycle.

6.1 Paths

Given an abstract PCS complex $\mathcal{P}$, a ***path*** is defined as a triplet $\pi = (v_{\text{start}}, (h_j)_{j \in [0..N]}, v_{\text{end}}) \in \Pi = V \times H^*_{|} \times V$ satisfying the following constraints:

- if $N = 0$ (i.e., the sequence (h_j) is empty), then $v_{\text{start}} = v_{\text{end}}$
- if $N > 0$ (i.e., the sequence (h_j) is not empty), then
  - $v_{\text{start}} = \text{start}(h_1)$
  - $\forall j \in [1..N-1], \text{end}(h_j) = \text{start}(h_{(j+1)})$
  - $\text{end}(h_N) = v_{\text{end}}$

Intuitively, a path starts at a given vertex v_{start}, then travels along $N \geq 0$ edges e_i with a given orientation β_i, and finally ends its course at a vertex v_{end}. If $N = 0$, we conveniently use the notation $\pi = [v]$ instead of $\pi = (v, [\,], v)$. If $N > 0$, we conveniently use the notation $\pi = [h_1, \ldots, h_N]$ instead of $\pi = (v_{\text{start}}, [h_1, \ldots, h_N], v_{\text{end}})$, since the start and end vertices can be inferred from the halfedges. The integer $N \in \mathbb{N}$ is called the ***length*** of the path. While the notion of path shares similarities with the notion of cycle (e.g., can be reduced to a single vertex), we note that the concept of "simple path" does not exist: a path necessarily starts and ends at given vertices (possibly equal), therefore it cannot contain closed edges. To better emphasize the differences between paths and cycles, we use the following terminology: if $N = 0$, we refer to the path as a ***trivial path*** (rather than a "Steiner path"); and if $N > 0$, we refer to the path as a ***non-trivial path*** (rather than a "non-simple path").

6.2 Flipping halfedges, paths and cycles

Given a halfedge $h = (e, \beta)$, we define its ***flipped halfedge*** as:

$$\overline{h} = (e, \overline{\beta}), \quad \text{where } \overline{\beta} = \begin{cases} \bot & \text{if } \beta = \top \\ \top & \text{if } \beta = \bot \end{cases} \tag{57}$$

Given a path $\pi = (v_{\text{start}}, [h_1, \ldots, h_N], v_{\text{end}})$, we define its ***flipped path*** as:

$$\overline{\pi} = (v_{\text{end}}, [\overline{h_N}, \ldots, \overline{h_1}], v_{\text{start}}) \tag{58}$$

Given a cycle γ, we define its ***flipped cycle*** as:

$$\overline{\gamma} = \begin{cases} [v] & \text{if } \gamma = [v] \text{ is a Steiner cycle} \\ [\overline{h^{\circ}}^{N}] & \text{if } \gamma = [h^{\circ N}] \text{ is a simple cycle} \\ [\overline{h_N}, \ldots, \overline{h_1}] & \text{if } \gamma = [h_1, \ldots, h_N] \text{ is a non-simple cycle} \end{cases} \tag{59}$$

6.3 Converting open halfedges to paths and paths to cycles

An open halfedge h can always be interpreted as a path π of length $N(\pi) = 1$, using the following conversion:

$$\begin{array}{rcl} H_{|} & \to & \Pi \\ h & \mapsto & [h] = (v_{\text{start}}(h), [h], v_{\text{end}}(h)) \end{array} \tag{60}$$

For conciseness, we will often omit the brackets and simply write h instead of $[h]$ when it is clear from the context that we interpret h as a path.

Similarly, a path satisfying $v_{\text{start}} = v_{\text{end}}$ can always be interpreted as a cycle (more specifically, a Steiner cycle if $N = 0$ and a non-simple cycle if $N > 0$), using the following conversion:

$$\begin{array}{rcl} \{\pi \in \Pi \mid v_{\text{start}} = v_{\text{end}}\} & \to & \Gamma \\ \pi & \mapsto & [\pi) = \begin{cases} [v_{\text{start}}] & \text{if } N = 0 \\ [h_1, \ldots, h_N] & \text{if } N > 0 \end{cases} \end{array} \tag{61}$$

For conciseness, we will often omit the brackets and parentheses and simply write π instead of $[\pi)$ when it is clear from the context that we interpret π as a cycle.

6.4 Concatenating paths

Given two paths $\pi = (v_{\text{start}}, [h_1, \dots, h_N], v_{\text{end}})$ and $\pi' = (v'_{\text{start}}, [h'_1, \dots, h'_{N'}], v'_{\text{end}})$ satisfying $v_{\text{end}} = v'_{\text{start}}$, we define the ***concatenation*** of π with π' by:

$$[\pi, \pi'] = (v_{\text{start}}, [h_1, \dots, h_N, h'_1, \dots, h'_{N'}], v'_{\text{end}}) \tag{62}$$

Since this operation is associative (i.e., $[[\pi, \pi'], \pi''] = [\pi, [\pi', \pi'']]$), we conveniently omit the extra brackets and simply write $[\pi_1, \dots, \pi_m]$ when concatenating more than two paths together. Also, since open halfedges can be interpreted as paths of length one, we extend the notation to concatenate paths and halfedges, leading to expressions such as $[\pi_1, (e, \top), \overline{\pi_2}, (e, \bot)]$. If $v_{\text{start}}(\pi_1) = v_{\text{start}}(e)$, this expression can subsequently be implicitly interpreted as a cycle, so we would simply write $\gamma = [\pi_1, (e, \top), \overline{\pi_2}, (e, \bot)]$ instead of the painful to read formal expression $\gamma = [[[[\pi_1, [(e, \top)]], \overline{\pi_2}], [(e, \bot)]]]$.

In pseudocode, we will often use the wording "Append h to π", which means "$\pi \leftarrow [\pi, h]$". Finally, we note that as per the definitions, concatenating with a trivial path is a no-operation. For instance, if π_2 is trivial, then $[\pi_1, \pi_2, \pi_3] = [\pi_1, \pi_3]$. In other words, all trivial paths are neutral elements for the concatenation operation.

6.5 Rotating non-simple cycles

Intuitively, we want a cycle to represent a loop travelling along edges with a given orientation but without a given starting point. However, for clarity of the exposition, we have formally defined a non-simple cycle as a sequence of halfedges, which means that even though we do not "need it", a "first halfedge" has to be arbitrarily chosen, specifying somehow an unwanted starting point. A negative consequence of this superfluous information is that if $h_1 \neq h_2$, then the two cycles $\gamma_1 = [h_1, h_2]$ and $\gamma_2 = [h_2, h_1]$ are two different mathematical objects (i.e., $\gamma_1 \neq \gamma_2$), even though they intuitively "represent the same cycle". To capture this intuitive notion, we define the following equivalence relation between non-simple cycles:

$$\gamma_1^+ \sim \gamma_2^+ \quad \Leftrightarrow \quad \exists d \in \mathbb{N},\ \forall j \in [1..N],\ h_j(\gamma_1^+) = h_{j-d}(\gamma_2^+) \tag{63}$$

In other words, two non-simple cycles γ_1^+ and γ_2^+ are ***equivalent*** if and only if γ_2^+ can be obtained from γ_1^+ by choosing a different "starting point", formally done via an operation called a ***rotation***, defined by:

$$\begin{array}{rccl} \text{Rot}_d : & \Gamma^+ & \to & \Gamma^+ \\ & \gamma^+ = [h_1, \dots, h_N] & \mapsto & \text{Rot}_d(\gamma^+) = [h_{(1+d) \bmod N}, \dots, h_{(N+d) \bmod N}] \end{array} \tag{64}$$

Using this operation, the equivalence relation can be rewritten as:

$$\gamma_1^+ \sim \gamma_2^+ \quad \Leftrightarrow \quad \exists d \in \mathbb{N},\ \gamma_2^+ = \text{Rot}_d(\gamma_1^+) \tag{65}$$

We extend the equivalence relation to all types of cycles by defining:

- Two Steiner cycles $\gamma_1^\bullet$ and $\gamma_2^\bullet$ are equivalent iff $v(\gamma_1^\bullet) = v(\gamma_2^\bullet)$.
- Two simple cycles γ_1° and γ_2° are equivalent iff $e^\circ(\gamma_1^\circ) = e^\circ(\gamma_2^\circ)$, $\beta(\gamma_1^\circ) = \beta(\gamma_2^\circ)$, and $N(\gamma_1^\circ) = N(\gamma_2^\circ)$
- Two cycles γ_1 and γ_2 of different nature (i.e., non-simple, Steiner or simple) are not equivalent.

We note that while a more carefully crafted definition of cycles would avoid the need for such an equivalence relation, we would lose a lot of clarity and the convenience of referring to a halfedge via its index, reason why we did not go that way. In addition, this simpler definition is closer to our actual implementation and hence has a practical value. Finally, we also note that using a circular linked list instead of an indexed sequence does *not* avoid the theoretical and practical need for an equivalence relation, since the circular linked list must still arbitrarily point to one element of the list, and hence testing for "logical equality" between two circular linked lists also requires to take into account rotations, in this case simply achieved by pointing to a different element in the list.

6.6 Extracting subpaths from paths and non-simple cycles

Given a path $\pi = (v_{\text{start}}, [h_1, \ldots, h_N], v_{\text{end}})$ and two indices j_{start} and j_{end} satisfying $0 \leq j_{\text{start}} \leq j_{\text{end}} \leq N$, we define the ***subpath*** $\pi' = \pi[j_{\text{start}}; j_{\text{end}}]$ by:

$$\pi[j_{\text{start}}; j_{\text{end}}] = \begin{cases} [v_{\text{end}}] & \text{if } j_{\text{end}} = 0 \\ [v_{\text{end}}(h_{j_{\text{end}}})] & \text{if } j_{\text{end}} > 0 \text{ and } j_{\text{start}} = j_{\text{end}} \\ [h_{j_{\text{start}}+1}, \ldots, h_{j_{\text{end}}}] & \text{otherwise (i.e., if } j_{\text{start}} < j_{\text{end}}) \end{cases} \tag{66}$$

Given a non-simple cycle $\gamma = [h_1, \ldots, h_N]$ and two indices j_{start} and j_{end}, we define the ***subpath*** $\pi' = \gamma[j_{\text{start}}; j_{\text{end}}]$ by:

$$\gamma[j_{\text{start}}; j_{\text{end}}] = \begin{cases} [v_{\text{end}}(h_{\overline{j_{\text{end}}}})] & \text{if } \overline{j_{\text{start}}} = \overline{j_{\text{end}}} \\ [h_{\overline{j_{\text{start}}+1}}, \ldots, h_{\overline{j_{\text{end}}}}] & \text{if } \overline{j_{\text{start}}+1} \leq \overline{j_{\text{end}}} \\ [h_{\overline{j_{\text{start}}+1}}, \ldots, h_N, h_1, \ldots, h_{\overline{j_{\text{end}}}}] & \text{if } \overline{j_{\text{start}}+1} > \overline{j_{\text{end}}} \end{cases} \tag{67}$$

where $\overline{j} = j \bmod N$. The above formal definition can be implemented with the following pseudocode:

SubPath($\gamma \in \Gamma^+$, $j_{\text{start}} \in \mathbb{N}$, $j_{\text{end}} \in \mathbb{N}$)

```
1 π ← [v_{j_start}(γ)]
2 j ← j_start
3 while j ≢ j_end (mod N(γ)) do
4     j ← j + 1
5     Append h_j(γ) to π
6 return π
```

7 Topological operators on abstract PCS complexes

In this section, we present topological operators that transform a valid abstract PCS complex into another valid abstract PCS complex, given a relevant input. These operators apply to the VGC as well: just ignore all genuses and orientabilities. This is possible since, except in two exceptional cases, genuses and orientabilities are never used to determine what actions to take: they are only used to compute other genuses and orientabilities. The two exceptional cases are CutNonOrientableFaceAtNonDisconnectingOrientingClosedEdge() and CutNonOrientableFaceAtNonDisconnectingOrientingOpenEdge(). In these cases, the "if" branching can be seen as two alternatives that are both valid. Since the input always include a valid abstract PCS complex, and that the output is always a valid abstract PCS complex, we do not mention them, and instead we assume that we are working on a globally accessible abstract PCS complex $\mathcal{P} = (C, \dim, \epsilon, g, \widehat{\partial})$ that is modified in-place by the topological operator.

7.1 Cell creation

Creating a cell means adding to C a new symbol c that is not already contained in C, and defining the value of $\dim(c)$ and $\widehat{\partial}c$ for this new symbol. If $\dim(c) = 2$, then we need to define $\epsilon(c)$ and $g(c)$ as well. For vertices and edges, the corresponding topological operators are:

CreateVertex()

```
1 Let v ∉ C                          ▷ Memory allocation in real-life code, cf. next paragraph
2 dim(v) ← 0
3 ∂̂v ← ∅
4 Insert v in C
5 return v
```

CreateClosedEdge()

1 Let $e^\circ \notin C$
2 $\dim(e^\circ) \leftarrow 1$
3 $\widehat{\partial} e^\circ \leftarrow \varnothing$
4 Insert e° in C
5 **return** e°

CreateOpenEdge($v_{\text{start}} \in V$, $v_{\text{end}} \in V$)

1 Let $e \notin C$
2 $\dim(e) \leftarrow 1$
3 $\widehat{\partial} e \leftarrow (v_{\text{start}}, v_{\text{end}})$
4 Insert e in C
5 **return** e

In real-life code, "finding a new symbol not already in C" is typically a memory allocation. For instance, in C++, if C is defined as `set<Cell *> C;`, then "Let $c \notin C$" translates to `Cell * c = new Cell;`. Also, defining the value $\dim(c)$ may translate to "do nothing" when implemented with an object-oriented language where the type of c already tells you if it's a vertex, an edge, or a face. We clarify this with a C++ snippet:

```
class Cell
{
    virtual int dimension() const=0;
    virtual set<Cell*> boundary() const=0;
};
class Vertex: public Cell
{
    int dimension() const { return 0; }
    set<Cell*> boundary() const { return set<Cell*>; }
};
class Edge: public Cell
{
    Vertex * start, end; // Both NULL if closed edge

    Edge() : start(NULL), end(NULL) {}                      // Create closed edge
    Edge(Vertex * vs, Vertex * ve) : start(vs), end(ve) {} // Create open edge

    int dimension() const { return 1; }
    set<Cell*> boundary() const
    {
        set<Cell*> res;
        if(start == NULL) // Closed edge
        {
             return res;
        }
        else              // Open edge
        {
            res.insert(start);
            res.insert(end);
            return res;
        }
    }
};
```

And then the method CreateClosedEdge() would simply be

```
Edge * createClosedEdge()
{
    Edge * e = new Edge();
    C.insert(e);
    return e;
}
```

Finally, the most atomic way to create a face is in several steps: one step to create a face without boundary, and then one step per cycle to add. Cycles can also be removed afterwards:

CreateFace($\epsilon_f \in \{\circlearrowleft, \not\circlearrowleft\}$, $g_f \in \mathbb{N}$)

```
1 Let f ∉ C
2 dim(f) ← 2
3 ∂̂f ← [ ]                                   ▷ Empty sequence of cycles
4 ϵ(f) ← ϵ_f
5 g(f) ← g_f
6 Insert f in C
7 return f
```

AddSteinerCycleToFace($f \in F$, $v \in V$)

```
1 Append γ• = [v] to ∂̂f
```

AddSimpleCycleToFace($f \in F$, $e^\circ \in E_\circ$, $\beta \in \{\top, \bot\}$, $N \in \mathbb{N}$)

```
1 Append γ° = [(e°, β)^N] to ∂̂f
```

AddNonSimpleCycleToFace($f \in F$, $\gamma \in \Gamma^+$)

```
1 Append γ to ∂̂f
```

AddCycleToFace($f \in F$, $\gamma \in \Gamma$)

```
1 Append γ to ∂̂f
```

RemoveCyclesFromFace($f \in F$, $I \subset \mathbb{N}$)

```
1 Δ ← [ ]
2 for all i ∈ [1..k(f)], i ∉ I do
3     Append γ_i(f) to Δ
4 ∂̂f ← Δ
```

RemoveCycleFromFace($f \in F$, $i \in \mathbb{N}$)

```
1 RemoveCyclesFromFace(f,{i})
```

7.2 Cell deletion

In the general case, deleting a cell by simply removing it from C would result in the general case in an invalid abstract PCS complex. For instance, if $C = \{v_1, v_2, e\}$ with $\widehat{\partial} e = (v_1, v_2)$, them removing v_1 would result in $C = \{v_2, e\}$ with $\widehat{\partial} e = (v_1, v_2)$, which is clearly an invalid abstract PCS complex since we must have $v_{start}(e) \in V$ which is not anymore the case. More generally, removing a cell c from C is valid if and only if we have $\mathrm{star}(c) = \varnothing$. Hence, one possible way to define "deletion" is to remove altogether of c and $star(c)$, operation that we call "hard delete", and that is safely achieved by the following recursive method:

HardDelete($c \in C$)

```
1 while ∃c' ∈ star(c) do
2     HardDelete(c')
3 Remove c from C                  ▷ In real-life code, remove from set, then release memory
```

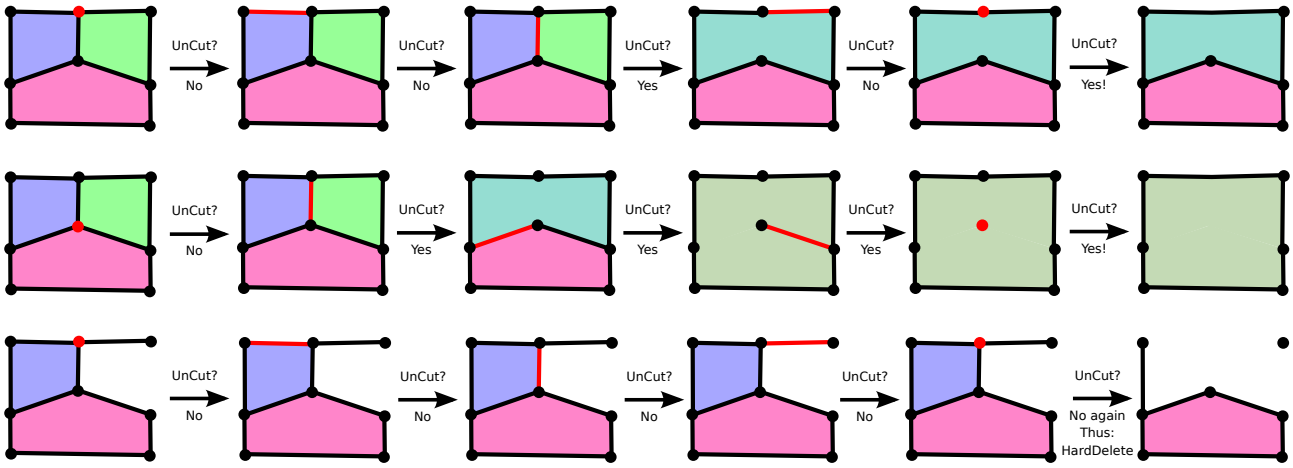


Figure 26: Three scenarios using SmartDelete().

But there is a less destructive way to remove c from C: perform an *atomic simplification* at c, as defined in Section 9 and illustrated in Figure 34. This operation is equivalent to the UnCut() topological operator defined in Section 7.6. However, not all cells are candidate for atomic simplification. So we may think of a method "if can be atomically simplified, atomically simplify; otherwise, hard delete". But this approach is still too destructive: in the first two scenarios in Figure 26, it would be equivalent to hard delete, while we can see that a less destructive approach exists. This approach is "if can be simplified, simplify; otherwise, hard delete", where simplifying a cell c corresponds to recursively simplify all its star cells first, then atomically simplify c, if possible. This "smart delete" operation is implemented by the following topological operators:

SmartDelete($c \in C$)

```
1 if c ∈ V then
2     SmartDeleteVertex(c)
3 else if c ∈ E then
4     SmartDeleteEdge(c)
5 else if c ∈ F then
6     SmartDeleteFace(c)
```

SmartDeleteFace($f \in F$)

```
1 HardDelete(f)
```

SmartDeleteEdge($e \in E$)

```
1 if CanUncutAtEdge(e) then
2     UnCutAtEdge(e)
3 else
4     HardDelete(e)
```

SmartDeleteVertex($v \in V$)

```
1 if CanUncutAtVertex(v) then
2     UnCutAtVertex(v)
3 else
4     for all Edge e ∈ star(c) do
5         if CanUncutAtEdge(e) then UnCutAtEdge(e)
6     if CanUncutAtVertex(v) then
7         UnCutAtVertex(v)
8     else
9         HardDelete(v)
```

7.3 Glue

Gluing is a rather simple topological operator, both conceptually and to implement. To glue two cells c_1 and c_2 of same "type", the idea is to create a new cell c (in case some geometry is associated to the topololy, the geometry of c would be averaging the ones of c_1 and c_2), then to replace every occurrence of c_1 or c_2 (in the semantic boundary of other cells) by c, and delete c_1 and c_2 (note that because c_1 or c_2 do not belong anymore to the boundary of any cell, we have $\mathrm{star}(c_1) = \mathrm{star}(c_2) = \varnothing$, thus deleting them simply means removing them from C). Hence, to glue two vertices v_1 or v_2, you should replace every occurrence of v_1 or v_2 (as a start vertex, end vertex, or Steiner cycle) by the new "glued" vertex v.

GlueVertices($v_1 \in V$, $v_2 \in V$**)**

Require: $v_1 \neq v_2$
$v \leftarrow$ CreateVertex()
for all open edge $e \in \mathrm{star}(v_1) \cup \mathrm{star}(v_2)$ **do**
  if $v_{\mathrm{start}}(e) = v_1$ OR $v_{\mathrm{start}}(e) = v_2$ **then**
    $v_{\mathrm{start}}(e) \leftarrow v$
  if $v_{\mathrm{end}}(e) = v_1$ OR $v_{\mathrm{end}}(e) = v_2$ **then**
    $v_{\mathrm{end}}(e) \leftarrow v$
for all face $f \in \mathrm{star}(v_1) \cup \mathrm{star}(v_2)$ **do**
  for all Steiner cycle $\gamma_i^\bullet \in \widehat{\partial} f$ **do**
    if $\gamma_i^\bullet = [v_1]$ OR $\gamma_i^\bullet = [v_2]$ **then**
      $\gamma_i^\bullet \leftarrow [v]$
HardDelete(v_1)
HardDelete(v_2)
return v

Gluing two edges is ambiguous: one needs to decide on a chosen relative orientation first. If there is geometry available, simple heuristics should be enough (for instance using the sign of a dot product). Once orientation is decided, we are left to glue two halfedges (e_1, β_1) and (e_2, β_2). To achieve this, we first glue their start vertices and end vertices together (if any), then we create a new edge e, and replace every occurrence of $(e_1, \top)$, $(e_1, \bot)$, $(e_2, \top)$, or $(e_2, \bot)$ by either $(e, \top)$ or $(e, \bot)$.

GlueClosedHalfedges($(e_1^\circ, \beta_1) \in H_\circ$, $(e_2^\circ, \beta_2) \in H_\circ$**)**

Require: $e_1^\circ \neq e_2^\circ$
$e^\circ \leftarrow$ CreateClosedEdge()
for all face $f \in \mathrm{star}(e_1^\circ) \cup \mathrm{star}(e_2^\circ)$ **do**
  for all simple cycle $\gamma_i^\circ = [(e_i^\circ, \beta_i)^{N_i}] \in \widehat{\partial} f$ **do**
    if $e_i^\circ = e_1^\circ$ **then**
      $\gamma_i^\circ \leftarrow [(e^\circ, (\beta_i \Leftrightarrow \beta_1))^{N_i}]$ ▷ "$\beta \Leftrightarrow \beta'$" returns $\top$ if $\beta = \beta'$, $\bot$ otherwise
    else if $e_i^\circ = e_2^\circ$ **then**
      $\gamma_i^\circ \leftarrow [(e^\circ, (\beta_i \Leftrightarrow \beta_2))^{N_i}]$
HardDelete(e_1°)
HardDelete(e_2°)
return e°

GlueOpenHalfedges($h_1 = (e_1, \beta_1) \in H_|$, $h_2 = (e_2, \beta_2) \in H_|$)

Require: $e_1 \neq e_2$
if $v_{\text{start}}(h_1) = v_{\text{start}}(h_2)$ **then**
  $v_{\text{start}} \leftarrow v_{\text{start}}(h_1)$
else
  $v_{\text{start}} \leftarrow \text{GlueVertices}(v_{\text{start}}(h_1), v_{\text{start}}(h_2))$
if $v_{\text{end}}(h_1) = v_{\text{end}}(h_2)$ **then**
  $v_{\text{end}} \leftarrow v_{\text{end}}(h_1)$
else
  $v_{\text{end}} \leftarrow \text{GlueVertices}(v_{\text{end}}(h_1), v_{\text{end}}(h_2))$
$e \leftarrow \text{CreateOpenEdge}(v_{\text{start}}, v_{\text{end}})$
for all face $f \in \text{star}(e_1) \cup \text{star}(e_2)$ **do**
  for all non-simple cycle $\gamma_i = [h_1, \ldots, h_{N_i}] \in \widehat{\partial} f$ **do**
    for all halfedge $h_j = (e_j, \beta_j) \in \gamma_i$ **do**
      if $e_j = e_1$ **then**
        $h_j \leftarrow (e, (\beta_j \Leftrightarrow \beta_1))$
      else if $e_j = e_2$ **then**
        $h_j \leftarrow (e, (\beta_j \Leftrightarrow \beta_2))$
HardDelete(e_1)
HardDelete(e_2)
return e

7.4 UnGlue

Informally, *unglue* is the "reverse" topological operation of *glue*. However, this is not always completely true, as creating a vertex shared by three edges requires two glue operations, that can be reversed with a single unglue. We illustrate this with the few examples below:

$$\left\{ \begin{array}{c} v_1 \\ v_2 \end{array} \right\} \xrightarrow{\text{Glue}(v_1,v_2)} \left\{ \begin{array}{c} v \end{array} \right\} \xrightarrow{\text{UnGlueAt}(v)} \left\{ \begin{array}{c} v \end{array} \right\}$$

$$\left\{ \begin{array}{c} v_1 \\ v_2 \\ v_1' \\ v_2' \\ e_1 = (v_1, v_1') \\ e_2 = (v_2, v_2') \end{array} \right\} \xrightarrow{\text{Glue}(v_1,v_2)} \left\{ \begin{array}{c} v \\ v_1' \\ v_2' \\ e_1 = (v, v_1') \\ e_2 = (v, v_2') \end{array} \right\} \xrightarrow{\text{UnGlueAt}(v)} \left\{ \begin{array}{c} v_1 \\ v_2 \\ v_1' \\ v_2' \\ e_1 = (v_1, v_1') \\ e_2 = (v_2, v_2') \end{array} \right\}$$

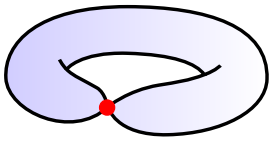
$$\left\{ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_1' \\ v_2' \\ v_3' \\ e_1 = (v_1, v_1') \\ e_2 = (v_2, v_2') \\ e_3 = (v_3, v_3') \end{array} \right\} \xrightarrow{\text{Glue}(v_1,v_2)} \left\{ \begin{array}{c} v \\ v_3 \\ v_1' \\ v_2' \\ v_3' \\ e_1 = (v, v_1') \\ e_2 = (v, v_2') \\ e_3 = (v_3, v_3') \end{array} \right\} \xrightarrow{\text{Glue}(v,v_3)} \left\{ \begin{array}{c} v' \\ v_1' \\ v_2' \\ v_3' \\ e_1 = (v', v_1') \\ e_2 = (v', v_2') \\ e_3 = (v', v_3') \end{array} \right\} \xrightarrow{\text{UnGlueAt}(v')} \left\{ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_1' \\ v_2' \\ v_3' \\ e_1 = (v_1, v_1') \\ e_2 = (v_2, v_2') \\ e_3 = (v_3, v_3') \end{array} \right\}$$

Fundamentally, UnGlue(c) duplicates c as many times as it is "used" by cells of higher dimension, or do nothing if $\text{star}(c) = \varnothing$. We formalize now the notion of "use", which is similar to the vertex-use, edge-use and face-use in the *radial-edge* data structure [Weiler, 1985], but not exactly identical. The fundamental difference is that while in the radial-edge data structure, these uses are *explicit objects* (for instance, vertex-uses are ordered in a cyclic doubly-linked list around the vertex they represent), they are only *implicit* in abstract PCS complexes.

Another less significant difference is that the radial-edge data structure does not support Steiner cycles and closed edges, but we believe it could be easily extended to support them. Finally, since the radial-edge data structure is designed to represent solid 3D objects, it also defines volumes via *shells* (shells are for volumes what cycles are for surfaces) and hence defines *face-uses*, while we stop at the dimension 2 and hence do not need them.

Vertex-use A vertex v can be used in three different ways:

- As a start or end vertex of an open edge e that has no incident face (i.e., $\widehat{\partial} e = \varnothing$). Such a use is called ***end-vertex-use*** and denoted $\textcircled{v}_{e,\beta}$ with $\beta \in \{\text{start}, \text{end}\}$. Note that an open edge e uses twice the same vertex if it has no incident faces and $v_{\text{start}}(e) = v_{\text{end}}(e)$. Note also that if e has incident faces, then it is not considered as using any of its end vertices (otherwise redundant with *corner-vertex-uses*).
- As a Steiner cycle $\gamma_i^\bullet$ of a face f. Such a use is called ***Steiner-vertex-use*** and denoted $\textcircled{v}_{f,i}$. Note that the same vertex can be used as Steiner more than once by the same face. For instance, consider the "pinched torus" made of one vertex v and one face f such that $\widehat{\partial} f = [[v], [v]]$.



- As the vertex $v_j(\gamma_i)$, junction between the consecutive halfedges h_j and h_{j+1} in a non-simple cycle γ_i of a face f. Such a use is called ***corner-vertex-use*** and denoted $\textcircled{v}_{f,i,j}$

Open-edge-use An open edge e can only be used in one way: as an open halfedge h_j in a non-simple cycle γ_i of a face f. Such a use is called ***open-edge-use*** and denoted $\textcircled{e}_{f,i,j}$.

Closed-edge-use A closed edge e° can only be used in one way: as a simple cycle γ_i° of a face f. If N_i (i.e., $N(\gamma_i^\circ)$) is greater than one, then the closed edge e° is considered to be used as many times by the face. Such a use is called ***closed-edge-use*** and denoted $\textcircled{\overset{\circ}{e}}_{f,i,j}$, where j allows to distinguish repeated uses in the same simple cycle. An example where $N_i = 2$ is the "cut-Möbius" illustrated in Figure 27, bottom-middle.

Once this notion of *use* is defined, the unglue topological operators are conceptually simple: to unglue at a cell c, you create a new cell c_k for each use $\textcircled{c}_k$ of c, and replace c by c_k for this specific use. After this operation, c is not used anymore and hence we delete it. This scheme is well illustrated by the implementation of UnGlueAtOpenEdge:

UnGlueAtOpenEdge($e \in E_|$)

```
1 if star(e) = ∅ then
2     Do nothing.
3 else
4     for all face f ∈ star(e) do
5         for all non-simple cycle γ_i ∈ ∂̂f do
6             for all halfedge h_j ∈ γ_i do
7                 if e(h_j) = e then                    ▷ Found open-edge-use ⓔ_{f,i,j}
8                     e(h_j) ← CreateOpenEdge(v_start(e), v_end(e))
9     HardDelete(e)
```

The case of closed edges is as easy to implement, but conceptually challenging. If $N_i = 1$ for all simple cycles γ_i° using e°, then there are no difficulties. However, the case $N_i > 1$ is not as straightforward. To understand what the algorithm should do in this case, let us clarify what a "repeated closed edge" represents. Consider a Möbius strip represented by its minimal PCS decomposition (Figure 27, bottom-left):

- one closed edge e_1°: its unique boundary edge
- one face $f = (\emptyset, 1, [\ [(e_1^\circ, \top)]\ ])$: non-orientable, genus-1, one simple cycle

The closed edge e_1° is only used once by f. Indeed, if we arbitrarily choose an orientation for this closed curve, we can see that locally, f is only "at the left side" of e_1°, or "at the right side", but not on both sides. It is well-known that if you take scissors and cut this Möbius strip in half along its length, you obtain a single

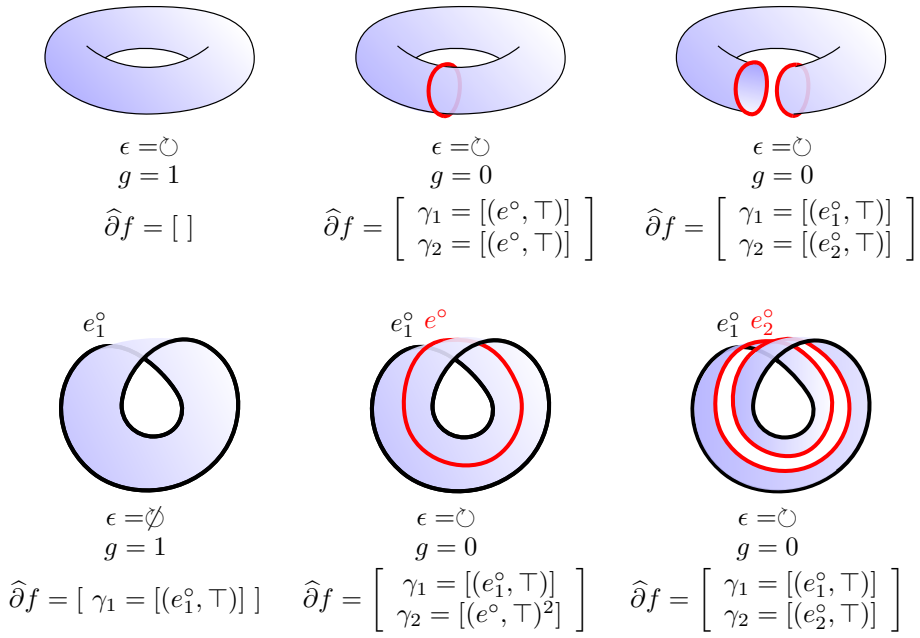
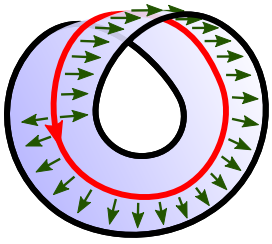


Figure 27: The "cut-torus" (top row, middle column) and "cut-Möbius" (bottom row, middle column) are two examples of abstract PCS complexes where a closed edge e° is used twice by the same face. In the case of the cut-torus, the two closed-edge-uses of e° come from two simple cycles, while in the case of the cut-Möbius, the two closed-edge-uses of e° come from a single cycle repeating e° twice. We show these examples before the cut (left), then after the cut (middle), then after ungluing at the cut edge e° (right). Ungluing the cut-torus or the cut-Möbius at e° gives the same abstract PCS complex: the cylinder. It has no vertices, two closed edges e_1° and e_2°, and a face f such that $\widehat{\partial} f = (\circlearrowleft, 0, [\ [h_1^\circ]; [h_2^\circ]\ ])$. Indeed, the two surfaces depicted in the right column are both homeomorphic to $\mathbb{F}_{\circlearrowleft,0,2}$.

orientable surface [1] (Figure 27, bottom-right). In terms of abstract PCS complexes, this operation can be decomposed into two atomic topological operators:

1. The first topological operator is CutNonOrientableFaceAtNonDisconnectingOrientingClosedEdge(f) (see Section 7.5.6), and corresponds to "tracing" the red closed edge e° along the middle of the Möbius strip (Figure 27, bottom-middle). In terms of PCS complexes, this corresponds to partition f into two cells: e° and $f \backslash e^\circ$. With this new (not minimal) decomposition of the Möbius strip, the cell f is now homeomorphic to $\text{int}(\mathbb{F}_{\circlearrowleft,0,2})$, while it was homeomorphic to $\text{int}(\mathbb{F}_{\not\circlearrowleft,1,1})$ before the cut. However, this does not change the whole topological space X that the PCS complex represents (i.e., the union of cells), which is still homeomorphic to the Möbius strip $\mathbb{F}_{\not\circlearrowleft,1,1}$ ("Cutting" is simply decomposing the same space into more cells as will be discussed later).

   After this cut, we can observe that if we arbitrarily choose an orientation for this closed curve e°, then f is actually "both at the left side and the right side" of e°. This explains why f actually uses e° twice. But unlike the "cut-torus" (Figure 27, top-middle), these two uses are from the *same* cycle. To understand why, pick a point on e°, then pick one side of the face (for instance, the "left side"). If you move along e° while keeping in mind which side of f you picked, then after one turn you will realize that *you end up at the other side of* e°. Hence, you have to perform two complete turns around e° to actually complete the cycle, that continuously goes through the two closed-edges-uses.

2. The second topological operator is UnglueAtClosedEdge(e°), that actually changes the topological space

[1] If you have never done this experiment before, we strongly recommend that you physically verify this by yourself. Take a rectangular piece of paper, and construct a Möbius strip by gluing the two small edges together with a half twist. Then, cut it along its length with scissors. Verify that it is two-sided by coloring its sides with different colors.

X by "unconnecting" the two closed-edges-uses of e°. Similarly to the cut-torus example (Figure 27, top), this is achieved by "duplicating the geometry" of e°. However, unlike the cut-torus example where this duplicated geometry is distributed among two closed edges e_1° and e_2°, the duplicated geometry belongs to the same closed edge e_2°, making the closed edge twice as long as it was initially (cf. Figure 27, bottom-right). Combinatorially, this duplication of geometry is conceptual, and it simply means that the simple cycle γ_i° that uses multiple times e° is transformed into a simple cycle γ_i° that uses only once a new closed edge.

Note that it is also possible to have $N_i \geq 3$. For instance, take three rectangles glued together along a long edge, then glue their short edges in the same way you would construct a Möbius strip, but with a third-twist instead of a half-twist. With this understanding, we can finally define the topological operator UnglueAtClosedEdge(e°): for every simple cycle $\gamma_i^\circ = [(e_i^\circ, \beta_i)^{N_i}]$ that uses e° (possibly $N_i > 1$ times), we create a new closed edge e_i° and change γ_i° into $[(e_i^\circ, \beta_i)]$.

UnGlueAtClosedEdge($e^\circ \in E_\circ$)

```
1 if star(e°) = ∅ then
2     Do nothing.
3 else
4     for all face f ∈ star(e°) do
5         for all simple cycle γ_i° ∈ ∂̂f do
6             if e°(γ_i°) = e° then                ▷ Found closed-edge-uses (e°)_{f,i,1} to (e°)_{f,i,N(γ_i°)}
7                 e°(γ_i°) ← CreateClosedEdge()
8                 N(γ_i°) ← 1
9     HardDelete(e°)
```

Finally, to unglue at a vertex, the important difference is that for this operation to be valid, all cells in the star of v must be unglued first. Then, we handle independently the different use cases.

UnGlueAtVertex($v \in V$)

```
1 if star(v) = ∅ then
2     Do nothing.
3 else
4     for all edge e ∈ star(v) do                  ▷ Unglue at all (necessarily open) star edges of v
5         UnGlueAtOpenEdge(e)
6     for all edge e ∈ star(v) do
7         if star(e) = ∅ then
8             if v_start(e) = v then               ▷ End-vertex-use (v)_{e,start}
9                 v_start(e) ← CreateVertex()
10            if v_end(e) = v then                 ▷ End-vertex-use (v)_{e,end}
11                v_end(e) ← CreateVertex()
12    for all face f ∈ star(v) do
13        for all Steiner cycle γ_i• = [v_i] ∈ ∂̂f do
14            if v_i = v then                      ▷ Steiner-vertex-use (v)_{f,i}
15                v(γ_i•(f)) ← CreateVertex()
16        for all non-simple cycle γ_i ∈ ∂̂f do
17            for all halfedges h_j ∈ γ_i do
18                if v_end(h_j) = v then           ▷ Corner-vertex-use (v)_{f,i,j}
19                    v_{f,i,j} ← CreateVertex()
20                    v_end(h_j) ← v_{f,i,j}
21                    v_start(h_{j+1}) ← v_{f,i,j}
22    HardDelete(v)
```

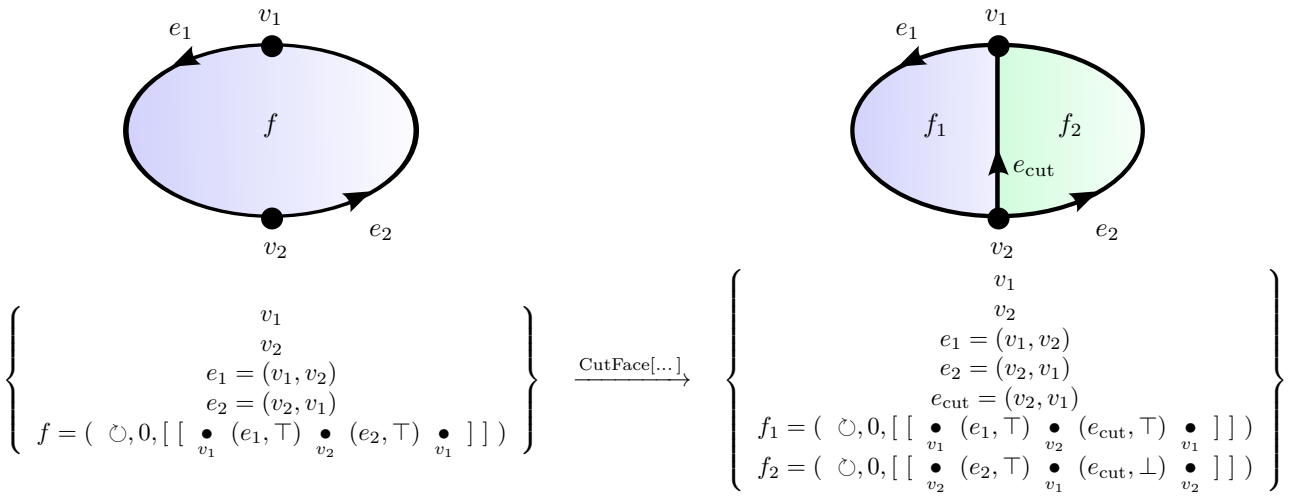


Figure 28: An abstract PCS complex is transformed into another abstract PCS complex, as a result of the cut topological operator CutOrientableFaceAtDisconnectingOpenEdge(f,1,1,2,0,0,[ ]).

7.5 Cut

Given a (non-abstract) *PCS complex*, a ***cut*** is defined as partitioning a cell c into new cells $\{c_{\text{cut}}, c_1, c_2, \dots\}$, where c_{cut} is a proper subset of c and where $\{c_1, c_2, \dots\}$ are the connected components of $c \setminus c_{\text{cut}}$. We say that "$c$ is cut at c_{cut}". A ***valid cut*** is a cut such that the resulting cell decomposition is a valid PCS complex. For instance, if a face f is cut at an open edge $e_{\text{cut}} \subset f$, then e_{cut} must start and end at vertices in ∂f. From now on, whenever we say "a cut", we mean a valid cut. It can be shown that a cut necessarily satisfies $\dim(c_{\text{cut}}) < \dim(c)$. Subsequently, it can be shown that $c \setminus c_{\text{cut}}$ is either connected or made of two connected components, thus c is partitioned into either two components $\{c_{\text{cut}}, c'\}$ or three components $\{c_{\text{cut}}, c_1, c_2\}$. This intuitive result should become clear with the different examples illustrated in this section.

The cut topological operator on *abstract PCS complex* that we describe in this section is the combinatorial counterpart of the pointset definition given above. This means that a given abstract PCS complex $\mathcal{P}$ is transformed into another abstract PCS complex $\mathcal{P}'$ such that the PCS complex $|\mathcal{P}'|$ could have been obtained by cutting a cell c of $|\mathcal{P}|$ at some subcell $c_{\text{cut}} \subset c$. Interestingly, this means that the abstract cell c_{cut} of $\mathcal{P}'$ is actually an *output* of the cut topological operator, as illustrated in Figure 28, whereas it is more easily interpreted as an *input* with the pointset definition (i.e.: "cut *here*"). Because an abstract PCS complex is a purely combinatorial object, its faces are purely abstract and not assumed to be realized as pointsets (or even as abstract triangulations), and therefore it is not possible to say "cut *here*", and we should instead say "cut *this way*". For instance, in Figure 28, it would be along the lines of "cut f at an open edge starting at v_2 and ending at v_1". However, while this sentence entirely specifies the cut in the simple example in Figure 28, things are actually much more complicated in the general case because:

- If f uses v_1 or v_2 more than once, then the sentence is ambiguous: the actual vertex-uses must be specified instead of "just" the vertices.
- If f contains several cycles, then the sentence is ambiguous: it is necessary to specify which cycles must be transferred to f_1 and which cycles must be transferred to f_2.
- If the genus of f is non-zero (e.g., a torus with a hole), then the sentence is ambiguous: a cut from v_2 to v_1 may or may not disconnect f, depending on the actual path of e_{cut} in a pointset sense, and this information has to be specified combinatorially somehow.
- Other ambiguities that are detailed later.

To exhaustively cover all the possible cases, and find out what combinatorial input is *necessary and sufficient* to fully determine the cut, we have to *classify* all the different *ways* a cell can be cut. Only then we can rigorously define "cut *this way*", and make sure that we are not missing any *way* to cut a cell. For instance, the cut topological operator performed in Figure 28 can be more accurately expressed as "cut the orientable face f

at an open edge starting at the vertex-use $\textcircled{v}_{f,1,1}$, ending at the vertex-use $\textcircled{v}_{f,1,2}$, disconnecting f into two (necessarily orientable) faces, both of genus zero, and both receiving no cycles from f". To do this, you would call the method CutOrientableFaceAtDisconnectingOpenEdge(f,1,1,2,0,0,[ ]).

We provide below an informal overview of the different cases to consider. The exhaustive list is provided by the following sections, and in particular the different ways to cut a face at an edge are illustrated in Figure 30 and 31.

- **Cutting an open edge (at a vertex)**
  An open edge e becomes a vertex v and two open edges e_1 and e_2.
- **Cutting a closed edge (at a vertex)**
  A closed edge e° becomes a vertex v and the open edge $e' = e^\circ \setminus v$.
- **Cutting a face at a vertex**
  A face f becomes a vertex v and the face $f' = f \setminus v$.
- **Cutting a face at a closed edge, disconnecting it**
  A face f becomes a closed edge e° and two faces f_1 and f_2. The holes, handles or crosscaps of f are distributed among f_1 and f_2. The cycle $[(e^\circ, \top)]$ is added to f_1 and the cycle $[(e^\circ, \bot)]$ is added to f_2.
- **Cutting a face at a closed edge, not disconnecting it**
  A face f becomes a closed edge e° and the face $f' = f \setminus e^\circ$. If f is orientable, its genus is decreased by one, and the two cycles $[(e^\circ, \top)]$ and $[(e^\circ, \bot)]$ are added. If f is non-orientable, it may become orientable or not, its genus may be decreased by one or more, and either the cycle $[(e^\circ, \top)]$ is added twice, or the single cycle $[(e^\circ, \top)^2]$ is added.
- **Cutting a face at an open edge starting/ending at the same hole, disconnecting it**
  A face f becomes an open edge e and two faces f_1 and f_2. The handles or crosscaps of f are distributed among f_1 and f_2. All cycles except one are distributed among f_1 and f_2. The last cycle $\gamma_i = [\pi_1, \pi_2]$ is split by adding $[\pi_1, (e, \top)]$ to f_1 and adding $[\pi_2, (e, \bot)]$ to f_2.
- **Cutting a face at an open edge starting/ending at the same hole, not disconnecting it**
  A face f becomes an open edge e and the face $f' = f \setminus e$. If f is orientable, its genus is decreased by one, and the cycle $\gamma_i = [\pi_1, \pi_2]$ is split into $[\pi_1, (e, \top)]$ and $[\pi_2, (e, \bot)]$. If f is non-orientable, it may become orientable or not, its genus may be decreased by one or more, and either the two cycles $[\pi_1, (e, \top)]$ and $[\overline{\pi_2}, (e, \top)]$ are added, or the single cycle $[\pi_1, (e, \top), \overline{\pi_2}, (e, \top)]$ is added.
- **Cutting a face at an open edge starting/ending at different holes**
  A face f becomes an open edge e and the face $f' = f \setminus e$. Two cycles γ_{i_1} and γ_{i_2} are merged into the single cycle $[\gamma_{i_1}, (e, \top), \gamma_{i_2}, (e, \bot)]$.

7.5.1 Cutting an open edge (at a vertex)

Cutting an open edge e is a very simple operation that consists in splitting e in half by inserting a vertex in its interior, resulting in two open edges e_1 and e_2 and one vertex v. One only has to take care of replacing the halfedges using e by two halfedges using respectively e_1 and e_2 with the appropriate orientation. The three words "at a vertex" are in parenthesis in the title of the section and not part of the topological operator name because cutting an open edge is necessarily done at a vertex due to the requirement $\dim(c_{\text{cut}}) < \dim(c)$.

CutOpenEdge($e \in E_|$)

```
1  v ← CreateVertex()
2  e_1 ← CreateOpenEdge(v_start(e), v)
3  e_2 ← CreateOpenEdge(v, v_end(e))
4  for all face f ∈ star(e) do
5      for all non-simple cycle γ_i ∈ ∂̂f do
6          γ'_i ← [ ]
7          for all halfedges h_j ∈ γ_i do
8              if e(h_j) = e then
9                  if β(h_j) = ⊤ then
10                     Append (e_1, ⊤) to γ'_i
11                     Append (e_2, ⊤) to γ'_i
12                 else
13                     Append (e_2, ⊥) to γ'_i
14                     Append (e_1, ⊥) to γ'_i
15             else
16                 Append h_j to γ'_i
17         γ_i(f) ← γ'_i
18 HardDelete(e)
```

7.5.2 Cutting a closed edge (at a vertex)

Cutting a closed edge e° follows the same idea, and it results in one open edge e and one vertex v.

CutClosedEdge($e^\circ \in E_\circ$)

```
1  v ← CreateVertex()
2  e ← CreateOpenEdge(v, v)
3  for all face f ∈ star(e°) do
4      for all simple cycle γ°_i = [(e°_i, β_i)^{N_i}] ∈ ∂̂f do
5          if e°_i = e° then
6              γ_i(f) ← [ •_v (e, β_i) •_v ··· •_v (e, β_i) •_v ]   ▷ Replace the simple cycle by a non-simple cycle
                                                                      that repeats N_i times the open edge e
7  HardDelete(e°)
```

7.5.3 Cutting a face at a vertex

A trivial way to cut a face is via a vertex. This means that we decompose the face f into a new vertex v and the new face $f \setminus v$.

CutFaceAtVertex($f \in F$)

```
1  v ← CreateVertex()
2  AddSteinerCycleToFace(f,v)
```

7.5.4 Cutting a face at an edge

Cutting a face at an open or closed edge is a much harder operation because there are many non-trivial and non-equivalent ways a face can be cut (cf. Figure 29): the face may become disconnected or not, its orientability and genus may change or not, and different cycles may be added, split, or merged together. If the abstract PCS complex is realized as a triangulation, and the cut edge is given as a subset of the edges in the triangulation,

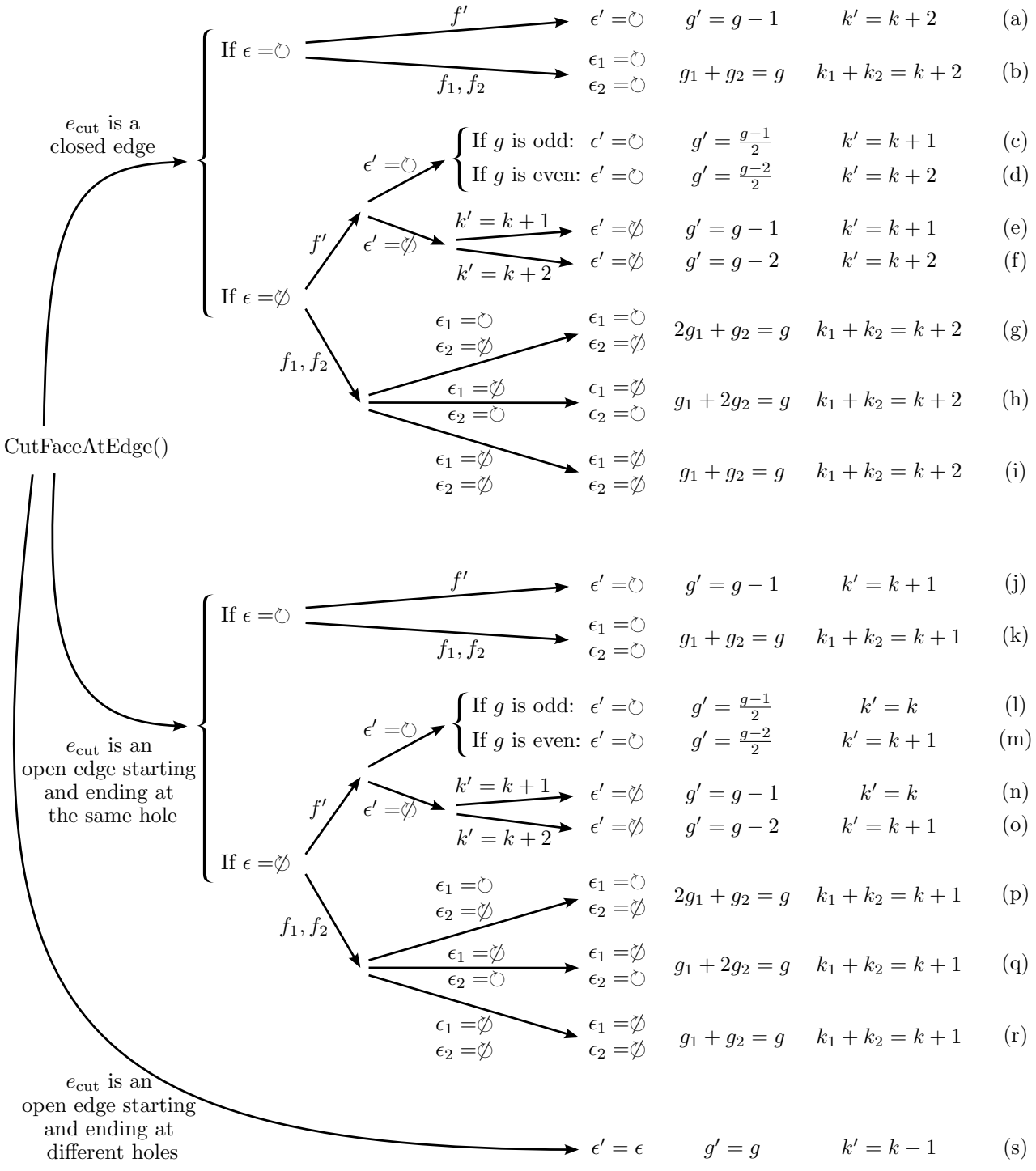


Figure 29: Exhaustive classification of the 19 different ways a face can be cut at an edge. The branching "if"s represent known information about the abstract PCS complex that is about to be cut. The branching arrows represent information about e_{cut} that cannot be algorithmically determined, and hence that has to be given as input of the PCS topological operator (either as parameters or by calling different methods). We only show here the unknown information that leads to different orientabilities, genus formulae, number of faces, or number of cycles (e.g.: is e_{cut} closed or not? Does e_{cut} disconnect the face?). Additional parameters to give to the topological operators include: if e_{cut} disconnects f, which cycles to transfer to f_1 or f_2? What are the new genuses g_1 and g_2? If e_{cut} is open, at which vertex-uses does it start and end? Should some cycles be flipped?

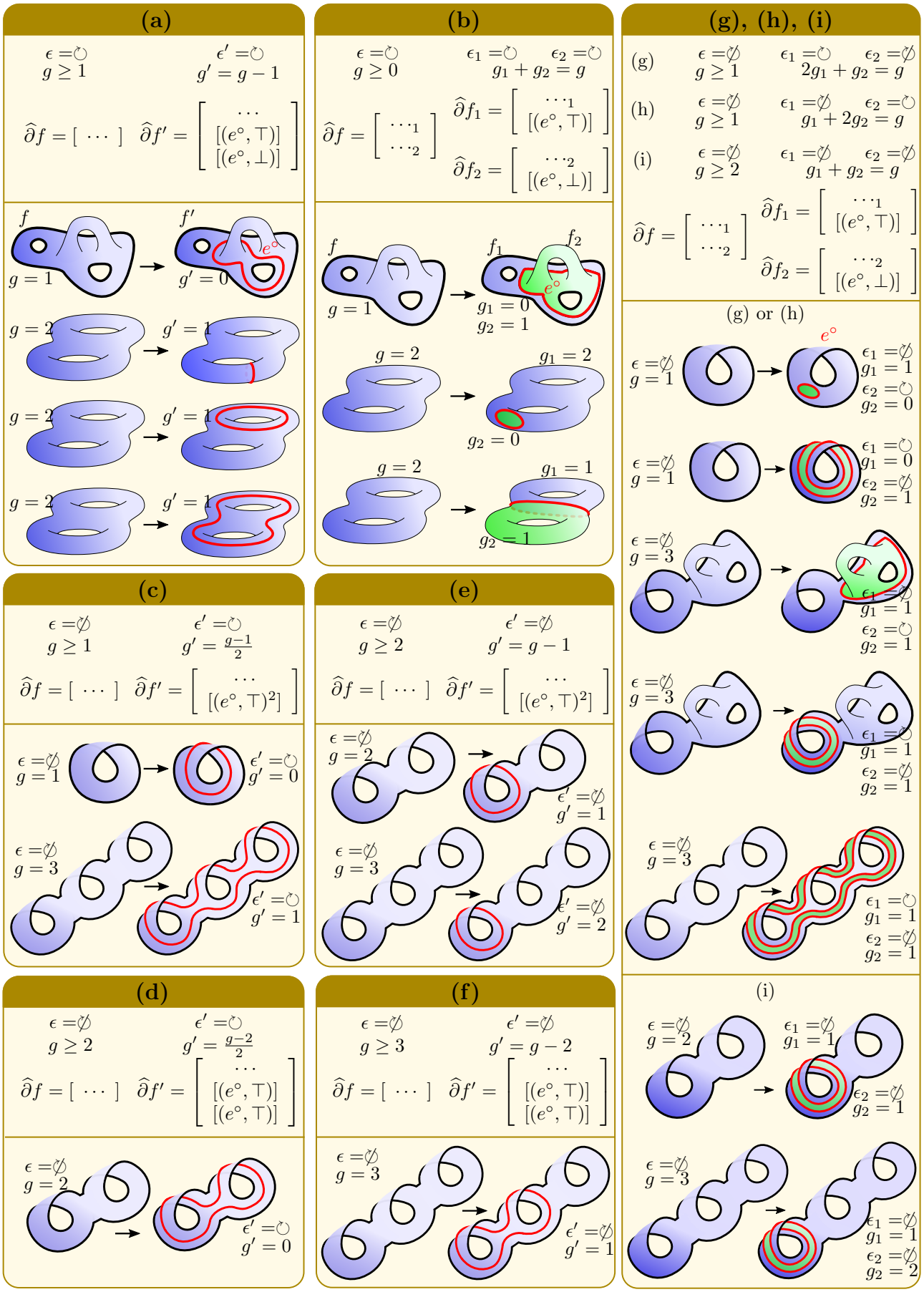


Figure 30: The different ways to cut a face at a closed edge. The labelling letters refer to the classification provided in Figure 29.

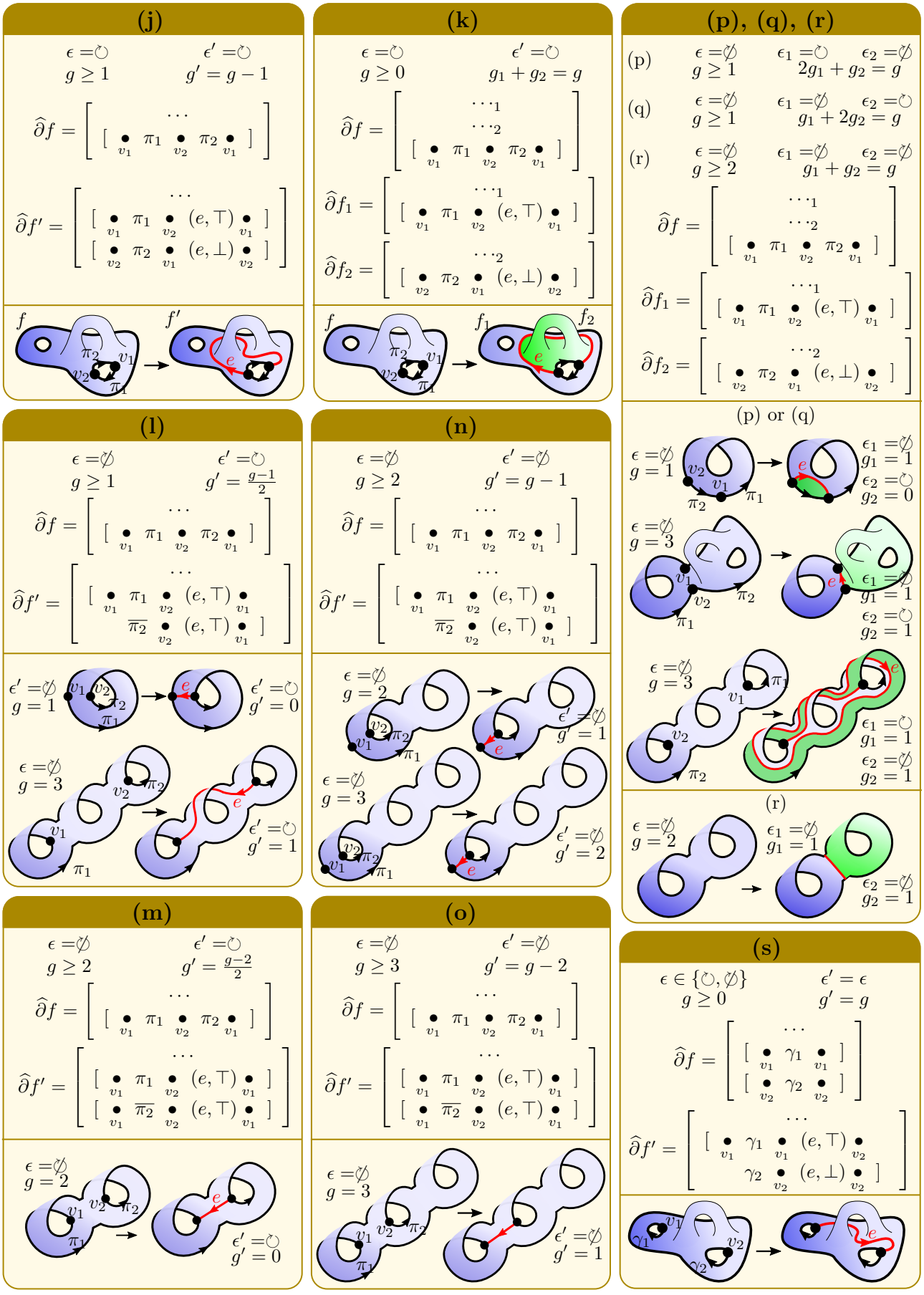


Figure 31: The different ways to cut a face at an open edge. The labelling letters refer to the classification provided in Figure 29.

then it is possible to *compute* in which case we are, and perform the appropriate operation. However, at the combinatorial level of the abstract PCS complex, no such realization as triangulation is assumed, therefore "how" the face is cut has to be specified somehow. For instance, if a face f is a sphere with k holes, and we cut it at a closed edge e°, then we know for sure that this disconnects f into two faces f_1 and and f_2. However, without more information, there is no way to know combinatorially which holes of f must be transferred to f_1, and which holes must be transferred to f_2. Hence, this information has to be given as input of the topological operator. In this case, the operator could be:

CutSphereAtClosedEdge($f \in F$, $I \subseteq \mathbb{N}$)

Require: $\epsilon(f) = \circlearrowleft$ and $g(f) = 0$

```
e° ← CreateClosedEdge()                    ▷ Create the cut edge e° and the two faces f1 and f2
f1 ← CreateFace(↻, 0)
f2 ← CreateFace(↻, 0)
for all cycle γi of f do                   ▷ Distribute the cycles of f among f1 and f2
    if i ∈ I then
        AddCycleToFace(f1,γi(f))
    else
        AddCycleToFace(f2,γi(f))
AddSimpleCycleToFace(f1,e°,⊤,1)            ▷ Add the cycle [(e°, ⊤)] to f1 and the cycle [(e°, ⊥)] to f2
AddSimpleCycleToFace(f2,e°,⊥,1)
HardDelete(f)                              ▷ Delete f
```

The above topological operator is quite simple, but things get much more complicated when f is not a sphere, and especially when f is non-orientable. In order to cover all the different cases with a finite but exhaustive set of topological operators, it is necessary to *classify* all these different cases. We call this the ***face-cut classification***, summarized in Figure 29, illustrated in Figure 30 and Figure 31, and detailed in the following subsections.

7.5.5 Cutting an orientable face at a closed edge

The first way to cut a face is via a closed edge included in the face. We recall that the geometric realization of a face is $\text{int}(\mathbb{F}_{\epsilon,g,k})$, and all the possibilities are illustrated in Figure 3. As can be seen in Figure 29, there exist many ways to choose a closed edge e° inside the interior of a face f. In this section and the following, we classify all of them.

First, let us consider the case where f is orientable. Let e° be a closed edge included in f. Thus, the pointset $f \setminus e^\circ$ is either connected or it is not. If $f \setminus e^\circ$ is connected, this completely determines the cut, i.e. any choice of e° included in an orientable face f such that $f \setminus e^\circ$ is connected leads to the same PCS complex up to homeomorphism, i.e. they have the same abstract PCS complex. This abstract PCS complex is obtained from the abstract PCS complex before the cut by decreasing the genus of f by one, and adding the two cycles $[(e^\circ, \top)]$ and $[(e^\circ, \bot)]$ to f. This is performed by the topological operator below:

CutOrientableFaceAtNonDisconnectingClosedEdge($f \in F$)

Require: $\epsilon(f) = \circlearrowleft$ and $g(f) \geq 1$

```
g(f) ← g(f) − 1
e° ← CreateClosedEdge()
AddSimpleCycleToFace(f,e°,⊤,1)
AddSimpleCycleToFace(f,e°,⊥,1)
```

If, at the contrary, $f \setminus e^\circ$ is not connected, then this means that it has two connected components f_1 and f_2, both orientable and satisfying $g(f) = g(f_1) + g(f_2)$, where the cycles of f are distributed among f_1 and f_2, the cycle $[(e^\circ, \top)]$ is added to f_1, and the cycle $[(e^\circ, \bot)]$ is added to f_2. However, the actual values of $g(f_1)$ and $g(f_2)$, as well as which cycles are transferred to f_1 and which cycles are transferred to f_2 cannot be

determined combinatorially without an underlying triangulation, and must therefore be an input of the following PCS topological operator. We note that the "CutSphereAtClosedEdge" operator that we have presented as a motivating example is redundant with this operator and therefore is not part of the classification.

CutOrientableFaceAtDisconnectingClosedEdge($f \in F$, $g_1 \in \mathbb{N}$, $g_2 \in \mathbb{N}$, $I \subseteq \mathbb{N}$)

Require: $\epsilon(f) = \circlearrowleft$ and $g(f) = g_1 + g_2$

$e^\circ \leftarrow$ CreateClosedEdge() ▷ Create the cut edge e° and the two faces f_1 and f_2
$f_1 \leftarrow$ CreateFace($\circlearrowleft$, g_1)
$f_2 \leftarrow$ CreateFace($\circlearrowleft$, g_2)
for all cycle γ_i of f **do** ▷ Distribute the cycles of f among f_1 and f_2
    if $i \in I$ **then**
        AddCycleToFace(f_1,$\gamma_i(f)$)
    else
        AddCycleToFace(f_2,$\gamma_i(f)$)
AddSimpleCycleToFace(f_1,e°,$\top$,1) ▷ Add the cycle $[(e^\circ, \top)]$ to f_1 and the cycle $[(e^\circ, \bot)]$ to f_2
AddSimpleCycleToFace(f_2,e°,$\bot$,1)
HardDelete(f) ▷ Delete f

7.5.6 Cutting a non-orientable face at a closed edge

In this section, let us consider the case where a non-orientable f is cut at a closed edge e°. The pointset $f' = f \setminus e^\circ$ is either connected or it is not, and let us first consider the case where it is connected. At the contrary to the orientable case presented in the previous section, the information "$f \setminus e^\circ$ is connected" does not fully determine the cut, unless $g(f) = 1$. More specifically, it is in fact always possible to choose e° such that f' becomes orientable, but if $g(f) \geq 2$ then it is also possible to choose e° such that f' stays non-orientable. Reasoning with the Euler characteristic, it can be shown that if f' is orientable, then this information fully determines the cut, which is given by the following topological operator:

CutNonOrientableFaceAtNonDisconnectingOrientingClosedEdge($f \in F$)

Require: $\epsilon(f) = \emptyset$

if $g(f)$ is odd **then**
    $\epsilon(f) \leftarrow \circlearrowleft$
    $g(f) \leftarrow \frac{g-1}{2}$
    $e^\circ \leftarrow$ CreateClosedEdge()
    AddSimpleCycleToFace(f,e°,$\top$,2)
else
    $\epsilon(f) \leftarrow \circlearrowleft$
    $g(f) \leftarrow \frac{g-2}{2}$
    $e^\circ \leftarrow$ CreateClosedEdge()
    AddSimpleCycleToFace(f,e°,$\top$,1)
    AddSimpleCycleToFace(f,e°,$\top$,1)

However, if f' is non-orientable, then there still remains some ambiguity. Specifically, whenever $g \geq 2$ it is possible to cut by preserving non-orientability and adding only one boundary (i.e., adding the cycle $[(e^\circ, \top)^2]$), and whenever $g \geq 3$ it is also possible to cut by preserving non-orientability and adding two boundaries (i.e., add the cycle $[(e^\circ, \top)]$ twice). In the first scenario, it can be shown that the genus is decreased by one, and in the second scenario it can be shown that the genus is decreased by two. Therefore, this leads to the following two topological operators:

CutNonOrientableFaceAtNonDisconnectingNonOrientingOddClosedEdge($f \in F$)

Require: $\epsilon(f) = \not\circlearrowleft$ and $g \geq 2$

```
1 g(f) ← g − 1
2 e° ← CreateClosedEdge()
3 AddSimpleCycleToFace(f,e°,⊤,2)
```

CutNonOrientableFaceAtNonDisconnectingNonOrientingEvenClosedEdge($f \in F$)

Require: $\epsilon(f) = \not\circlearrowleft$ and $g \geq 3$

```
1 g(f) ← g − 2
2 e° ← CreateClosedEdge()
3 AddSimpleCycleToFace(f,e°,⊤,1)
4 AddSimpleCycleToFace(f,e°,⊤,1)
```

Now that we have finished to consider all the cases where $f \setminus e^\circ$ was connected, we are about to consider the cases where $f \setminus e^\circ$ is not connected, and therefore has two connected components f_1 and f_2. In this case, as with the orientable case, the cycles of f must be distributed among f_1 and f_2, the cycle $[(e^\circ, \top)]$ is added to f_1, and the cycle $[(e^\circ, \bot)]$ is added to f_2. Since f is non-orientable, it can be shown that f_1 and f_2 cannot be both orientable, but all three other combinations are possible: f_1 orientable and f_2 non-orientable; f_1 non-orientable and f_2 orientable; or both f_1 and f_2 non-orientable (however, the latter is only possible if $g(f) \geq 2$). Reasoning with the Euler characteristic, it can be shown that for each of these three cases, we have the genus relation, respectively: $g(f) = 2g(f_1) + g(f_2)$; $g(f) = g(f_1) + 2g(f_2)$; and $g(f) = g(f_1) + g(f_2)$. However, whether f_1 and f_2 are orientable and the actual values of $g(f_1)$ and $g(f_2)$ cannot be determined algorithmically without an underlying triangulation. Therefore, they are all input of the following topological operator that spans all the three cases:

CutNonOrientableFaceAtDisconnectingClosedEdge($f \in F$, $\epsilon_1, \epsilon_2 \in \{\circlearrowleft, \not\circlearrowleft\}$, $g_1, g_2 \in \mathbb{N}$, $I \subseteq \mathbb{N}$)

Require: $\epsilon(f) = \not\circlearrowleft$
Require: $\epsilon_1 = \not\circlearrowleft$ or $\epsilon_2 = \not\circlearrowleft$
Require: $(\epsilon_1 = \circlearrowleft$ and $\epsilon_2 = \not\circlearrowleft) \Rightarrow (g_2 \geq 1$ and $g(f) = 2g_1 + g_2)$
Require: $(\epsilon_1 = \not\circlearrowleft$ and $\epsilon_2 = \circlearrowleft) \Rightarrow (g_1 \geq 1$ and $g(f) = g_1 + 2g_2)$
Require: $(\epsilon_1 = \not\circlearrowleft$ and $\epsilon_2 = \not\circlearrowleft) \Rightarrow (g_1 \geq 1$, $g_2 \geq 1$, and $g(f) = g_1 + g_2)$

```
1  e° ← CreateClosedEdge()                    ▷ Create the cut edge e° and the two faces f1 and f2
2  f1 ← CreateFace(ε1, g1)
3  f2 ← CreateFace(ε2, g2)
4  for all cycle γi of f do                   ▷ Distribute the cycles of f among f1 and f2
5      if i ∈ I then
6          AddCycleToFace(f1,γi(f))
7      else
8          AddCycleToFace(f2,γi(f))
9  AddSimpleCycleToFace(f1,e°,⊤,1)            ▷ Add the cycle [(e°, ⊤)] to f1 and the cycle [(e°, ⊥)] to f2
10 AddSimpleCycleToFace(f2,e°,⊥,1)
11 HardDelete(f)                              ▷ Delete f
```

7.5.7 Cutting a face at an open edge starting and ending at the same hole

As illustrated in Figure 29, this case is very similar to cutting at a closed edge, and follows the same classification. The difference is that instead of adding the two cycles $[(e^\circ, \top)]$ and $[(e^\circ, \bot)]$ (resp., twice the cycle $[(e^\circ, \top)]$, or the single cycle $[(e^\circ, \top)^2]$), we remove one cycle γ_i (the cycle corresponding to the starting/ending hole), split it into two paths π_1 and π_2, then add the two cycles $[\pi_1, (e, \top)]$ and $[\pi_2, (e, \bot)]$ (resp., the two cycles $[\pi_1, (e, \top)]$

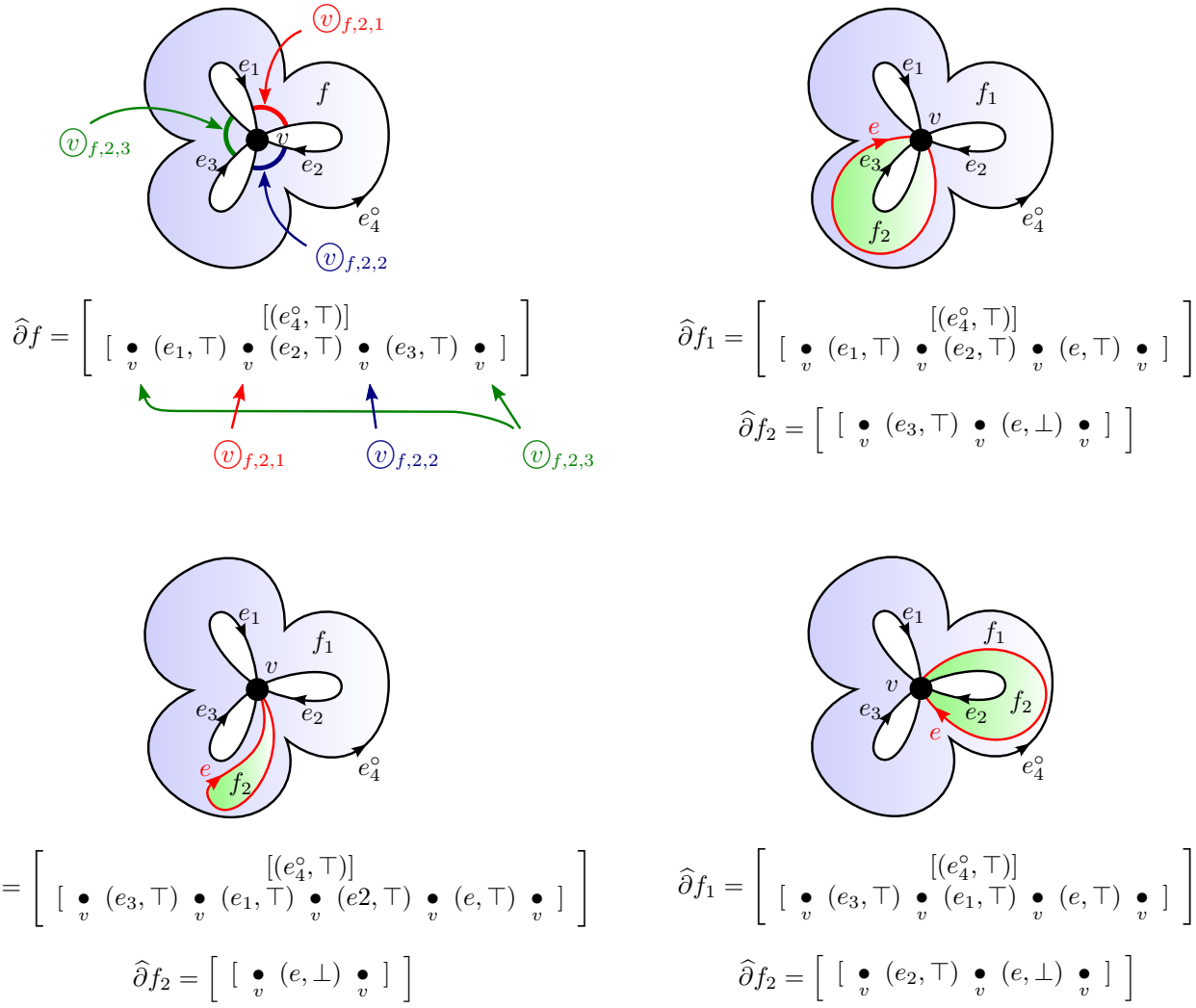


Figure 32: Three different cuts that start and end at the same vertex, but with different vertex-uses.

and $[\overline{\pi_2}, (e, \top)]$, or the single cycle $[\pi_1, (e, \top), \overline{\pi_2}, (e, \top)]$).

As illustrated in Figure 32, one issue is that the same vertex may be used several times by the same cycle, and hence knowing $v_{\text{start}}(e)$ and $v_{\text{end}}(e)$ is in general not enough information to combinatorially determine at which two indices the cycle γ_i must be split. Therefore, these indices must be explicitly provided as input of the topological operator, in the form of two integers j_{start} and j_{end}, in addition to the integer i specifying γ_i. Finally, we note that the same vertex-use $\textcircled{v}_{f,i,j}$ can be specified as both start and end vertex-use of the cut (cf Figure 32, bottom-left), in which case either π_1 or π_2 is empty, while the other is equal to the whole cycle γ_i. To disambiguate which is which, the caller of the operator must indicate either $j_{\text{end}} = j_{\text{start}}$ (to get $\pi_1 = \gamma_i$ and $\pi_2 = [\,]$), or $j_{\text{end}} = j_{\text{start}} + N(\gamma_i)$ (to get $\pi_1 = [\,]$ and $\pi_2 = \gamma_i$). In the special case where γ_i is a Steiner cycle, then j_{start} and j_{end} are not necessary and are simply ignored. We note that γ_i cannot be a simple cycle, since a simple cycle do not have any vertex-use.

Combining the above observations with the classification already given for cutting at a closed edge, we obtain six topological operators that are reported in this section. But first, we define the helper method SplitCycle() splitting a cycle γ into two paths π_1 and π_2, given two indices j_{start} and j_{end} indicating where to split γ:

SplitCycle($\gamma \in \Gamma$, $j_{\text{start}} \in \mathbb{N}$, $j_{\text{end}} \in \mathbb{N}$)

Require:
$$\begin{cases} \gamma \text{ is a Steiner cycle, or} \\ \begin{cases} \gamma \text{ is a non-simple cycle, and} \\ j_{\text{start}} \in [1..N(\gamma)], \text{ and} \\ \begin{cases} j_{\text{end}} = j_{\text{start}}, \text{ or} \\ j_{\text{end}} = j_{\text{start}} + N(\gamma), \text{ or} \\ j_{\text{end}} \in [1..N(\gamma)] \text{ and } j_{\text{start}} \neq j_{\text{end}} \end{cases} \end{cases} \end{cases}$$

if γ is a Steiner cycle **then** ▷ $\gamma = [v]$
  $v_{\text{start}} \leftarrow v(\gamma)$
  $v_{\text{end}} \leftarrow v(\gamma)$
  $\pi_1 \leftarrow [v(\gamma)]$
  $\pi_2 \leftarrow [v(\gamma)]$
else
  if $j_{\text{end}} = j_{\text{start}}$ **then** ▷ $\gamma = [\ h_1\ \cdots\ h_{j_{\text{start}}}\ \underset{v_{j_{\text{start}}}}{\bullet}\ h_{j_{\text{start}}+1}\ \cdots\ h_N\ ]$
    $v_{\text{start}} \leftarrow v_{j_{\text{start}}}(\gamma)$
    $v_{\text{end}} \leftarrow v_{j_{\text{start}}}(\gamma)$
    $\pi_1 \leftarrow [\ h_{j_{\text{start}}+1}\ \cdots\ h_N\ h_1\ \cdots\ h_{j_{\text{start}}}\ ]$
    $\pi_2 \leftarrow [v_{j_{\text{start}}}(\gamma)]$
  else if $j_{\text{end}} = j_{\text{start}} + N(\gamma)$ **then**
    $v_{\text{start}} \leftarrow v_{j_{\text{start}}}(\gamma)$
    $v_{\text{end}} \leftarrow v_{j_{\text{start}}}(\gamma)$
    $\pi_1 \leftarrow [v_{j_{\text{start}}}(\gamma)]$
    $\pi_2 \leftarrow [\ h_{j_{\text{start}}+1}\ \cdots\ h_N\ h_1\ \cdots\ h_{j_{\text{start}}}\ ]$
  else if $j_{\text{start}} < j_{\text{end}}$ **then** ▷ $\gamma = [\ h_1\ \cdots\ h_{j_{\text{start}}}\ \underset{v_{j_{\text{start}}}}{\bullet}\ h_{j_{\text{start}}+1}\ \cdots\ h_{j_{\text{end}}}\ \underset{v_{j_{\text{end}}}}{\bullet}\ h_{j_{\text{end}}+1}\ \cdots\ h_N\ ]$
    $v_{\text{start}} \leftarrow v_{j_{\text{start}}}(\gamma)$
    $v_{\text{end}} \leftarrow v_{j_{\text{end}}}(\gamma)$
    $\pi_1 \leftarrow [\ h_{j_{\text{end}}+1}\ \cdots\ h_N\ h_1\ \cdots\ h_{j_{\text{start}}}\ ]$
    $\pi_2 \leftarrow [\ h_{j_{\text{start}}+1}\ \cdots\ h_{j_{\text{end}}}\ ]$
  else ▷ $\gamma = [\ h_1\ \cdots\ h_{j_{\text{end}}}\ \underset{v_{j_{\text{end}}}}{\bullet}\ h_{j_{\text{end}}+1}\ \cdots\ h_{j_{\text{start}}}\ \underset{v_{j_{\text{start}}}}{\bullet}\ h_{j_{\text{start}}+1}\ \cdots\ h_N\ ]$
    $v_{\text{start}} \leftarrow v_{j_{\text{start}}}(\gamma)$
    $v_{\text{end}} \leftarrow v_{j_{\text{end}}}(\gamma)$
    $\pi_1 \leftarrow [\ h_{j_{\text{end}}+1}\ \cdots\ h_{j_{\text{start}}}\ ]$
    $\pi_2 \leftarrow [\ h_{j_{\text{start}}+1}\ \cdots\ h_N\ h_1\ \cdots\ h_{j_{\text{end}}}\ ]$
return $(v_{\text{start}}, v_{\text{end}}, \pi_1, \pi_2)$

We now report all the six different methods that can be used to cut a face at an open edge starting and ending at the same hole. In addition to specific parameters, all these methods have in common the parameters $f \in F$, $i \in \mathbb{N}$, $j_{\text{start}} \in \mathbb{N}$, $j_{\text{end}} \in \mathbb{N}$ with the following requirement that we report here for conciseness:

$$\begin{cases} i \in [1..k(f)], \text{ and} \\ \begin{cases} \gamma_i(f) \text{ is a Steiner cycle, or} \\ \begin{cases} \gamma_i(f) \text{ is a non-simple cycle, and} \\ j_{\text{start}} \in [1..N(\gamma_i(f))], \text{ and} \\ \begin{cases} j_{\text{end}} = j_{\text{start}}, \text{ or} \\ j_{\text{end}} = j_{\text{start}} + N(\gamma_i(f)), \text{ or} \\ j_{\text{end}} \in [1..N(\gamma_i(f))] \text{ and } j_{\text{start}} \neq j_{\text{end}} \end{cases} \end{cases} \end{cases} \end{cases} \tag{68}$$

CutOrientableFaceAtNonDisconnectingOpenEdge($f \in F$, $i \in \mathbb{N}$, $j_{\text{start}} \in \mathbb{N}$, $j_{\text{end}} \in \mathbb{N}$)

Require: $\epsilon(f) = \circlearrowleft$ and $g(f) \geq 1$
Require: Equation 68

$(v_{\text{start}}, v_{\text{end}}, \pi_1, \pi_2) \leftarrow \text{SplitCycle}(\gamma_i(f), j_{\text{start}}, j_{\text{end}})$
$g(f) \leftarrow g(f) - 1$
$e \leftarrow \text{CreateOpenEdge}(v_{\text{start}}, v_{\text{end}})$
$\text{AddNonSimpleCycleToFace}(f, [\pi_1, (e, \top)])$
$\text{AddNonSimpleCycleToFace}(f, [\pi_2, (e, \bot)])$
$\text{RemoveCycleFromFace}(f, i)$

CutOrientableFaceAtDisconnectingOpenEdge($f \in F$, $i \in \mathbb{N}$, $j_{\text{start}}, j_{\text{end}} \in \mathbb{N}$, $g_1, g_2 \in \mathbb{N}$, $I \subseteq \mathbb{N}$)

Require: $\epsilon(f) = \circlearrowleft$ and $g(f) = g_1 + g_2$
Require: Equation 68

$(v_{\text{start}}, v_{\text{end}}, \pi_1, \pi_2) \leftarrow \text{SplitCycle}(\gamma_i(f), j_{\text{start}}, j_{\text{end}})$
$e \leftarrow \text{CreateOpenEdge}(v_{\text{start}}, v_{\text{end}})$ ▷ Create the cut edge e and the two faces f_1 and f_2
$f_1 \leftarrow \text{CreateFace}(\circlearrowleft, g_1)$
$f_2 \leftarrow \text{CreateFace}(\circlearrowleft, g_2)$
for all cycle $\gamma_{i'}$ of f, $i' \neq i$ **do** ▷ Distribute the cycles of f, except γ_i, among f_1 and f_2
    if $i' \in I$ **then**
        $\text{AddCycleToFace}(f_1, \gamma_{i'}(f))$
    else
        $\text{AddCycleToFace}(f_2, \gamma_{i'}(f))$
$\text{AddNonSimpleCycleToFace}(f_1, [\pi_1, (e, \top)])$ ▷ Add the cycle $[\pi_1, (e, \top)]$ to f_1 and $[\pi_2, (e, \bot)]$ to f_2
$\text{AddNonSimpleCycleToFace}(f_2, [\pi_2, (e, \bot)])$
$\text{HardDelete}(f)$ ▷ Delete f

CutNonOrientableFaceAtNonDisconnectingOrientingOpenEdge($f \in F$, $i \in \mathbb{N}$, $j_{\text{start}} \in \mathbb{N}$, $j_{\text{end}} \in \mathbb{N}$)

Require: $\epsilon(f) = \not\circlearrowleft$
Require: Equation 68

$(v_{\text{start}}, v_{\text{end}}, \pi_1, \pi_2) \leftarrow \text{SplitCycle}(\gamma_i(f), j_{\text{start}}, j_{\text{end}})$
if $g(f)$ is odd **then**
    $\epsilon(f) \leftarrow \circlearrowleft$
    $g(f) \leftarrow \frac{g-1}{2}$
    $e \leftarrow \text{CreateOpenEdge}(v_{\text{start}}, v_{\text{end}})$
    $\text{AddNonSimpleCycleToFace}(f, [\pi_1, (e, \top), \overline{\pi_2}, (e, \top)])$
    $\text{RemoveCycleFromFace}(f, i)$
else
    $\epsilon(f) \leftarrow \circlearrowleft$
    $g(f) \leftarrow \frac{g-2}{2}$
    $e \leftarrow \text{CreateOpenEdge}(v_{\text{start}}, v_{\text{end}})$
    $\text{AddNonSimpleCycleToFace}(f, [\pi_1, (e, \top)])$
    $\text{AddNonSimpleCycleToFace}(f, [\overline{\pi_2}, (e, \top)])$
    $\text{RemoveCycleFromFace}(f, i)$

CutNonOrientableFaceAtNonDisconnectingNonOrientingOddOpenEdge($f \in F$, $i, j_{\text{start}}, j_{\text{end}} \in \mathbb{N}$)

Require: $\epsilon(f) = \emptyset$ and $g \geq 2$
Require: Equation 68

$(v_{\text{start}}, v_{\text{end}}, \pi_1, \pi_2) \leftarrow \text{SplitCycle}(\gamma_i(f),\ j_{\text{start}},\ j_{\text{end}})$
$g(f) \leftarrow g - 1$
$e \leftarrow \text{CreateOpenEdge}(v_{\text{start}},\ v_{\text{end}})$
$\text{AddNonSimpleCycleToFace}(f,\ [\pi_1, (e, \top), \overline{\pi_2}, (e, \top)])$
$\text{RemoveCycleFromFace}(f, i)$

CutNonOrientableFaceAtNonDisconnectingNonOrientingEvenOpenEdge($f \in F$, $i, j_{\text{start}}, j_{\text{end}} \in \mathbb{N}$)

Require: $\epsilon(f) = \emptyset$ and $g \geq 3$
Require: Equation 68

$(v_{\text{start}}, v_{\text{end}}, \pi_1, \pi_2) \leftarrow \text{SplitCycle}(\gamma_i(f),\ j_{\text{start}},\ j_{\text{end}})$
$g(f) \leftarrow g - 2$
$e \leftarrow \text{CreateOpenEdge}(v_{\text{start}},\ v_{\text{end}})$
$\text{AddNonSimpleCycleToFace}(f,\ [\pi_1, (e, \top)])$
$\text{AddNonSimpleCycleToFace}(f,\ [\overline{\pi_2}, (e, \top)])$
$\text{RemoveCycleFromFace}(f, i)$

CutNonOrientableFaceAtDisconnectingOpenEdge($f \in F$, $i, j_{\text{start}}, j_{\text{end}}, \epsilon_1, \epsilon_2, g_1, g_2,\ I \subseteq \mathbb{N}$)

Require: $\epsilon(f) = \emptyset$
Require: $\epsilon_1 = \emptyset$ or $\epsilon_2 = \emptyset$
Require: $(\epsilon_1 = \circlearrowleft$ and $\epsilon_2 = \emptyset) \Rightarrow (g_2 \geq 1$ and $g(f) = 2g_1 + g_2)$
Require: $(\epsilon_1 = \emptyset$ and $\epsilon_2 = \circlearrowleft) \Rightarrow (g_1 \geq 1$ and $g(f) = g_1 + 2g_2)$
Require: $(\epsilon_1 = \emptyset$ and $\epsilon_2 = \emptyset) \Rightarrow (g_1 \geq 1,\ g_2 \geq 1,$ and $g(f) = g_1 + g_2)$
Require: Equation 68

$(v_{\text{start}}, v_{\text{end}}, \pi_1, \pi_2) \leftarrow \text{SplitCycle}(\gamma_i(f),\ j_{\text{start}},\ j_{\text{end}})$
$e \leftarrow \text{CreateOpenEdge}(v_{\text{start}},\ v_{\text{end}})$ ▷ Create the cut edge e and the two faces f_1 and f_2
$f_1 \leftarrow \text{CreateFace}(\epsilon_1,\ g_1)$
$f_2 \leftarrow \text{CreateFace}(\epsilon_2,\ g_2)$
for all cycle $\gamma_{i'}$ of f, $i' \neq i$ **do** ▷ Distribute the cycles of f, except γ_i, among f_1 and f_2
    if $i \in I$ **then**
        $\text{AddCycleToFace}(f_1, \gamma_{i'}(f))$
    else
        $\text{AddCycleToFace}(f_2, \gamma_{i'}(f))$
$\text{AddNonSimpleCycleToFace}(f_1,\ [\pi_1, (e, \top)])$ ▷ Add the cycle $[\pi_1, (e, \top)]$ to f_1 and $[\pi_2, (e, \bot)]$ to f_2
$\text{AddNonSimpleCycleToFace}(f_2,\ [\pi_2, (e, \bot)])$
$\text{HardDelete}(f)$ ▷ Delete f

7.5.8 Cutting a face at an open edge starting and ending at different holes

Finally, the last case to consider is when the cut edge e is an open edge that starts and ends at different holes, represented by different cycles γ_{i_1} and γ_{i_2} of f. Fortunately, this case is actually very easy to handle, as it can be shown that it never disconnects f, and preserves its orientability and genus. Therefore, its only action is to merge the two cycles γ_{i_1} and γ_{i_2} into a single cycle, by joining them with e, as per the algorithm below:

RotatedCycle($\gamma \in \Gamma$, $j \in \mathbb{N}$)

Require: γ is a Steiner cycle, or a non-simple cycle with $j \in [1..N(\gamma)]$

1 **if** γ is a Steiner cycle **then** ▷ $\gamma = [v]$
2 $\quad v' \leftarrow v(\gamma)$
3 $\quad \gamma' \leftarrow \gamma$
4 **else** ▷ $\gamma = [\ h_1\ \cdots\ h_j \underset{v_j}{\bullet} h_{j+1}\ \cdots\ h_N\ ]$
5 $\quad v' \leftarrow v_j(\gamma)$
6 $\quad \gamma' \leftarrow [\ h_{j+1}\ \cdots\ h_N\ h_1\ \cdots\ h_j\ ]$
7 **return** (v', γ')

CutFaceAtOpenEdge($f \in F$, $i_1 \in \mathbb{N}$, $i_2 \in \mathbb{N}$, $j_1 \in \mathbb{N}$, $j_2 \in \mathbb{N}$)

Require: $(i_1, i_2) \in [1..k(f)]^2$
Require: $\gamma_{i_1}(f)$ is a Steiner cycle, or a non-simple cycle with $j_1 \in [1..N(\gamma_{i_1}(f))]$
Require: $\gamma_{i_2}(f)$ is a Steiner cycle, or a non-simple cycle with $j_2 \in [1..N(\gamma_{i_2}(f))]$

1 $(v_1, \gamma_1) \leftarrow$ RotatedCycle(i_1, j_1)
2 $(v_2, \gamma_2) \leftarrow$ RotatedCycle(i_2, j_2)
3 $e \leftarrow$ CreateOpenEdge(v_1, v_2)
4 AddNonSimpleCycleToFace(f, $[\gamma_1, (e, \top), \gamma_2, (e, \bot)]$)
5 RemoveCyclesFromFace(f,$\{i_1, i_2\}$)

7.5.9 Flipping cycles of non-orientable faces

When cutting a non-orientable face, there is one additional subtelty that has been omitted for clarity. It starts with the observation that orientations of cycles matter for orientable faces, but do not matter for non-orientable faces. This means that it is always possible to flip the orientation of any cycle of any non-orientable face, and this will result in a homeomorphic abstract PCS complex (i.e., their geometric realization is homeomorphic). Therefore, the topological operator below is essentially a no-operation, and can be performed at any time without changing what PCS complex it represents:

FlipCycle($f \in F$, $i \in \mathbb{N}$)

Require: $\epsilon(f) = \not\circlearrowleft$ and $i \in [1..k(f)]$

1 $\gamma_i(f) \leftarrow \overline{\gamma_i(f)}$

However, it *cannot* be performed for orientable faces since it could lead to non-homeomorphic PCS complexes. For instance, consider two abstract PCS complexes, each of them being made of one closed cycle e° and one face f. In the first abstract PCS complex, the face is $f = (\circlearrowleft, 0, [\ [(e^\circ, \top)], [(e^\circ, \top)]\ ])$, while in the second abstract PCS complex, the face is $f = (\circlearrowleft, 0, [\ [(e^\circ, \top)], [(e^\circ, \bot)]\ ])$. In both cases, it is possible to uncut at e°, but the resulting PCS complexes are not homeomorphic: one leads to a Klein bottle while the other leads to a torus, as formalized below:

$$f = (\circlearrowleft, 0, \begin{bmatrix} [(e^\circ, \top)] \\ [(e^\circ, \top)] \end{bmatrix}) \quad \xrightarrow{\text{UnCutAt}(e^\circ)} \quad f' = (\not\circlearrowleft, 2, [\ ]) \tag{69}$$

$$f = (\circlearrowleft, 0, \begin{bmatrix} [(e^\circ, \top)] \\ [(e^\circ, \bot)] \end{bmatrix}) \quad \xrightarrow{\text{UnCutAt}(e^\circ)} \quad f' = (\circlearrowleft, 1, [\ ]) \tag{70}$$

This has to be compared with the non-orientable case, where indeed orientation does not matter, as illustrated by the examples below:

$$f = (\not\emptyset, 1, \begin{bmatrix} [(e^\circ, \top)] \\ [(e^\circ, \top)] \end{bmatrix}) \quad \xrightarrow{\text{UnCutAt}(e^\circ)} \quad f' = (\not\emptyset, 3, [\,]) \tag{71}$$

$$f = (\not\emptyset, 1, \begin{bmatrix} [(e^\circ, \top)] \\ [(e^\circ, \bot)] \end{bmatrix}) \quad \xrightarrow{\text{UnCutAt}(e^\circ)} \quad f' = (\not\emptyset, 3, [\,]) \tag{72}$$

Therefore, when a non-orientable face f generates an orientable face f', f_1 or f_2 under the action of a cut, in addition to give as input which cycles to transfer to the orientable face, it is also necessary to give as input what orientations to give to these cycles, orientations that could be computed if e_{cut} was given as edges of an underlying triangulation. Also, in the case where a non-orientable face is cut at an open edge starting and ending at the same hole, there are in fact two possible non-homeomorphic outcomes of the cut: either merging γ_{i_1} and γ_{i_2} into $[\gamma_{i_1}, (e, \top), \gamma_{i_2}, (e, \bot)]$, or merging them into $[\gamma_{i_1}, (e, \top), \overline{\gamma_{i_2}}, (e, \bot)]$.

Instead of making the input of the topological operators more complicated that it already is, this can simply be achieved by calling FlipCycle() as many times as necessary before calling one of the CutNonOrientableFace[...] methods (or CutFaceAtOpenEdge() if f is non-orientable), and this sequence can be seen as the whole cut operator.

7.6 Uncut

We now present the uncut topological operator, which is the reverse of the cut operator. Since all the important ideas have already been covered in the previous section, we provide here the algorithm but do not comment it extensively. Nevertheless, here are two important observations:

- Given a cell c, it is not always possible to "uncut at c". More specifically, it is possible to uncut at c if and only if c may have been created as the cut cell of a cut topological operator.
- At the contrary to the cut operator, the uncut operator is *not* ambiguous. This means that indicating which cell to uncut at is the only necessary input. One way to interpret this fundamental difference between cut and uncut is that before the cut, we *do not know yet* e_{cut}, and hence we have to fully specify combinatorially *how* it cuts a given face. However, for the reverse operation, e_{cut} does exist, and hence we know exactly how it is used, e.g. as a frontier between two known faces. Merging back these two faces into one face is a non-ambiguous process, but during which information about e_{cut} is lost, reason why the reverse process is ambiguous.

CanUnCutAt($c \in C$)

```
1 if c ∈ V then
2     return CanUnCutAtVertex(c)
3 else if c ∈ E_∘ then
4     return CanUnCutAtClosedEdge(c)
5 else if c ∈ E_| then
6     return CanUnCutAtOpenEdge(c)
7 else if c ∈ F then
8     return false
```

CanUnCutAtVertex($v \in V$)

```
1  if star(v) = ∅ then
2      return false
3  else
       ▷ Count the number of end-vertex-uses, including edges with incident faces (unlike UnGlue).
4      N_incident−edges ← 0
5      N_end−vertex−use ← 0
6      for all edge e ∈ star(v) do
7          N_incident−edges ← N_incident−edges + 1
8          if v_start(e) = v then                          ▷ End-vertex-use (v)_{e,start}
9              N_end−vertex−use ← N_end−vertex−use + 1
10         if v_end(e) = v then                            ▷ End-vertex-use (v)_{e,end}
11             N_end−vertex−use ← N_end−vertex−use + 1
       ▷ Count the number of Steiner-vertex-uses.
12     N_Steiner−vertex−use ← 0
13     for all face f ∈ star(v) do
14         for all Steiner cycle γ_i^• = [v_i] ∈ ∂̂f do
15             if v_i = v then                             ▷ Steiner-vertex-use (v)_{f,i}
16                 N_Steiner−vertex−use ← N_Steiner−vertex−use + 1
       ▷ Check if v could have been created via CutFaceAtVertex().
17     if N_Steiner−vertex−use = 1 and N_end−vertex−use = 0 then
18         return true
       ▷ Check if v could have been created via CutClosedEdge(). This requires v to have a single incident
         edge e = (v, v), and cycles using e must be of the form [(e, β)^N].
19     if N_Steiner−vertex−use = 0 and N_end−vertex−use = 2 and N_incident−edges = 1 then
20         e ← only edge in star(v)
21         for all face f ∈ star(v) do
22             for all non-simple cycle γ_i ∈ ∂̂f do
23                 if γ_i uses e and ∄(β, N) s.t. γ_i = [(e, β)^N] then
24                     return false
25         return true
       ▷ Check if v could have been created via CutOpenEdge(). This requires v to have exactly two incident
         edges e_1 and e_2 each using v once, and cycles using v must not do any "switch-back" at v.
26     if N_Steiner−vertex−use = 0 and N_end−vertex−use = 2 and N_incident−edges = 2 then
27         (e_1, e_2) ← the two edges in star(v)
28         for all face f ∈ star(v) do
29             for all non-simple cycle γ_i ∈ ∂̂f do
30                 for all j ∈ [1..N(γ_i)] do
31                     if v_j = v and e_j(γ_i) = e_{j+1}(γ_i) then
32                         return false
33         return true
       ▷ All other cases mean that v could not have been created via a cut
34     return false
```

CanUnCutAtClosedEdge($e^\circ \in E_\circ$)

```
1  if star(e°) = ∅ then
2      return false
3  else
4      N_incident−faces ← 0
5      N_cycles−using−e ← 0
6      N_closed−edge−use ← 0
7      for all face f ∈ star(e°) do
8          N_incident−faces ← N_incident−faces + 1
9          for all simple cycle γ_i° ∈ ∂̂f do
10             if e°(γ_i°) = e° then                ▷ Closed-edge-uses (ê°)_{f,i,1} to (ê°)_{f,i,N(γ_i°)}
11                 N_cycles−using−e ← N_cycles−using−e + 1
12                 N_closed−edge−use ← N_closed−edge−use + N(γ_i°)
13     if N_closed−edge−use = 2 then
14         return true
15     else
16         return false
```

CanUnCutAtOpenEdge($e \in E_|$)

```
1  if star(e) = ∅ then
2      return false
3  else
4      N_incident−faces ← 0
5      N_cycles−using−e ← 0
6      N_open−edge−use ← 0
7      for all face f ∈ star(e) do
8          N_incident−faces ← N_incident−faces + 1
9          for all non-simple cycle γ_i ∈ ∂̂f do
10             CycleAlreadyCounted ← false
11             for all j ∈ [1..N(γ_i)] do
12                 if e_j(γ) = e then                ▷ Open-edge-use (e)_{f,i,j}
13                     N_open−edge−use ← N_open−edge−use + 1
14                     if not CycleAlreadyCounted then
15                         N_cycles−using−e ← N_cycles−using−e + 1
16                         CycleAlreadyCounted ← true
17     if N_open−edge−use = 2 then
18         return true
19     else
20         return false
```

UnCutAt($c \in C$)

```
1  if c ∈ V then
2      UnCutAtVertex(c)
3  else if c ∈ E_∘ then
4      UnCutAtClosedEdge(c)
5  else if c ∈ E_| then
6      UnCutAtOpenEdge(c)
7  else if c ∈ F then
8      Do nothing
```

UnCutAtVertex($v \in V$)

```
1  if NOT CanUnCutAtVertex(v) then
2      Do nothing
3  else
       ▷ Handle case where v could have been created via CutFaceAtVertex().
4      if N_{Steiner−vertex−use} = 1 and N_{end−vertex−use} = 0 then
5          f ← only face in star(v)
6          i ← index of Steiner cycle of f using v
7          RemoveCycleFromFace(f,i)
       ▷ Handle case where v could have been created via CutClosedEdge().
8      if N_{Steiner−vertex−use} = 0 and N_{end−vertex−use} = 2 and N_{incident−edges} = 1 then
9          e° ← CreateClosedEdge()
10         e ← only edge in star(v)
11         for all face f ∈ star(v) do
12             for all non-simple cycle γ_i ∈ ∂̂f do
13                 if γ_i uses e then
14                     (β, N) ← values such that γ_i = [(e, β)^N]
15                     γ_i(f) ← [(e°, β)^N]
16         HardDelete(e)
       ▷ Handle case where v could have been created via CutOpenEdge().
17     if N_{Steiner−vertex−use} = 0 and N_{end−vertex−use} = 2 and N_{incident−edges} = 2 then
           ▷ Compute h_1 and h_2, the two halfedges such that --h_1--> • --h_2--> (v).
18         (e_1, e_2) ← the two edges in star(v)
19         if v_end(e_1) = v then β_1 ← ⊤ else β_1 ← ⊥
20         if v_start(e_2) = v then β_2 ← ⊤ else β_2 ← ⊥
21         h_1 ← (e_1, β_1); h_2 ← (e_2, β_2)
           ▷ Create the new open edge e = (v_start(h_1), v_end(h_2)).
22         e ← CreateOpenEdge(v_start(h_1),v_end(h_2))
           ▷ Replace every occurrence of --h_1--> • --h_2--> by (e, ⊤) and every occurrence of <--h_2-- • <--h_1-- by (e, ⊥).
23         for all face f ∈ star(v) do
24             for all non-simple cycle γ_i ∈ ∂̂f do
25                 γ'_i ← [ ]
26                 for all j ∈ [1..N(γ_i)] do
27                     if e_j(γ_i) = e_1 then
28                         Do nothing.
29                     else if e_j(γ_i) = e_2 then
30                         if β_j(γ_i) = β_2 then
31                             Append (e, ⊤) to γ'_i
32                         else
33                             Append (e, ⊥) to γ'_i
34                     else
35                         Append h_j(γ_i) to γ'_i
36                 γ_i(f) ← γ'_i
           ▷ Delete e_1 and e_2.
37         HardDelete(e_1)
38         HardDelete(e_2)
       ▷ In any of the previous cases, delete v.
39     HardDelete(v)
```

UnCutAtClosedEdge($e^\circ \in E_\circ$)

```
1  if NOT CanUnCutAtClosedEdge(e°) then
2      Do nothing
3  else
4      if N_incident−faces = 1 then                    ▷ 1 face f′: Case (a), (c), (d), (e), or (f) (cf. Figure 30)
5          f′ ← face using e°
6          if N_cycles−using−e = 1 then                ▷ 1 cycle γ_i = [(e°, β)²]: Case (c) or (e)
7              i ← index of cycle of f′ using e°
8              RemoveCycleFromFace(f′,i)
9              if ϵ(f′) =∅ then                        ▷ f′ non-orientable: Case (e)
10                 g(f′) ← g(f′) + 1
11             else                                    ▷ f′ orientable: Case (c)
12                 ϵ(f′) ←∅
13                 g(f′) ← 2g(f′) + 1
14         else          ▷ 2 cycles γ_{i1} = [(e°, β1)] and γ_{i2} = [(e°, β2)]: Case (a), (d), or (f)
15             (i1, i2) ← indices of cycles of f′ using e°
16             β1 ← β(γ_{i1}(f′))
17             β2 ← β(γ_{i2}(f′))
18             RemoveCyclesFromFace(f′,{i1, i2})
19             if ϵ(f′) =∅ then                        ▷ f′ non-orientable: Case (f)
20                 g(f′) ← g(f′) + 2
21             else if β1 = β2 then                    ▷ f′ orientable, β1 = β2: Case (d)
22                 ϵ(f′) ←∅
23                 g(f′) ← 2g(f′) + 2
24             else                                    ▷ f′ orientable, β1 ≠ β2: Case (a)
25                 g(f′) ← g(f′) + 1
26     else          ▷ 2 faces f1 and f2, 2 cycles γ_{i1} = [(e°, β1)] and γ_{i2} = [(e°, β2)]: Case (b), (g), (h), or (i)
27         (f1, f2) ← faces using e°
28         i1 ← index of cycle of f1 using e°
29         i2 ← index of cycle of f2 using e°
30         β1 ← β(γ_{i1}(f1))
31         β2 ← β(γ_{i2}(f2))
32         if ϵ(f1) =∅ and ϵ(f2) =∅ then               ▷ ϵ1 =∅, ϵ2 =∅: Case (i)
33             f ← CreateFace(∅, g(f1) + g(f2))
34         else if ϵ(f1) =∅ and ϵ(f2) =○ then          ▷ ϵ1 =∅, ϵ2 =○: Case (h)
35             f ← CreateFace(∅, g(f1) + 2g(f2))
36         else if ϵ(f1) =○ and ϵ(f2) =∅ then          ▷ ϵ1 =○, ϵ2 =∅: Case (g)
37             f ← CreateFace(∅, 2g(f1) + g(f2))
38         else                                        ▷ ϵ1 =○, ϵ2 =○: Case (b)
39             f ← CreateFace(○, g(f1) + g(f2))
40             if β1 = β2 then
41                 for all i ∈ [1..k(f2)] do
42                     FlipCycle(f2,i)
43         for all i ∈ [1..k(f1)], i ≠ i1 do
44             AddCycleToFace(f,γ_i(f1))
45         for all i ∈ [1..k(f2)], i ≠ i2 do
46             AddCycleToFace(f,γ_i(f2))
47         HardDelete(f1)
48         HardDelete(f2)
49     HardDelete(e°)
```

UnCutAtOpenEdge($e \in E_|$)

```
1  if NOT CanUnCutAtOpenEdge(e) then
2      Do nothing
3  else
4      if N_incident−faces = 1 then                ▷ 1 face f′: Case (j), (l), (m), (n), (o), or (s) (cf. Figure 31)
5          f′ ← face using e
6          if N_cycles−using−e = 1 then            ▷ 1 cycle γ_i = [π1, (e, β1), π2, (e, β2)]: Case (l), (n), or (s)
7              i ← index of the cycle of f′ using e
8              γ_i ← γ_i(f′)
9              (j1, j2) ← indices of the two halfedges of γ_i using e
10             β1 ← β_{j1}(γ_i)
11             β2 ← β_{j2}(γ_i)
12             π1 ← SubPath(γ_i,j1,j2 − 1)
13             π2 ← SubPath(γ_i,j2,j1 − 1)
14             RemoveCycleFromFace(f′,i)
15             if β1 = β2 then
16                 AddCycleToFace(f′,[π1, π̄2])
17                 if ϵ(f′) =✆ then                ▷ β1 = β2, f′ non-orientable: Case (n)
18                     g(f′) ← g(f′) + 1
19                 else                            ▷ β1 = β2, f′ orientable: Case (l)
20                     ϵ(f′) ←✆
21                     g(f′) ← 2g(f′) + 1
22             else                                ▷ β1 ≠ β2: Case (s)
23                 AddCycleToFace(f′,[π1])
24                 AddCycleToFace(f′,[π2])
25         else          ▷ 2 cycles γ_{i1} = [π1, (e, β1)] and γ_{i2} = [π2, (e, β2)]: Case (j), (m), or (o)
26             (i1, i2) ← indices of the cycles of f′ using e
27             γ_{i1} ← γ_{i1}(f′)
28             γ_{i2} ← γ_{i2}(f′)
29             j1 ← index of the halfedge of γ_{i1} using e
30             j2 ← index of the halfedge of γ_{i2} using e
31             β1 ← β_{j1}(γ_{i1})
32             β2 ← β_{j2}(γ_{i2})
33             π1 ← SubPath(γ_{i1},j1,j1 − 1)
34             π2 ← SubPath(γ_{i2},j2,j2 − 1)
35             RemoveCyclesFromFace(f′,{i1, i2})
36             if ϵ(f′) =✆ then                    ▷ f′ non-orientable: Case (o)
37                 if β1 = β2 then
38                     AddCycleToFace(f′,[π1, π̄2])
39                 else
40                     AddCycleToFace(f′,[π1, π2])
41                 g(f′) ← g(f′) + 2
42             else if β1 = β2 then                ▷ f′ orientable, β1 = β2: Case (m)
43                 AddCycleToFace(f′,[π1, π̄2])
44                 ϵ(f′) ←✆
45                 g(f′) ← 2g(f′) + 2
46             else                                ▷ f′ orientable, β1 ≠ β2: Case (j)
47                 AddCycleToFace(f′,[π1, π2])
48                 g(f′) ← g(f′) + 1
49     else      ▷ 2 faces f1 and f2, 2 cycles γ_{i1} = [π1, (e, β1)] and γ_{i2} = [π2, (e, β2)]: Case (k), (p), (k), or (r)
50         (f1, f2) ← faces using e
51         i1 ← index of cycle of f1 using e
52         i2 ← index of cycle of f2 using e
53         γ_{i1} ← γ_{i1}(f1)
```

```
54      γ_{i_2} ← γ_{i_2}(f_2)
55      j_1 ← index of the halfedge of γ_{i_1} using e
56      j_2 ← index of the halfedge of γ_{i_2} using e
57      β_1 ← β_{j_1}(γ_{i_1})
58      β_2 ← β_{j_2}(γ_{i_2})
59      π_1 ← SubPath(γ_{i_1},j_1,j_1 − 1)
60      π_2 ← SubPath(γ_{i_2},j_2,j_2 − 1)
61      if ε(f_1) =∅ and ε(f_2) =∅ then                          ▷ ε_1 =∅, ε_2 =∅: Case (r)
62          f ← CreateFace(∅, g(f_1) + g(f_2))
63      else if ε(f_1) =∅ and ε(f_2) =↻ then                     ▷ ε_1 =∅, ε_2 =↻: Case (q)
64          f ← CreateFace(∅, g(f_1) + 2g(f_2))
65      else if ε(f_1) =↻ and ε(f_2) =∅ then                     ▷ ε_1 =↻, ε_2 =∅: Case (p)
66          f ← CreateFace(∅, 2g(f_1) + g(f_2))
67      else                                                     ▷ ε_1 =↻, ε_2 =↻: Case (k)
68          f ← CreateFace(↻, g(f_1) + g(f_2))
69          if β_1 = β_2 then
70              for all i ∈ [1..k(f_2)] do
71                  FlipCycle(f_2,i)
72      for all i ∈ [1..k(f_1)], i ≠ i_1 do
73          AddCycleToFace(f,γ_i(f_1))
74      for all i ∈ [1..k(f_2)], i ≠ i_2 do
75          AddCycleToFace(f,γ_i(f_2))
76      if β_1 = β_2 then
77          AddCycleToFace(f,[π_1, π_2^-])
78      else
79          AddCycleToFace(f,[π_1, π_2])
80      HardDelete(f_1)
81      HardDelete(f_2)
82  HardDelete(e°)
```

8 Vector graphics complex

In the previous sections, we have first introduced the notion of *PCS complex*: a cell decomposition of a topological space. Then, we have introduced the notion of *abstract PCS complex*: a combinatorial structure made of abstract vertices, edges, halfedges, cycles, and faces. Each abstract PCS complex $\mathcal{P}$ is the combinatorial description of a PCS complex that we call the *geometric realization* of $\mathcal{P}$. Conversely, each PCS complex $\mathcal{K}$ can be combinatorially described by an abstract PCS complex that we call the *presentation scheme* of $\mathcal{K}$. This duality made possible to infer what the topological operators on abstract PCS complexes should be (as opposed to arbitrarily choose one that seems to make sense). In this section, we see that vector graphics complexes (VGCs) can be defined in terms of abstract PCS complexes, and provide more theoretical insight about VGCs using what has been learnt about PCS complexes.

8.1 Definitions

A ***vector graphics complex*** (VGC) is defined as an abstract PCS complex without the information of orientability and genus. More formally, a VGC is a tuple $\mathcal{P} = (C, \dim, \widehat{\partial})$, where C, dim, and $\widehat{\partial}$ have the constraints defined in Section 5.1. In a sense, a VGC can be seen as an abstract PCS complex where orientabilities and genuses are "unknown" since in the context of a vector graphics program, we do not have access to such information. Therefore, a given VGC face can *potentially* be considered to have any arbitrary genus or orientability. This departs fundamentally from the concept of planar maps, which are assumed to be embeddable in the plane, and consequently whose faces are necessarily orientable of genus zero.

Given a VGC, an ***immersion in*** $\mathbb{R}^2$ can be defined, as detailed in Section 4.2 of [Dalstein et al., 2014], by augmenting the structure with a point $p_v \in \mathbb{R}^2$ for each vertex, a map $\Gamma_e : [0,1] \to \mathbb{R}^2$ for each open edge, and a map $\Gamma_e : \mathbb{S}^1 \to \mathbb{R}^2$ for each closed edge. This defines a pointset $|c| \subset \mathbb{R}^2$ for each cell c. However, we note that this *does not* define a geometric realization, because the pointsets are allowed to overlap, or even not be connected.

A ***geometric realization*** can be defined by assigning an orientability-genus pair to each face, which defines an abstract PCS complex, which defines a unique geometric realization (cf. Section 5.3). Orientability-genus pairs can be assigned either arbitrarily, or via *heuristics* based on a given immersion. For instance, the left column of Figure 33 shows three different immersions of the same VGC: the first one "looks like" a disk (orientable surface of genus 0), the second one "looks like" a Möbius strip (non-orientable surface of genus 1), and the last one "looks like" an orientable surface of genus 1. Hence, a reasonable heuristic would respectively assign the following orientabily-genus pairs: $(\circlearrowleft, 0)$, $(\not\circlearrowleft, 1)$, and $(\circlearrowleft, 1)$. Since different orientability-genus pairs lead to different geometric realizations, we see that the geometric realization is not invariant to planar deformations of the immersion (i.e., deforming the edges of the VGC). In practice, this is never an issue since there is no *need* to define/construct such a geometric realization. It is only useful for a *theoretical understanding* of the topological spaces represented by vector graphics complexes, which is useful to *design* the topological operators.

8.2 Consequences of non-planarity

Planar maps [Baudelaire and Gangnet, 1989, Asente et al., 2007], extended to support closed edges and Steiner cycles, can be seen as the subset of PCS complexes that can be embedded in the plane, i.e. that are *planar*, while a VGC can potentially represent *any* PCS complex. This seemingly harmless difference has in fact tremendous consequences. Indeed, a PCS complex that is planar has the following properties that general PCS complexes do not have:

1. Every face is orientable of genus zero.
2. Every edge has exactly two face-uses (considering the exterior as an infinite face)

By analyzing the face-cut classification (cf. Figure 29), one can see that the first property implies that the cut topological operator on planar maps is "not ambiguous". More precisely, cutting a face f at a closed edge is fully determined by which cycle to transfer to f_1 and which cycle to transfer to f_2 (Figure 30, Case (b)). Similarly, cutting a face f at an open edge is fully determined by the two vertex-uses $\textcircled{v}_{f,i,j}$ and $\textcircled{v}_{f,i',j'}$ where the cut occurs, and if $i = i'$ by which cycle to transfer to f_1 and which cycle to transfer to f_2 (Figure 31, Case (k) if $i = i'$, Case (s) if $i \neq i'$). The case of cutting with an open edge when $i = i'$ is illustrated in Figure 33 (a), fully detailed in Figure 28.

However, this "non-ambiguity" is no longer true for VGCs. Since a VGC face may potentially be non-orientable and/or of strictly positive genus, any of the cases illustrated Figure 30 and 31 may apply, whereas only the cases (b), (k), and (s) apply to planar maps. In other words, heuristics based on the 2D geometry of edges have to be designed to decide which one of the many non-equivalent cut algorithms should be used. For instance, as illustrated Figure 33, using the "planar-map cut" may not capture the user's intent. In these cases, other algorithms should be used, which correspond to interpreting the face as non-orientable (Figure 33(c)), or orientable but with a strictly positive genus (Figure 33(e)). Finding good heuristics is still an open problem, but the PCS formalism introduced in this report has made possible to identify the exhaustive list of possible algorithms to choose from.

The second property has consequences on how planar maps are implemented: halfedges are "true halfedges" in the sense of the halfedge data structure. They are typically implemented with a *reference semantics* (i.e., as pointers), they go by pair and refer to their "opposite halfedge" and "next halfedge". However, since edges of VGCs may be used three times or more, the term "halfedge" is actually an abuse of language, merely a more convenient, shorter name for "oriented edge". Vertices, edges and faces are the objects with a reference semantics, but halfedges have a value semantics. A structure similar to the radial-edge structure could be used instead, in which case a reference semantics for halfedges could be used (i.e., edge-uses are actual objects with an identity), but it is not necessary and is not the choice that we personally made.

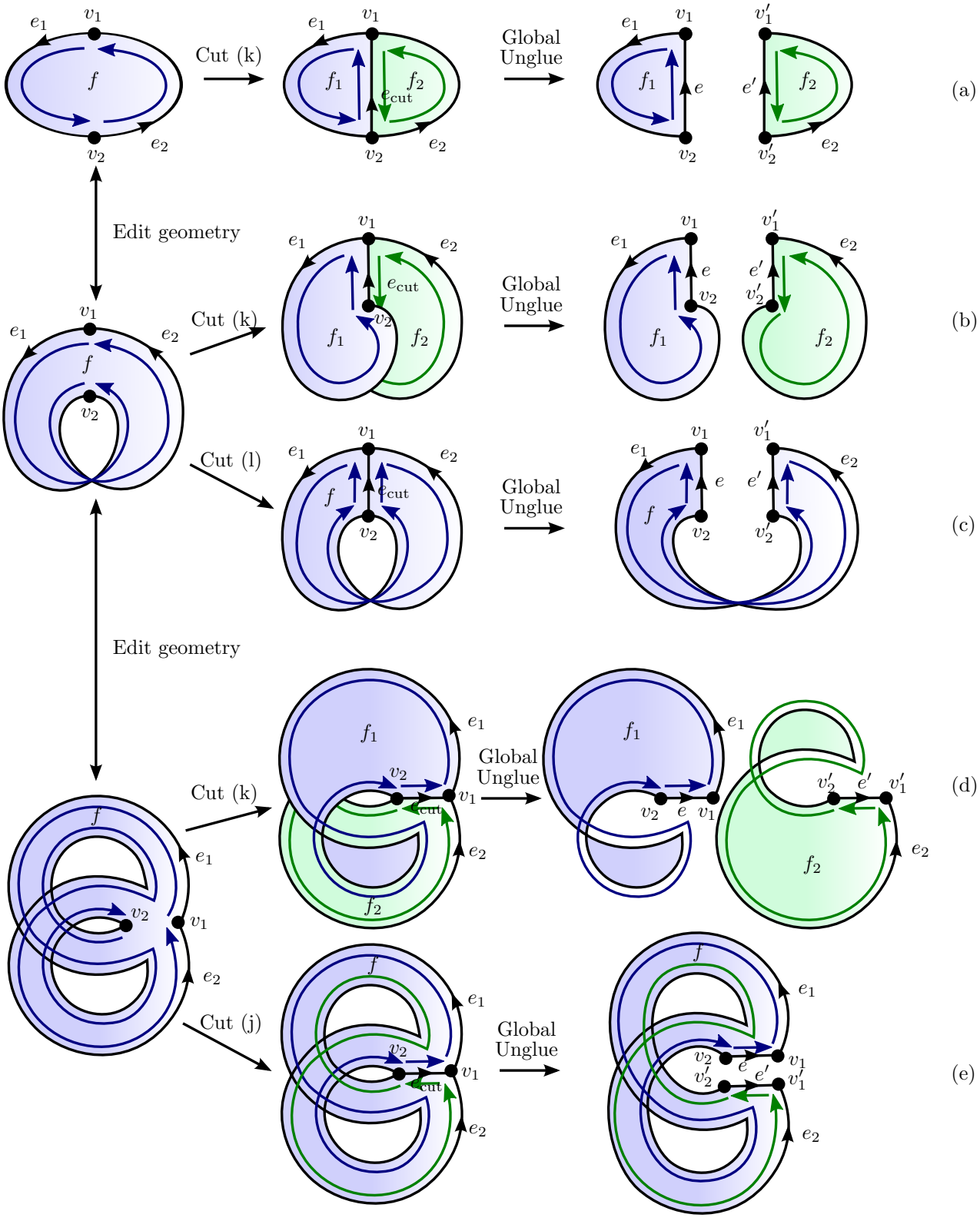


Figure 33: Left column: Three different immersions of the same VGC. Middle column: the result of applying a given cut algorithm to the VGC (the letters (k), (l) and (j) refer to Figure 31). Right column: the result of applying unglue to all cells and modifying slightly the geometry, for better visualization of the cut. This illustrates that unlike planar maps, cutting a VGC is an ambiguous operation. While there is only one way a planar map face can be cut (i.e., applying Cut (k), cf. row (a)), there are many non-equivalent ways a VGC face can be cut. Choosing the "planar map way" may lead to unexpected results (cf. rows (b) and (d)), in which case choosing an alternative cut algorithm may better capture the user's intent (cf. rows (c) and (e)).

8.3 Towards an immersion-independent geometric realization

We have seen that by assigning orientability-genus pairs to faces, it is possible to define a geometric realization of a VGC as the geometric realization of an abstract PCS complex. However, this geometric realization depends on the pairs chosen, which can be interpreted as depending on a given immersion of the VGC. From this observation, an interesting question is: is it possible to define a geometric realization directly for the VGC, i.e. without assigning orientability-genus pairs and interpreting the VGC as an abstract PCS complex? If the answer was "yes", then the whole concept of PCS complex would seem unnecessary to study the VGC. In this section, we show that the short answer is "no" (cf. next paragraph), or at the very least "yes, but..." (last paragraph).

In this section, let us try to define a geometric realization directly for the VGC, and see what is the issue that arises. It is fair to assume that the geometric realization of a face must be homeomorphic to a compact 2-manifold (or its interior, but we omit this detail for clarity). This is the widely accepted definition to capture the intuitive notion of "surface". Therefore, we know that it *must* be homeomorphic to $F_{\epsilon,g,k}$ for some ϵ, g, and k, where k should obviously be the number of cycles of the face. There comes the issue: if we want the geometric realization to define a *unique* topological space up to homeomorphism, then since the pair (ϵ, g) is not part of the combinatorial information, we have to abritrarily to decide on a value (ϵ_0, g_0), and use this same value for *all* faces, for *all* VGCs. For instance, planar maps make the choice of $\epsilon_0 =$ ◌ and $g_0 = 0$. In our case, we know that this choice is not suitable, since it would imply that only the "planar maps cuts" are available, and we know that other kinds of cuts are also useful. Allowing such cuts implies $g_0 > 0$. However, we have also seen that applying these cuts decreases the genus of the face, which contradicts the statement that all faces of all VGCs must have the same genus g_0. Therefore, defining a geometric realization directly for the VGC is impossible, unless we weaken our initial assumption that the geometric realization of a face is homeomorphic to a compact 2-manifold.

A more intuitive way to understand this contradiction is the following. Consider a VGC made of a single face with a single cycle (e.g., Figure 33, top-left). It "looks like" a disk, i.e. an orientable surface of genus zero. However, by deforming the edges bounding the face, you can make it "look like" a Möbius strip, thus cuts specific to Möbius strips must *also* be available. After such a cut, the face looks orientable again. But by deforming this face again, you can still make it look non-orientable again (the whole VGC would look like a "double Möbius strip with one cut"). By iterating this process, we see that VGC faces must be allowed to be cut an arbitrary number of time with a genus-decreasing cut. However, no compact 2-manifold can be cut an arbitrarily large number of times this way, since their genus is finite. Thus, the only way to define a unique geometric realization for VGCs is to use *non-compact* 2-manifold to realize faces.

Fortunately, the reasoning above leads us towards an appropriate non-compact 2-manifold: a non-orientable surface of *infinite* genus. More specifically, let us define $\mathbb{F}_{\emptyset,\infty,k}$ as the connected sum of a (countably) infinite number of projective planes, with k holes. Such a space does exist and can be formally defined, but it is not compact. It can be informally visualized by replacing g by ∞ in Figure 3, bottom-right. Using this exotic topological space, it is actually possible to define a geometric realization directly for the VGC. The geometric realization of a vertex is a point, the geometric realization of an open edge is the interior of a segment, the geometric realization of a closed edge is a circle, and the geometric realization of a face is the interior of $\mathbb{F}_{\emptyset,\infty,k}$. Hence, a most accurate answer to our initial question is actually: yes, a geometric realization can be defined directly for VGCs, but we must waive the assumption that faces are homeomorphic to *compact* 2-manifolds, and this is no less complex that studying the PCS complex as an intermediate concept.

8.4 Conclusion

A VGC can be seen as an abstract PCS complex without the orientability and genus information. Conversely, an abstract PCS complex can be defined from a VGC by assigning an orientability-genus pair to each face, which makes possible to define a *geometric realization* of the VGC. However, this geometric realization is not unique: choosing different orientability-genus pairs lead to different (and non-homeomorphic) geometric realizations. Unless we are willing to consider exotic non-compact topological spaces (e.g., infinite connected sum of projective planes), it is impossible to define a geometric realization directly for the VGC, which theoretically justifies the relevance of the PCS complex for the study of the VGC.

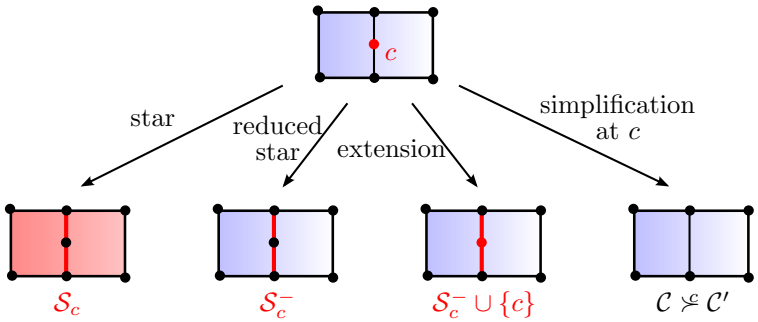


Figure 34: Reduced star and atomic simplification.

Since VGC faces can be interpreted as non-orientable and/or with strictly positive genus, then we have learnt from the study of the PCS complex that cutting a face is an ambiguous operation (unlike with planar maps). This means that for any given VGC face and any given immersion, there are several non-equivalent ways the face can be cut. They are all perfectly valid operations, but one of them may better capture the user's intent, and choosing which one it is requires heuristics based on the immersion.

To conclude, the VGC has proven to be one of these mathematical objects that appear very simple on the surface, but that hide a complexity deeper than one might expect. The intermediate concept of PCS complex is necessary to comprehend this complexity.

9 Simplification and equivalence of cell complexes

The goal of this last section is to formally define what *minimal decomposition* means, to be able to formulate our main conjecture stating that if we are given a PCS complex $\mathcal{K} = (X, \mathcal{C})$, then X has a unique minimal PCS decomposition $\mathcal{K}_m = (X, \mathcal{C}_m)$.

We first define the notion of atomic simplification, which is transforming a cell complex into another by merging a cell and its *reduced star* into a single cell. This operation is equivalent to the *uncut* topological operator presented in the previous section. If a cell complex can be transformed into another cell complex via a finite sequence of atomic simplifications or/and de-simplifications, we say that they are equivalent. If a cell complex cannot be simplified, it is called *minimal*, or *simple*. We show that every cell complex admits a minimal equivalent cell complex, and in the specific case of the dimension two or less (i.e., the case of PCS complex), we conjecture that this minimal equivalent complex is unique. A potential proof is to triangulate the PCS complex, and show that any sequence of simplification leads to the unique decomposition discussed in [De Floriani et al., 2003].

9.1 Simplification of cell complexes

In this section, we define the concept of *simplification*, intuitively an operation transforming a cell complex $\mathcal{K} = (X, \mathcal{C})$ into another complex $\mathcal{K}' = (X, \mathcal{C}')$, decomposing the same space with strictly fewer cells.

Reduced star Let $\mathcal{K} = (X, \mathcal{C})$ be a cell complex and $c \in \mathcal{C}$. The ***reduced star*** of c is defined by:

$$\mathcal{S}_c^- = \begin{cases} \varnothing & \text{if } \mathcal{S}_c = \varnothing \\ \{c' \in \mathcal{S}_c \mid \dim c' = n_c^-\} & \text{otherwise, where } n_c^- = \min_{c' \in \mathcal{S}_c}(\dim c') \end{cases} \tag{73}$$

In other words, the reduced star is the star reduced to its cells of lowest dimension, as illustrated in Figure 34. Note that n_c^- is not necessarily equal to $\dim(c) + 1$, for instance if you consider the cell complex decomposing $\mathbb{S}^2$ as a vertex $v \in \mathbb{S}^2$ and the face $\mathbb{S}^2 \backslash v$, then we have $\mathcal{S}_v^- = \{f\}$.

Extension The ***extension*** of a cell c is defined as its reduced star extended by c itself:

$$\widehat{\mathcal{S}}_c^- = \{c\} \cup \mathcal{S}_c^- \tag{74}$$

$$\widehat{c} \;=\; <\widehat{\mathcal{S}}_c^- > \;=\; < c, \mathcal{S}_c^- > \tag{75}$$

Atomic simplification Let $\mathcal{K} = (X, \mathcal{C})$ be a cell complex and $c \in \mathcal{C}$. We say that $\mathcal{K}$ can be simplified at c, and we write $\mathcal{K} \succ\!\!\!\cdot^c$, if and only if the following constraints are satisfied:

- $\mathcal{S}_c \neq \varnothing$
- $\mathcal{K}' = (X, \mathcal{C}')$ is a cell complex, where $\mathcal{C}' = (\mathcal{C} \backslash \widehat{\mathcal{S}}_c^-) \cup \{\widehat{c}\}$

In this case, we write $\mathcal{K} \succ\!\!\!\cdot^c \mathcal{K}'$. For the dimension two or less, checking whether "$\mathcal{K}$ can be simplified at c" can be done combinatorially with the algorithm CanUnCut(c). If yes, then $\mathcal{K}'$ is obtained by the algorithm UnCut(c).

Anonymous atomic simplification Let $\mathcal{K} = (X, \mathcal{C})$ and $\mathcal{K}' = (X', \mathcal{C}')$ be two cell complexes. We define the binary relation:

$$\mathcal{K} \succ\!\!\!\cdot^\bullet \mathcal{K}' \quad \Leftrightarrow \quad \exists c \in \mathcal{C},\ \mathcal{K} \succ\!\!\!\cdot^c \mathcal{K}' \tag{76}$$

It follows directly that if $(X, \mathcal{C}) \succ\!\!\!\cdot^\bullet (X', \mathcal{C}')$ then $X' = X$. Therefore, as an abuse of notation, we will often write $\mathcal{C} \succ\!\!\!\cdot^\bullet \mathcal{C}'$ instead of $\mathcal{K} \succ\!\!\!\cdot^\bullet \mathcal{K}'$, but it must be clear that $\succ\!\!\!\cdot^\bullet$ and the related binary relations are defined on cell complexes and not cell decompositions, since their definition requires the existence of $\mathcal{B}_c$.

Proposition 8. *If $\mathcal{C} \succ\!\!\!\cdot^\bullet \mathcal{C}'$, then $|\mathcal{C}| > |\mathcal{C}'|$, where $|\mathcal{C}|$ denotes the number of cells in $\mathcal{C}$.*

Proof. If $\mathcal{C} \succ\!\!\!\cdot^\bullet \mathcal{C}'$, then $\exists c \in \mathcal{C},\ \mathcal{C} \succ\!\!\!\cdot^c \mathcal{C}'$ and we have $\mathcal{C}' = (\mathcal{C} \backslash \widehat{\mathcal{S}}_c^-) \cup \{\widehat{c}\}$, thus $|\mathcal{C}'| = |\mathcal{C}| - |\widehat{\mathcal{S}}_c^-| + 1$. Since $\mathcal{C}$ can be simplified at c, this means $\mathcal{S}_c \neq \varnothing$, thus $\mathcal{S}_c^- \neq \varnothing$, thus $\widehat{\mathcal{S}}_c^-$ contains at least two cells: c and one belonging to $\mathcal{S}_c^-$. Thus $|\widehat{\mathcal{S}}_c^-| \geq 2$, thus $|\mathcal{C}'| \leq |\mathcal{C}| - 2 + 1$, thus $|\mathcal{C}'| \leq |\mathcal{C}| - 1$. □

Simplification We define the binary relation $\succ$ to be the transitive closure of $\succ\!\!\!\cdot^\bullet$ (i.e., the minimal transitive relation containing $\succ\!\!\!\cdot^\bullet$).

Proposition 9. *$\mathcal{C} \succ \mathcal{C}'$ if and only if a finite sequence of atomic simplification transforms $\mathcal{C}$ into $\mathcal{C}'$. Formally:*

$$\mathcal{C}_0 \succ \mathcal{C}' \quad \Leftrightarrow \quad \begin{cases} \exists k \in \mathbb{N}^+, \\ \exists \mathcal{C}_1, \ldots, \mathcal{C}_{k-1} \text{ decomposing } X, \\ \forall i \in [0..k-1],\ \exists c_i \in \mathcal{C}_i, \\ \mathcal{C}_0 \succ\!\!\!\cdot^{c_0} \mathcal{C}_1 \succ\!\!\!\cdot^{c_1} \cdots \succ\!\!\!\cdot^{c_{k-2}} \mathcal{C}_{k-1} \succ\!\!\!\cdot^{c_{k-1}} \mathcal{C}' \end{cases} \tag{77}$$

Proof. An equivalent definition of the transitive closure is:

- $(\succ\!\!\!\cdot^\bullet)^1 = (\succ\!\!\!\cdot^\bullet)$
- $\forall k \in \mathbb{N}^+, (\succ\!\!\!\cdot^\bullet)^{k+1} = (\succ\!\!\!\cdot^\bullet) \circ (\succ\!\!\!\cdot^\bullet)^k$
- $(\succ) = (\succ\!\!\!\cdot^\bullet)^+ = \bigcup_{k \in \mathbb{N}^+} (\succ\!\!\!\cdot^\bullet)^k$

Thus if $\mathcal{C} \succ \mathcal{C}'$, there exists $k \in \mathbb{N}^+$ such that $\mathcal{C} (\succ\!\!\!\cdot^\bullet)^k \mathcal{C}'$, thus there exist $\mathcal{C}_1 \ldots, \mathcal{C}_{k-1}$ such that

$$\mathcal{C}_0 \succ\!\!\!\cdot^\bullet \mathcal{C}_1 \succ\!\!\!\cdot^\bullet \cdots \succ\!\!\!\cdot^\bullet \mathcal{C}_{k-1} \succ\!\!\!\cdot^\bullet \mathcal{C}'. \tag{78}$$

From the definition of $\succ\!\!\!\cdot^\bullet$ it comes (with $\mathcal{C}_k = \mathcal{C}'$):

$$\forall i \in [0..k-1], \exists c_i \in \mathcal{C}_i, \mathcal{C}_i \succ\!\!\!\cdot^{c_i} \mathcal{C}_{i+1} \tag{79}$$

that can be rewritten in

$$\mathcal{C}_0 \succ\!\!\!\cdot^{c_0} \mathcal{C}_1 \succ\!\!\!\cdot^{c_1} \cdots \succ\!\!\!\cdot^{c_{k-2}} \mathcal{C}_{k-1} \succ\!\!\!\cdot^{c_{k-1}} \mathcal{C}'. \tag{80}$$

The converse implication directly comes from the fact that $\succ$ is transitive and contains $\succ\!\!\!\cdot^\bullet$. □

Proposition 10. *If $\mathcal{C} \succ \mathcal{C}'$, then $|\mathcal{C}| > |\mathcal{C}'|$.*

Proof. If $\mathcal{C} \succ \mathcal{C}'$, then $\mathcal{C} \overset{c_0}{\succ} \mathcal{C}_1 \overset{c_1}{\succ} \cdots \overset{c_{k-2}}{\succ} \mathcal{C}_{k-1} \overset{c_{k-1}}{\succ} \mathcal{C}'$, then $|\mathcal{C}| > |\mathcal{C}_1| > \cdots > |\mathcal{C}_{k-1}| > |\mathcal{C}'|$, then $|\mathcal{C}| > |\mathcal{C}'|$. □

Proposition 11. *$\succ$ is a strict partial order.*

Proof. We verify below that it is irreflexive, transitive and asymmetric:

- Irreflexivity: we have $\neg(|\mathcal{C}| > |\mathcal{C}|)$, thus $\neg(\mathcal{C} \succ \mathcal{C})$.
- Transitivity: by definition.
- Asymmetry: If $\mathcal{C} \succ \mathcal{C}'$ then $(|\mathcal{C}| > |\mathcal{C}'|)$ then $\neg(|\mathcal{C}'| > |\mathcal{C}|)$ then $\neg(\mathcal{C}' \succ \mathcal{C})$. □

Minimal complex Let Ω be a set of cell complexes. A cell complex $\mathcal{K} \in \Omega$ is said to be a minimal element of Ω if it is minimal for $\succ$, i.e. if there are no $\mathcal{K}' \in \Omega$ such that $\mathcal{K} \succ \mathcal{K}'$. In other words, a cell complex is said to be minimal if it cannot be simplified to another cell complex in Ω. Formally:

$$\mathcal{K} \text{ minimal in } \Omega \quad \Leftrightarrow \quad \forall \mathcal{K}' \in \Omega,\ \neg(\mathcal{K} \succ \mathcal{K}') \tag{81}$$

By extension, if no set Ω is specified, $\mathcal{K}$ is said to be *minimal*, or *simple*, if it cannot be simplified:

$$\mathcal{K} = (X, \mathcal{C}) \text{ minimal} \quad \Leftrightarrow \quad \forall c \in \mathcal{C},\ \neg(\mathcal{K} \overset{c}{\succ}) \tag{82}$$

Proposition 12. *$\succ$ is a well-founded strict partial order, i.e. every non-empty set of cell complexes Ω has a minimal element.*

Proof. Let $n_m = \min\{|\mathcal{C}| \mid \mathcal{C} \in \Omega\}$ (exists because $<$ on $\mathbb{N}$ is well-founded), and $\mathcal{C}_m$ such as $|\mathcal{C}_m| = n_m$. By definition of n_m, we have $\neg(|\mathcal{C}'| < |\mathcal{C}_m|)$ for each $\mathcal{C}'$ in Ω, thus $\neg(\mathcal{C}_m \succ \mathcal{C}')$, thus $\mathcal{C}_m$ is a minimal element of Ω. □

Corollary 2. *There are no infinite descending chains:*

$$\mathcal{C}_0 \succ \mathcal{C}_1 \succ \cdots \succ \mathcal{C}_k \succ \cdots \tag{83}$$

Proof. Well-founded strict partial orders do not have infinite descending chains. □

Corollary 3. *Let X be a topological space admitting a cell complex $\mathcal{K}$. Then there exists $\mathcal{K}_m$ decomposing X such that $\mathcal{K}_m$ is minimal.*

Proof. Let Ω be the set of all cell complexes decomposing X. It is non-empty since $\mathcal{K} \in \Omega$, thus there exists $\mathcal{K}_m$ minimal using Proposition 12. □

Finally, we conclude this section by defining the weak versions of the simplification binary operators.

Weak atomic simplification Let $\mathcal{K} = (X, \mathcal{C})$ be a cell complex and $c \in \mathcal{C}$. We conveniently write $\mathcal{K} \overset{c}{\succcurlyeq} \mathcal{K}'$ to define $\mathcal{K}'$ as being equal to:

- $\mathcal{K}$ if $\mathcal{K}$ cannot be simplified at c.
- the atomic simplification of $\mathcal{K}$ at c otherwise.

Weak anonymous atomic simplification We define the relation $\overset{\bullet}{\succcurlyeq}$ to be the reflexive closure of $\overset{\bullet}{\succ}$:

$$\mathcal{K} \overset{\bullet}{\succcurlyeq} \mathcal{K}' \quad \Leftrightarrow \quad \begin{cases} \mathcal{K} = \mathcal{K}', \text{or} \\ \mathcal{K} \overset{\bullet}{\succ} \mathcal{K}' \end{cases} \tag{84}$$

Weak simplification We define the relation $\succcurlyeq$ to be the reflexive closure of $\succ$:

$$\mathcal{K} \succcurlyeq \mathcal{K}' \quad \Leftrightarrow \quad \begin{cases} \mathcal{K} = \mathcal{K}', \text{or} \\ \mathcal{K} \succ \mathcal{K}' \end{cases} \tag{85}$$

9.2 Equivalence between cell complexes

Bi-directional atomic simplification We define the relation $\overset{\bullet}{\leftrightarrow}$ to be the symmetric closure of $\overset{\bullet}{\succ}$:

$$\mathcal{K} \overset{\bullet}{\leftrightarrow} \mathcal{K}' \quad \Leftrightarrow \quad \begin{cases} \mathcal{K} \overset{\bullet}{\succ} \mathcal{K}', \text{or} \\ \mathcal{K}' \overset{\bullet}{\succ} \mathcal{K} \end{cases} \tag{86}$$

Bi-directional simplification We define the relation $\leftrightarrow$ to be the transitive closure of $\overset{\bullet}{\leftrightarrow}$.

Proposition 13. *$\mathcal{K} \leftrightarrow \mathcal{K}'$ iff a finite sequence of atomic simplification or de-simplification transforms $\mathcal{K}$ into $\mathcal{K}'$:*

$$\mathcal{K} \leftrightarrow \mathcal{K}' \quad \Leftrightarrow \quad \mathcal{K} \overset{\bullet}{\leftrightarrow} \mathcal{K}_1 \overset{\bullet}{\leftrightarrow} \cdots \overset{\bullet}{\leftrightarrow} \mathcal{K}_{k-1} \overset{\bullet}{\leftrightarrow} \mathcal{K}' \tag{87}$$

Proof. Same as Proposition 9. □

Equivalence relation We define the relation $\equiv$ to be the reflexive closure of $\leftrightarrow$:

$$\mathcal{K} \equiv \mathcal{K}' \quad \Leftrightarrow \quad \begin{cases} \mathcal{K} = \mathcal{K}', \text{ or} \\ \mathcal{K} \leftrightarrow \mathcal{K}' \end{cases} \tag{88}$$

It is an equivalence relation, since it is symmetric, transitive and reflexive. Note that it is important to take the transitive closure after the symmetric closure: two decompositions $\mathcal{C}$ and $\mathcal{C}'$ can have the same number of cells (and thus we have neither $\mathcal{C} \succ \mathcal{C}'$ nor $\mathcal{C}' \succ \mathcal{C}$), but still could be obtained via a de-simplification followed by a simplification: $\mathcal{C} \prec \mathcal{C}'' \succ \mathcal{C}'$. In fact, we will see later that it is always possible.

Corollary 4. *Two cell complexes are equivalent if and only if they are equal or obtained from one another via a finite sequence of atomic simplification or de-simplification:*

$$\mathcal{K} \equiv \mathcal{K}' \quad \Leftrightarrow \quad \begin{cases} \mathcal{K} = \mathcal{K}', \text{ or} \\ \mathcal{K} \overset{\bullet}{\leftrightarrow} \mathcal{K}_1 \overset{\bullet}{\leftrightarrow} \cdots \overset{\bullet}{\leftrightarrow} \mathcal{K}_{k-1} \overset{\bullet}{\leftrightarrow} \mathcal{K}' \end{cases} \tag{89}$$

Proof. Combine Proposition 13 and definition of $\equiv$. □

Proposition 14. *Let $\mathcal{K} = (X, \mathcal{C})$ and $\mathcal{K}' = (X', \mathcal{C}')$ be two cell complexes. Then we have:*

$$\mathcal{K} \equiv \mathcal{K}' \;\Rightarrow\; X = X' \tag{90}$$

Proof. We have $\mathcal{K} \overset{\bullet}{\succcurlyeq} \mathcal{K}' \Rightarrow X = X'$ directly from the definition of $\overset{\bullet}{\succcurlyeq}$, which implies that $X = X'$ whenever $\mathcal{K}$ and $\mathcal{K}'$ are related by any of a closures of $\overset{\bullet}{\succcurlyeq}$ defined above. □

Proposition 15. *Let $\mathcal{K}$ and $\mathcal{K}'$ be two cell complexes. Then we have:*

$$\mathcal{K} \succcurlyeq \mathcal{K}' \;\Rightarrow\; \mathcal{K} \equiv \mathcal{K}' \tag{91}$$

Proof. $\mathcal{K} \succcurlyeq \mathcal{K}' \;\Rightarrow\; (\mathcal{K} = \mathcal{K}' \text{ or } \mathcal{K} \succ \mathcal{K}')$. In the first case, $\mathcal{K} \equiv \mathcal{K}'$ since $\equiv$ is reflexive. In the second case, we have $\mathcal{K} \overset{\bullet}{\succ} \cdots \overset{\bullet}{\succ} \mathcal{K}'$, hence $\mathcal{K} \overset{\bullet}{\leftrightarrow} \cdots \overset{\bullet}{\leftrightarrow} \mathcal{K}'$, hence $\mathcal{K} \equiv \mathcal{K}'$. □

Conjecture 1. *Let X be a topological space, and $\mathcal{K} = (X, \mathcal{C})$ and $\mathcal{K}' = (X, \mathcal{C}')$ be two cell complexes decomposing X. Then they admit a common "ancestor", i.e.:*

$$\exists \mathcal{K}'' = (X, \mathcal{C}'') \text{ such that } \begin{cases} \mathcal{K}'' \succcurlyeq \mathcal{K}, \text{ and} \\ \mathcal{K}'' \succcurlyeq \mathcal{K}' \end{cases} \tag{92}$$

We expect that a proof can be achieved by explicitely constructing $\mathcal{C}''$ as intersections of cells in $\mathcal{C}$ with cells in $\mathcal{C}'$. Then, we would have to prove that $\mathcal{K}'' = (X, \mathcal{C}'')$ is a cell complex. The following of this Section 9.2 (but no other sections) assumes that this conjecture is true.

Theorem 2 (Equivalence Theorem)**.** *Let $\mathcal{K} = (X, \mathcal{C})$ and $\mathcal{K}' = (X', \mathcal{C}')$ be two cell complexes. Then we have:*

$$\mathcal{K} \equiv \mathcal{K}' \;\Leftrightarrow\; X = X' \tag{93}$$

In other words: the equivalent cell complexes are exactly those decomposing the same space.

Proof. We have already $\mathcal{K} \equiv \mathcal{K}' \Rightarrow X = X'$. Let X be a topological space and $\mathcal{K} = (X, \mathcal{C})$ and $\mathcal{K}' = (X, \mathcal{C}')$ be two cell complexes decomposing X. Let $\mathcal{K}'' = \text{CommonAncestor}(\mathcal{K}, \mathcal{K}')$. We have $\mathcal{K}'' \succcurlyeq \mathcal{K}$ and $\mathcal{K}'' \succcurlyeq \mathcal{K}'$, thus $\mathcal{K}'' \equiv \mathcal{K}$ and $\mathcal{K}'' \equiv \mathcal{K}'$, thus $\mathcal{K} \equiv \mathcal{K}'$ by transitivity. □

Corollary 5. *Let X be a topological space, and $\mathcal{K} = (X, \mathcal{C})$ and $\mathcal{K}' = (X, \mathcal{C}')$ be two cell complexes decomposing X. Then it is possible to transform $\mathcal{K}$ into $\mathcal{K}'$ via a finite sequence of atomic simplification or de-simplification.*

Proof. We simply combine the result of the equivalence theorem with Corollary 4. □

9.3 Uniqueness of minimal PCS complex

We have seen, for arbitrary dimension, that if X admits a cell complex $\mathcal{K} = (X, \mathcal{C})$, then it also admits one $\mathcal{K}_m$ that is minimal. In fact, regardless whether Conjecture 1 is true or false, it also admits one which is both minimal and equivalent to $\mathcal{K}$ (i.e., that can be obtained from a finite sequence of simplification or de-simplification):

Proposition 16 (Minimal decomposition)**.** *Let $\mathcal{K} = (X, \mathcal{C})$ be a cell complex. Then there exists a minimal cell complex $\mathcal{K}_m$ such that $\mathcal{K} \equiv \mathcal{K}_m$.*

Proof. We have seen that there exists no infinite decreasing sequences of cell complexes. Hence, by defining $\mathcal{K}_0 = \mathcal{K}$, we can recursively define $\mathcal{K}_{i+1}$ by $\mathcal{K}_i \succcurlyeq^{c_i} \mathcal{K}_{i+1}$ while there exists a cell $c_i \in \mathcal{C}_i$ such that $\mathcal{K}_i$ can be atomically simplified at c_i. This procedure necessarily stops, and then there exists $N \in \mathbb{N}$ such that $\mathcal{K}_N$ cannot be atomically simplified at any cell and $\mathcal{K}_0 \succcurlyeq^{c_0} \cdots \succcurlyeq^{c_{N-1}} \mathcal{K}_N$. Thus, $\mathcal{K}_N$ minimal and $\mathcal{K} \succcurlyeq \mathcal{K}_N$, thus $\mathcal{K}_N$ minimal and $\mathcal{K} \equiv \mathcal{K}_N$. □

In the case of the dimension two or less, we conjecture that this minimal decomposition is unique, which would imply that performing simplifications in any order until it is not possible anymore leads necessarily to this unique minimal decomposition.

Conjecture 2 (Unique minimal decomposition)**.** *Let X be a topological space, and $\mathcal{K} = (X, \mathcal{C})$ and $\mathcal{K}' = (X, \mathcal{C}')$ be two minimal PCS complexes decomposing X. Then $\mathcal{K} = \mathcal{K}'$.*

10 Conclusion

Using a formalism borrowing concepts from algebraic topology and graph theory, we have introduced PCS complexes, and their combinatorial counterpart abstract PCS complexes. Thanks to this formal framework, we were able to carefully analyze the underlying pointset topology that vector graphics complexes represent.

Because VGC faces can be interpreted as non-orientable surfaces, or surfaces with strictly positive genus, a wide spectrum of non-equivalent cut algorithms may be applied (cf. Figure 30 and 31). This brings VGCs to a significantly richer category of topological spaces than planar maps, whose assumption of planarity implies strong properties which do not hold true for VGCs. For instance, one cannot assume that cutting a face with an edge starting and ending at the same cycle disconnects the face (cf. Figure 33). Which algorithm should be applied in a given situation is an ill-defined problem, since we do not have access to the actual non-planar topological space but only to a non-injective 2D projection of this space. Therefore, one has to design heuristics based on the geometry of edges to take a decision. However, no "perfect heuristic" exists, since there may not even be consensus among humans on which algorithm is preferable, due to the interpretative nature of the 2D depiction [Durand, 2002].

With this report, we hope to convince the reader that despite the apparent simplicity of 2D vector graphics, allowing both shared edges and self-intersections is not an easy task, which leads to complex and not well-studied topological objects. This might explain why despite several decades of research in the field, no data-structure existed combining the convenience of topological manipulation *and* self-intersections. The vector graphics complex is a first attempt, and we foresee that it will lead to various interesting future research, with a wide range of applications. More particularly, a very important but still open problem is how to render correctly self-overlapping faces (e.g., [Wiley and Williams, 2006]), and how to render correctly junctions between incident edges and faces.

Figure 35: The figure on the left has been achieved using 9 independent Bézier curves, depicted on the right from back to front. Editing such a figure is extremely time-consuming.

To conclude this report with a fun fact, we note that most of the figures in it have been created using the latest version of Inkscape, chosen for its nice LaTeX export, and powerful yet user-friendly interface overall. However, Inkscape obviously does not support the VGC (yet!), while most figures that we had to create are typically the kind that would be much easier to create with the VGC. Therefore, a lot of tricks were needed to achieve them, typically decomposing a single "PCS face" into many independent pieces (cf. Figure 35). The slightest edit was very time-consuming since all pieces should be modified for their geometry to coincide, which was on the one hand very frustrating, but on the other hand a very convincing proof by experience of the relevance of the VGC, that we could not resist to share.

References

[Asente et al., 2007] Asente, P., Schuster, M., and Pettit, T. (2007). Dynamic planar map illustration. *ACM Trans. Graph.*, 26(3):30:1–30:10.

[Baudelaire and Gangnet, 1989] Baudelaire, P. and Gangnet, M. (1989). Planar maps: An interaction paradigm for graphic design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '89, pages 313–318, New York, NY, USA. ACM.

[Dalstein et al., 2014] Dalstein, B., Ronfard, R., and van de Panne, M. (2014). Vector graphics complexes. *ACM Trans. Graph.*, 33(4).

[De Floriani et al., 2003] De Floriani, L., Morando, F., and Puppo, E. (2003). Representation of Non-manifold Objects Through Decomposition into Nearly Manifold Parts. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, SM '03, pages 304–309, New York, NY, USA. ACM.

[Dehn and Heegaard, 1907] Dehn, M. and Heegaard, P. (1907). Analysis situs. In *Enzyklopädie der Math. Wiss. III.1.1*, pages 153–220.

[Durand, 2002] Durand, F. (2002). An invitation to discuss computer depiction. In *Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering*, NPAR '02, pages 111–124, New York, NY, USA. ACM.

[Edelsbrunner and Harer, 2010] Edelsbrunner, H. and Harer, J. (2010). *Computational Topology: An Introduction*. Applied mathematics. American Mathematical Society.

[Gale, 1987] Gale, D. (1987). The Classification of 1-Manifolds: A Take-Home Exam. *The American Mathematical Monthly*, 94(2):170–175.

[Hatcher, 2001] Hatcher, A. (2001). *Algebraic Topology*.

[Lee, 2011] Lee, J. M. (2011). *Introduction to Topological Manifolds*. Springer New York.

[Rossignac and O'Connor, 1989] Rossignac, J. and O'Connor, M. (1989). *SGC: A Dimension-independent Model for Pointsets with Internal Structures and Incomplete Boundaries*. Research report. IBM T.J. Watson Research Center.

[Weiler, 1985] Weiler, K. (1985). Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments. *IEEE Computer Graphics and Applications*, 5(1):21–40.

[Wiley and Williams, 2006] Wiley, K. and Williams, L. R. (2006). Representation of interwoven surfaces in 2 1/2 D drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 65–74, New York, NY, USA. ACM.