Constraint-preserving labeled graph transformation for topology-based geometric modeling

Thomas Bellet[a], Agnès Arnould[b], Pascale Le Gall[a]

[a] *MICS, CentraleSupélec, France*
[b] *XLIM UMR CNRS 7252, University of Poitiers, France*

Abstract

As labeled graphs are particularly well adapted to represent objects in the context of topology-based geometric modeling, graph transformation theory is an adequate framework to implement modeling operations and check their consistency. In this article, objects are defined as a particular subclass of labeled graphs in which arc labels encode their topological structure (*i.e.* cell subdivision: vertex, edge, face, etc.) and node labels encode their embedding (*i.e.* relevant data: vertex positions, face colors, volume density, etc.). Object consistency is therefore defined by labeling constraints. To define modeling operations, we define a class of graph transformation rules dedicated to embedding computations. Dedicated graph transformation variables allow us to access the existing embedding from the underlying topological structure (*e.g.* collecting all the points of a face) in order to compute the new embedding using user-provided functions (*e.g.* compute the barycenter of several points). To ensure the safety of the defined operations, we provide syntactic conditions on rules that preserve the object consistency constraints.

Keywords: DPO graph transformation, topology-based geometric modeling, graph transformation with variables, labeled graphs, generalized maps, consistency preservation, static analysis, algebraic data types.

1. Introduction

In the early 1970s, the concept of graph transformation became of interest in computer science. Derived from string and tree rewriting techniques, this rule-based approach offers a very natural way to describe complex transformations on an intuitive level. For example, anyone observing the transformation given in Figure 1 easily understands the change made in the company organization: the new CEO hires a plant director who is in charge of manufacturing and purchase. Nowadays, thanks to their expressiveness, graph transformations have applications in many areas such as software engineering [1], concurrent and distributed systems [2], visual modeling [3] or database design [4].

Our interest concerns the application of graph transformations to topology-based geometric modeling [5], a field that deals with the representation and manipulation of objects according to their topological structure (cell subdivision) and their embedding (other types of information attached to their topological cells). As topological structures can be represented as a particular class of graphs, the use of graph transformations to define modeling operations have already been proposed in the past [6, 7, 8].

In this article, we propose a generic graph transformation approach that allows the implementation of modeling operations of any application domain. Indeed, object constructions and transformations depend on the targeted domain; *e.g.* add or remove some matter to sculpt (arts), add a window on a wall (architecture), rotate a gear (engineering), combine partial scans (archaeology), etc. The definition and implementation of such operations are always the longest parts of a modeler development. Every single operation has to be designed, coded, debugged and optimized. It is even common nowadays that modelers include a cleaning post-treatment function to fix inconsistencies in transformed objects when operations are too complex to maintain object consistency along the operation code. As this article will show, this need for a reliable and flexible framework to define and implement modeling operations can be efficiently addressed by graph transformations.

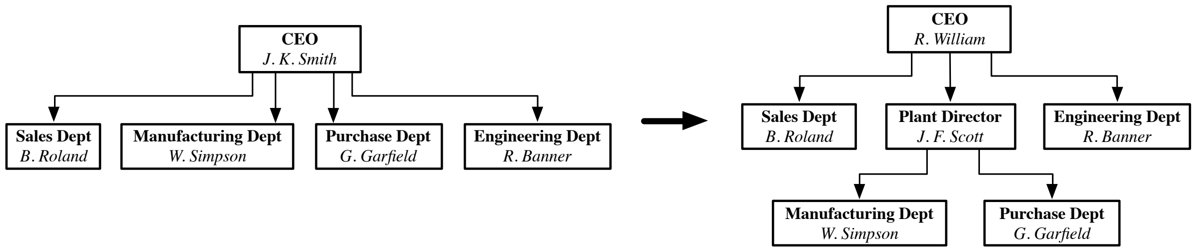
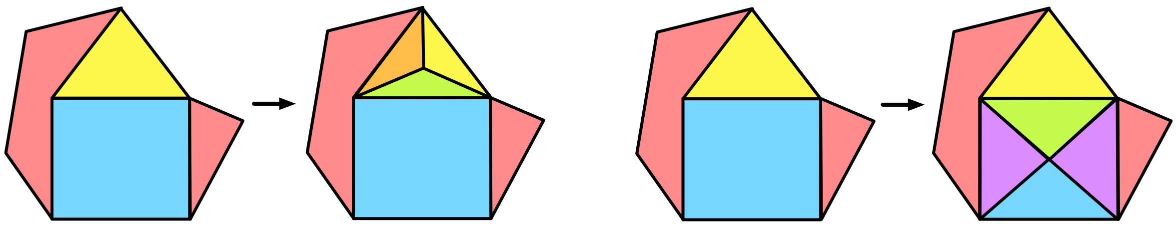


Figure 1: An example of graph transformation expressiveness

Let us take a first operation example with the face triangulation of colored 2D objects given in Figure 2. The topological structure of the object under transformation contains four faces (two triangles, a square and a pentagon) glued all together, while the embedding associates a color to each face. Both the topological structure and the embedding are transformed by the face triangulation. Topologically, the face is subdivided into triangles, while from the embedding point of view, colors are computed for the new faces as the mix of the subdivided face color and the neighboring face color.



(a) (b)

Figure 2: A modeling operation example

Using the topological model of generalized maps [5, 9, 10], objects are defined as a particular subclass of labeled graphs in which arc labels encode their topological structure and labeling constraints define their consistency. The implementation of modeling operations such as the face triangulation with graph transformations requires some dedicated features of graph transformations. In [11, 6, 12], we previously introduced dedicated rule variables to generically handle the topological aspect of modeling operations, *i.e.* to automatically compute the topological transformation depending on the cell size. For example, the face subdivision of the triangulation given in Figure 2 can be abstracted with a single rule scheme that will create three faces in the case of a triangle (Figure 2(a)), four faces in the case of a square (Figure 2(b)), or n faces in the case of an n-sided polygon.

This article addresses the embedding aspect of modeling operation. Considering a representation of embedded generalized maps as labeled graphs introduced in [13] and in which node labels and associated labeling constraints encode object embedding, we will study under which conditions relabeling graph transformations of [14] preserve the embedding consistency of the object under modeling. For example, in the case of the face triangulation of Figure 2, we will ensure that by construction, after the application of the corresponding rules, all nodes of a same face end up labeled with the same color. Based on [15], we will then introduce a rule-based language that allows the new embedding to be generically computed (*e.g.* to compute the face triangulations of Figure 2 for any colored object). Using dedicated graph transformation variables and terms, the resulting rule schemes will provide the means of both accessing the existing embedding through the topological structure and applying functions provided by the user with the embedding data type. For example, the triangulation of Figure 2(a) will be defined by a rule scheme in which the colors of the created faces are computed by applying the user function “mix of two colors” to the subdivided face color and the respective adjacent face colors. As the safety of this user-oriented language is as essential as its expressiveness, we will provide syntactic conditions to guide rule scheme design and ensure object consistency preservation.

This paper is organized as follows. First, Section 2 presents the labeled graph transformation fundamentals of our work. In particular, we describe the relabeling graph transformation introduced by [14] that will allow us to modify objects. We also present the use of variables in graph transformations as introduced by [15]. The context of topological-based geometric modeling is then presented in Section 3. We focus on the topological model of generalized maps [9] and we give the conditions from [6] under which graph transformations preserve the topological consistency. Section 4 then similarly presents our embedded version of generalized maps and conditions under which basic graph transformations preserve the embedding consistency. The next three sections focus on the rule-based language dedicated to embedding modifications. Section 5 introduces the rule scheme syntax, in particular terms that allow generic computation of new embedding values from existing ones in the object (*e.g.* to mix two unspecified face colors). Section 6 presents the rule scheme application, especially how schemes are instantiated to propagate minimally defined modifications (*e.g.* automatically change the color of all nodes of one face). Section 7 provides syntactic conditions on rule schemes to ensure embedding consistency preservation. Finally, Section 8 and 9 respectively present related work and concluding remarks.

2. Graph transformations

Graphs are non-linear structures, defined by a set of objects, usually called vertices or nodes, and a set of links between these objects, usually called edges or arcs. To avoid confusion with the specific vocabulary of geometric modeling, we will subsequently prefer to use the word "node" rather than the word "vertex", and the word "arc" rather than "edge" when refering to graph elements. Graphs are often depicted in a diagrammatic form with dots or circles to represent nodes and lines or curves to represent arcs between nodes, directed or undirected. Graph transformations commonly refer to rule-based languages designed to manipulate graphs. Among all graph transformation approaches, we choose the so-called double-pushout approach (or DPO) [16, 17, 18].

2.1. Double-pushout graph transformation

The DPO approach has been referred as algebraic since a transformation is expressed using two gluing diagrams defined in terms of category theory. More precisely, these diagrams are pushouts in the category of graphs and graph morphisms. In order to make the presentation more intuitive, let us consider the simple example of a DPO graph transformation given in Figure 3. The rule is given at the top of Figure 3 by the three graphs L, K and R and by the two graph inclusions[1] $L \hookleftarrow K \hookrightarrow R$. In this example, nodes of graphs are identified by letters (a, b or c) while nodes and arcs are labeled by numbers (1, 2, 3, 4 or 5). Intuitively, the left-hand side of the rule L is the pattern to transform, the right-hand side R is the transformed pattern, and the interface K is the preserved part common to both L and R. The graph morphism $L \to G$ allows the matched pattern (graph L) to be identified inside the graph under transformation (graph G) (in Figure 3, the match morphism coincides with the inclusion).

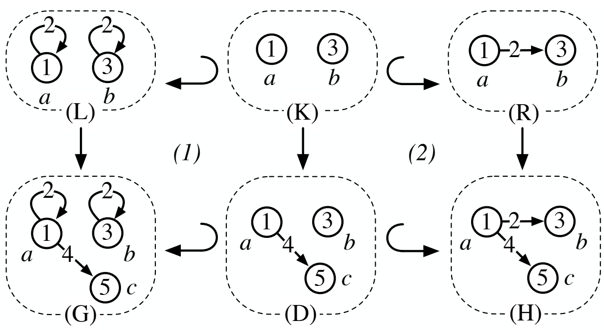


Figure 3: DPO graph transformation

[1]In full generally, one can consider standard graph morphisms instead of considering only inclusions.

In the first pushout (1), all elements (nodes and arcs) in L that are not in K are removed in D. In the example, the two arcs that loop on nodes a and b are removed: graph D results from the removing of loops on both nodes a and b in graph G. In the second pushout (2), all elements of R that are not already in K are added while all elements in K are preserved. In the example, an arc is added between the two preserved nodes a and b (graph H). When the match $L \to G$ meets some conditions, then the transformation is well defined, and the double-pushout construction defines a single graph H (up to graph isomorphism), that is the result of the application of the graph transformation $L \hookleftarrow K \hookrightarrow R$ through the match morphism $L \to G$.

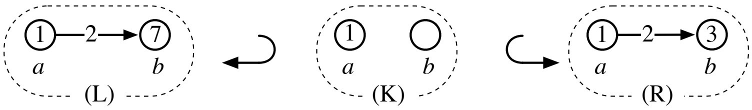


Figure 4: A relabeling rule

In classical DPO transformations, graph morphisms have to preserve both arc and nodes labels, and therefore prevent relabeling. Note that the relabeling of an arc can still be achieved by removing it while adding an arc with the new label between the same source and target nodes, but this not the case for the relabeling of a node. Consequently, we prefer the DPO approach of [14] that considers partially labeled graphs and therefore authorizes relabeling. Let us take the example of the relabeling rule of Figure 4. In the left-hand side L, node b is initially labeled by 7 while it is relabeled by 3 in the right-hand side R, and therefore unlabeled in the interface K. Note that the two morphisms $K \hookrightarrow L$ and $K \hookrightarrow R$ on partially labeled graphs L, K, R do not preserve labeling, in particular the undefined label of b.

We then present the main definitions and results of [14] on partially labeled graph transformations.

2.2. Graph transformations on partially labeled graphs

A *partially labeled graph* $G = (V_G, E_G, s_G, t_G, l_{G,V}, l_{G,E})$ consists of two finite sets V_G and E_G of *nodes* and *arcs*, two source and target functions $s_G, t_G : E_G \to V_G$, and two partial labeling functions[2] $l_{G,V} : V_G \to \mathcal{C}_V$ and $l_{G,E} : E_G \to \mathcal{C}_E$, where $\mathcal{C}_V$ and $\mathcal{C}_E$ are fixed sets of node and arc labels. We say that G is *totally labeled* if $l_{G,V}$ and $l_{G,E}$ are total functions. A path in a graph G is a sequence $e_1...e_k$ of arcs of G with $1 \leq k$, such $t_G(e_i) = s_G(e_{i+1})$ for each $1 \leq i < k$. $s_G(e_1)$ and $t_G(e_k)$ are respectively called the *path source* and the *path target* and the word $l_{G,E}(e_1)...l_{G,E}(e_k)$ is called the *path label*. Moreover, if $s_G(e_1) = t_G(e_k)$, the path is called a *cycle*. Thereafter, partially labeled graphs are simply called *graphs*.

A *graph morphism* $g : G \to H$ between two graphs G and H consists of two functions $g_V : V_G \to V_H$ and $g_E : E_G \to E_H$ that preserve sources, targets and labels, that is, $s_H \circ g_E = g_V \circ s_G$, $t_H \circ g_E = g_V \circ t_G$, and $l_H(g(x)) = l_G(x)$ for all[3] x in $Dom(l_G)$. A morphism g is *injective* (resp. *surjective*) if g_V and g_E are injective (resp. surjective), and is an *isomorphism* if it is injective, surjective and preserves undefinedness[4]. In the latter case, G and H are *isomorphic*. Furthermore, we call g an *inclusion* if $g(x) = x$ for all x in G. An inclusion is identified with the symbol $\hookrightarrow$ (or the symbol $\hookleftarrow$ if the target graph is introduced first). Finally, morphism composition is defined componentwise as function compositions.

Definition 1 (Rule). *A rule $r : L \hookleftarrow K \hookrightarrow R$ consists of two inclusions $K \hookrightarrow L$ and $K \hookrightarrow R$ such that:*
(1) for all $x \in L$, $l_L(x) = \bot$ implies $x \in K$ and $l_R(x) = \bot$,
(2) for all $x \in R$, $l_R(x) = \bot$ implies $x \in K$ and $l_L(x) = \bot$. ◀

Regarding [14], our rules are simplified since they are built with two inclusions for both sides instead of only one for the left-hand side. We call L the *left-hand side*, R the *right-hand side* and K the *interface* of r. Note that conditions (1) and (2) are trivially satisfied when L and R are totally labeled.

---

[2]Given two sets A and B, a partial function $f : A \to B$ is a function from subset A' of A to B. The set A' is the domain of f and is denoted by $Dom(f)$. We say that $f(x)$ is undefined, and write $f(x) = \bot$, if x is in $A - Dom(f)$.

[3]To simplify, we amalgamate nodes and arcs in statements that hold for both sets, by omitting the indices E and V.

[4]i.e. if $l_H(g(x)) = \bot$ for all x in $G \backslash Dom(l_G)$

A diagram of graph morphisms, as Figure 5(a), is a *pushout* if (i) $K \to R \to H = K \to D \to H$ and (ii) for every pair of graph morphisms $(R \to H', D \to H')$ with $K \to R \to H' = K \to D \to H'$, there is a unique morphism $H \to H'$ such that $R \to H' = R \to H \to H'$ and $D \to H' = D \to H \to H'$. The same diagram is a *pullback* if property (i) holds and (iii) if for every pair of graph morphisms $(K' \to R, K' \to D)$ with $K' \to R \to H = K' \to D \to H$, there is a unique morphism $K' \to K$ such that $K' \to R = K' \to K \to R$ and $K' \to D = K' \to K \to D$. A pushout is *natural* if it is also a pullback.

$$\begin{tikzcd}
K \arrow[r] \arrow[d] & R \arrow[d] \\
D \arrow[r] & H
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
L \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & K \arrow[l, hook'] \arrow[r, hook] \arrow[d] \arrow[dr, phantom, "(2)"] & R \arrow[d] \\
G & D \arrow[l, hook'] \arrow[r, hook] & H
\end{tikzcd}$$

(b)

Figure 5: A diagram and a direct derivation

Definition 2 (Direct derivation). *A direct derivation from a graph G to a graph H via a rule $r : L \hookleftarrow K \hookrightarrow R$ consists of two natural pushouts as Figure 5(b), where $m : L \to G$, called the match morphism, is injective. We write $G \Rightarrow_{r,m} H$ if there exists such a direct derivation.* ◀

In [14], authors studied under which conditions usual constructions in the category of totally labeled graphs such as pushouts, pullbacks or direct derivations can be transposed into the category of partially labeled graphs. In particular, properties concerning direct derivations are similar to those on totally labeled graphs. A match morphism $m : L \to G$ satisfies the *dangling condition* with respect to the inclusion $L \hookleftarrow K$, if no node in $m(L)\backslash m(K)$ is incident to an arc in $G\backslash m(L)$. Given a rule $r : L \hookleftarrow K \hookrightarrow R$ and a match morphism $m : L \to G$, there exists a direct derivation as in Figure 5(b) if and only if m satisfies the dangling condition. Moreover, in this case, D and H are unique up to isomorphism, and H is totally labeled if and only if G is totally labeled.

2.3. Graph transformation with variables

To meet the various application needs, graph transformations have been enriched with variables to make them generic. Intuitively, rules with variables describe as many concrete rules as there are possibilities to instantiate variables with concrete elements. Variable types have various purposes. For example, attribute variables allow label computations [19] or labeling constraints [20, 21], while graph variables [22, 23] and hyperedge variables [24, 25] allow structural transformations.

In this article, we will use the framework for graph transformation with variables introduced in [15]. We briefly describe its main elements: rule scheme, instantiation, and rule application.

Rule scheme. The sets $\mathcal{C}_V$ and $\mathcal{C}_E$ of node and arc labels are extended by a set X of *variable names*. Graphs built over X are called *graph schemes*. Then a *rule scheme* is a rule $r : L \hookleftarrow K \hookrightarrow R$ where L, K, and R are graph schemes. The *kernel* $\underline{G}$ of G is the graph obtained by removing all labels that contain a variable occurrence.

Instantiation. A *substitution function* σ specifies how variable names occurring in a rule scheme are substituted. The *instanciation* of a rule scheme $r : L \hookleftarrow K \hookrightarrow R$ according to σ defines a particular *rule instance* $r_\sigma : L_\sigma \hookleftarrow K_\sigma \hookrightarrow R_\sigma$ in which L_σ, K_σ and R_σ result from the substitution of variables in L, K, R by their images by σ. Note that r_σ is a rule without variables as defined in Definition 1.

Rule application. Let G be a graph, and $r : L \hookleftarrow K \hookrightarrow R$ be a rule scheme.

1. Identify a *kernel match* $\underline{m} : \underline{L} \to G$ of the kernel $\underline{L}$ of L in G (if it exists);
2. If possible, find a substitution σ such that there exists a morphism $m : L_\sigma \to G$ extending $\underline{m}$;
3. Construct the instance $r_\sigma : L_\sigma \hookleftarrow K_\sigma \hookrightarrow R_\sigma$ and apply r_σ to get the direct derivation $G \Rightarrow_{r_\sigma,m} H$.

In [15], this framework is used for three types of variables with specific purposes: attribute variables, clone variables and graph variables. In [6, 26], we introduced another type of variables that will be presented in Section 3.3. Called topological variables, they allow to generically define topological structure transformations (*e.g.* for the face triangulation of Figure 2, define the face subdivision independently of the face configuration: triangle, square, polyhedron, etc.).

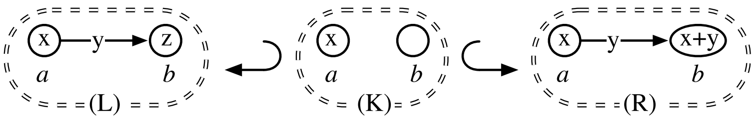


Figure 6: A rule scheme with attribute variables

For the embedding aspect of geometric operations, we first thought that the *attribute variables* of [15] were expressive enough. Their usage is illustrated by the rule scheme of Figure 6. For the matched arc, the new label of the target node becomes the sum of the initial label of the source node and of the initial label of the arc. Intuitively, the application of this rule scheme to a graph allows one to deduce a substitution of attributed variables occurring on the left-hand side L, and to compute the new labels by evaluation of the expressions of the right-hand side R. Note that the substitution $\sigma = \{x \mapsto 1, y \mapsto 2, z \mapsto 7\}$ would allow to derive the previous rule of Figure 4, according that $+$ is evaluated as the usual addition operation.

But attribute variables do not exactly fit our usage as we need to access the node labels through the node names to allow topological structure traversal (*e.g.* to access the adjacent face color in the case of the colored triangulation given in the introduction). In the sequel, we will introduce dedicated variables, but, roughly speaking, they will come down to attribute variable indirections in the most basic cases. Note also that by convention, rule schemes will be identified by dotted double lines circling their L, K and R graphs (see Figure 6) while single dotted lines are used for rule instance graphs (see Figure 4).

2.4. Data types

In our setting, objects will be modeled by labeled graphs whose nodes are labeled by geometric or dedicated data (thereafter referred to as *embedding*) such as the color of a face or the 2D position of a point. These data are clearly typed and provided with functions to perform computations on them.

In addition, as a same embedding value can appear multiple times in an object (*e.g.* in the transformed object of Figure 2, two faces have the same color), we need to identify these multiple occurrences when collecting object embedding values. We therefore consider for each type τ, the type $\tau^\bullet$, *multiset* of elements of type τ. A multiset may be viewed as a function that associates its multiplicity (a natural number) to each element. We use the following notation: $[\![]\!]$ for the empty multiset (of any type $\tau^\bullet$), $[\![a_1, \dots a_p]\!]$ for a multiset with p occurrences of elements of type τ. For example, the multiset that contains the element A with the multiplicity 1, the element B with the multiplicity 2, and where all other elements are of multiplicity 0, is denoted $[\![A, B, B]\!]$. Similarly, the multiset of face colors of the transformed object of Figure 2 is denoted $[\![\bullet, \bullet, \bullet, \bullet]\!]$.

We then present the main elements of term construction and evaluation.

Signature. A data type signature $\Omega = (S, F)$ consists of a set S of *type* names and a family of *function* names provided with a profile on $S \cup S^\bullet$ where $S^\bullet = \{s^\bullet \mid s \in S\}$ is the set of multisets over types in S. A function name f provided with a profile $s_1 \dots s_{m+1}$ with $s_i \in S \cup S^\bullet$ for $i \in 1..m+1$ is denoted $f : s_1 \times \dots \times s_m \to s_{m+1}$.

Terms. For an S-indexed family $X = \coprod_{s \in S} X_s$ of variables, the set $T_\Omega(X) = \coprod_{s \in S \cup S^\bullet} T_\Omega(X)_s$ of terms over Ω is the least set satisfying:

- for all variables x in X_s, $x \in T_\Omega(X)_s$;
- for all function names $f : s_1 \times \dots \times s_m \to s_{m+1}$ in F, for all terms $t_1 \in T_\Omega(X)_{s_1}, \dots, t_m \in T_\Omega(X)_{s_m}$, then $f(t_1, \dots, t_m) \in T_\Omega(X)_{s_{m+1}}$.

We note $t : s$ a term t in $T_\Omega(X)_s$.

Algebra. An Ω-algebra $\mathcal{A}$ consists of an S-indexed family of nonempty sets $\coprod_{s\in S} A_s$ and for each $f : s_1 \times \ldots \times s_m \to s_{m+1}$ in F, of a function $f^A : A_{s_1} \times \ldots \times A_{s_m} \to A_{s_{m+1}}$ where for $s_i \in S^\bullet$, that is $s_i = s^\bullet$ for some s in S, $A_{s_i} = \{[\![a_1, \ldots, a_p]\!] \mid 0 \leq p, \forall k \in 0..p, a_k \in A_s\}$.

Evaluation. For $\sigma = \coprod_{s\in S} \sigma_s$ an S-indexed family of assignments $\sigma_s : X_s \to A_s$, the evaluation $\sigma : T_\Omega(X)_s \to A_s$ of a term $t : s$ is defined as:

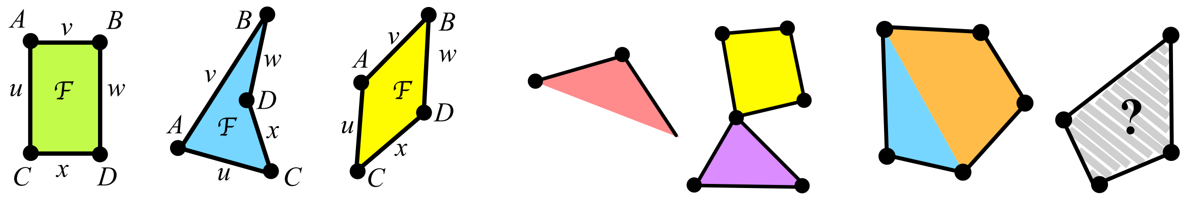
- for all variables x in X_s, $\sigma(x) = \sigma_s(x)$;
- for all function names $f : s_1 \times \ldots \times s_m \to s_{m+1}$ in F and all terms $t_1 : s_1, \ldots, t_m : s_m$, $\sigma(f(t_1, \ldots, t_m)) = f^A(\sigma(t_1), \ldots, \sigma(t_m))$.

In order to design a modeling operation, the user is expected to provide both a data type signature $\Omega = (S, F)$ and an Ω-algebra $\mathcal{A}$, with all the data types and functions required by its application domain to define its modeling operations. In the sequel, the considered user types are *point_2D*, *vector_2D* and *color*, that respectively model 2D point positions, 2D vectors and face colors. They are provided with all classical functions such as $+ : \mathit{point\_2D} \times \mathit{vector\_2D} \to \mathit{point\_2D}$ that represents the translation of a point by a vector, $\mathit{center} : \mathit{point\_2D} \times \mathit{point\_2D} \to \mathit{point\_2D}$ that computes the center of two points, $\mathit{bary} : \mathit{point\_2D}^\bullet \to \mathit{point\_2D}$ that computes the barycenter of a multiset of points, or $\mathit{mix} : \mathit{color} \times \mathit{color} \to \mathit{color}$ that defines the average color of two given colors. The algebra $\mathcal{A}$ will be left implicit. When needed, the carrier set $\mathcal{A}_\tau$ of a data type τ will be simply denoted $\lfloor \tau \rfloor$.

3. Topological generalized maps as partially labeled graphs

In topology-based modeling, objects are defined according to:

- their topological structure - *i.e.* their cell subdivision (vertices, edges, faces, volumes) and the adjacency relations between these cells; for example, the three objects of Figure 7(a) have the same topological structure: a closed face F that contains four edges and four vertices;
- their embedding, which includes all other types of information attached to their topological cells, including the geometric informations required to capture their shape; for the objects of Figure 7(a), geometric points are attached to topological vertices and colors are attached to faces.



(a) Objects of same topological structure (b) Topological inconsistencies (c) Embedding inconsistencies

Figure 7: Object inconsistencies

There are many topological structures that allow one to represent different classes of objects: tetrahedral [27] or polyhedral [28, 29, 10], fixed dimension (2D [28] or 3D [29]) or dimension-independent [27, 10], and most of them can be seen as a particular class of graphs. Among those, we choose the topological model of generalized maps (or G-maps) [5, 9, 10] because its mathematical definition can be rather easily encoded within a formal framework. In G-maps, the topological structure is handled by both the graph structure and the arc labels, while the embedding is defined by the node labels.

More precisely, the class of G-maps that represents valid objects is defined by labeling constraints. Hence, to define modeling operations with graph transformations, we investigate under which conditions G-map constraints, and so object consistency, are preserved along transformations. Examples of inconsistencies are given in Figure 7: an edge with a single extremity instead of two, or two faces glued along a vertex instead of an edge are topological inconsistencies, while a face embedded with two colors instead of one, or without any defined color are embedding inconsistencies.

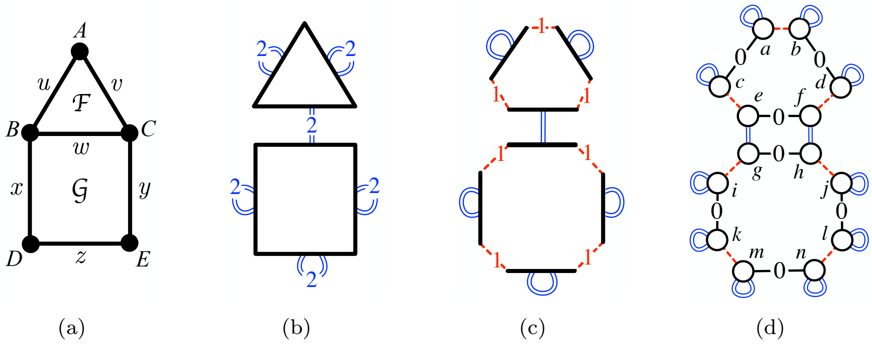


Figure 8: Topological decomposition of a geometric 2D object

3.1. Generalized maps

The representation of an object as a G-map intuitively comes from its decomposition into topological cells (vertices, edges, faces, volumes, etc.). For example, the 2D topological object of Figure 8(a) can be decomposed into a 2-dimensional G-map. The object is first decomposed into faces in Figure 8(b). These faces are *linked* along their common edge with a 2-relation: the index 2 indicates that two cells of dimension 2 (faces) share an edge. In the same way, faces are split into edges connected with the 1-relation in Figure 8(c). At last, edges are split into vertices by the 0-relation to obtain the 2-G-map of Figure 8(d). Nodes obtained at the end of the process are the G-map ones and the different i-relations become labeled arcs: for a 2-dimensional G-map, i belongs to $\{0, 1, 2\}$.

G-maps as therefore particular graphs whose arcs are labeled by integers: for a dimension n, n-G-maps are partially labeled graphs such that arcs are totally labeled in $\mathcal{C}_E = [0, n]$ where $[0, n]$ is the interval of integers between 0 and n. G-maps are non-oriented graphs: thus, for each i-arc of source v, of target v', there is also a corresponding reversed i-arc of source v' and target v. As usual, double reversed arcs are graphically represented by non oriented arcs (see Figure 8(d)). Note that in order to be more readable, in all figures given subsequently, we use the graphical codes introduced in Figure 8 (black line for 0-arcs, red dashed line for 1-arcs and blue double line for 2-arcs) instead of writing a label near the corresponding arc. So, the way non-oriented arcs will be drawn will implicitly indicate their label values.

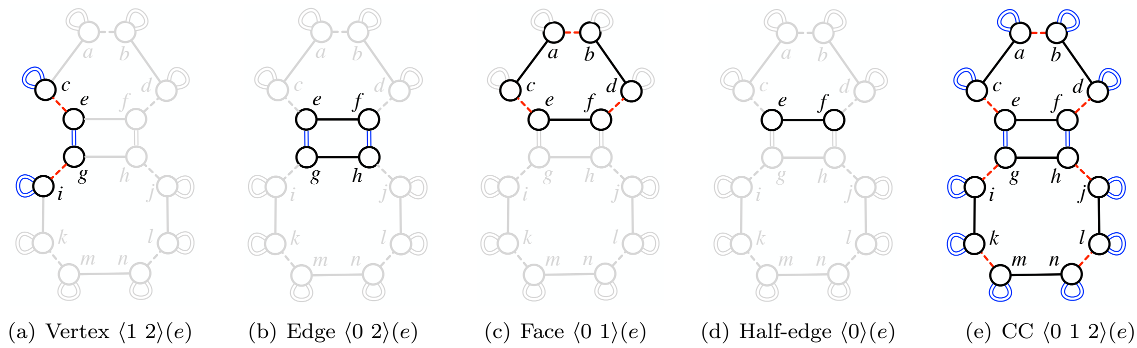


Figure 9: Orbits adjacent to e

Topological cells are not explicitly represented in G-maps but implicitly defined as subgraphs. They can be computed using graph traversals defined by an originating node and a given set of arc labels. For example, in Figure 9(a), the 0-cell (vertex) adjacent to e is the subgraph which contains e, the nodes that can be reached from node e using 1-arcs or 2-arcs (nodes c, e, g and i) and the arcs themselves. This subgraph is denoted by $G\langle 1\ 2\rangle(e)$, or simply $\langle 1\ 2\rangle(e)$ if the context (graph G) is obvious, and models the vertex B of Figure 8(a). In Figure 9(b), the 1-cell adjacent to e (edge w) is the subgraph $G\langle 0\ 2\rangle(e)$ that contains node e and nodes reachable through 0-arcs and 2-arcs (nodes e, f, g and h), and the corresponding arcs. Finally, in Figure 9(c), the 2-cell adjacent to e (face $\mathcal{F}$) is the subgraph denoted by $\langle 0\ 1\rangle(e)$ and built from node e with 1-arcs and 2-arcs.

In fact, topological cells (face, edge or vertex) are particular cases of *orbits* denoting subgraphs built from an originating node and a set of labels. The different *orbit types* of an n-G-map are all possible subsets of $[0, n]$ and are classically denoted by an ordered word o of label placed in brackets $\langle o \rangle$. In addition to the already mentioned orbit types ($\langle 0\ 1 \rangle$ for face, $\langle 0\ 2 \rangle$ for edge, $\langle 1\ 2 \rangle$ for vertex), let us give some other examples of orbit types: the orbit $\langle 0 \rangle(e)$ in Figure 9(d) represents the half-edge adjacent to e, and the orbit $\langle 0\ 1\ 2 \rangle(e)$ in Figure 9(e) represents the whole connected component.

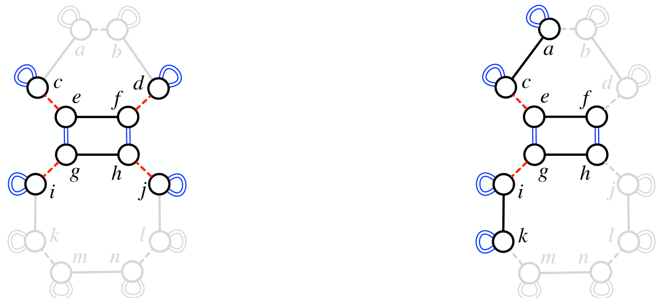
The following definition introduces the notions of topological graphs (which include G-maps but also their transformation patterns), orbits, orbit equivalences, and orbit completions. In particular, this last two notions will be useful later in the article to handle embedding transformations. Note also that according to the notation commonly used in geometric modeling, the arc labeling function of topological graph is denoted by α.

Definition 3 (n-topological graph and orbit). *A partially labeled graph $G = (V, E, s, t, l_V, \alpha)$ is an n-dimensional topological graph if the arc labeling function α is a total function with codomain $\mathcal{C}_E = [0, n]$.*

For $\langle o \rangle$ an orbit type of dimension n, let $\equiv_{G\langle o \rangle}$ be the equivalence orbit relation *defined on $V \times V$ as the reflexive, symmetric and transitive closure built from arcs with labels in o, i.e., ensuring that for each arc e of G labeled by a letter in o, we have $s(e) \equiv_{G\langle o \rangle} t(e)$.*

*For any node v of G, the $\langle o \rangle$-*orbit *(or simply* orbit*) of G adjacent to v is denoted by $G\langle o \rangle(v)$ and is defined as the subgraph of G whose set of nodes is the equivalence class of v using the equivalence relation $\equiv_{G\langle o \rangle}$, whose set of arcs are those labeled on o between nodes of $G\langle o \rangle(v)$, and such that source, target, labeling functions are the restrictions of the corresponding functions of G. If the context is clear, $G\langle o \rangle(v)$ is simply denoted $\langle o \rangle(v)$.*

*More generally, for any subgraph $G' \hookrightarrow G$, the $\langle o \rangle$-*completion *of G' in G, denoted by $G\langle o \rangle(G')$, is defined as the smallest subgraph of G whose set of nodes is the set of nodes of all equivalence classes of G' nodes using $\equiv_{G\langle o \rangle}$, whose arcs are arcs of G' together with all arcs of G that are labeled by a label of o and connect nodes of $G\langle o \rangle(G')$, and such that source, target, labeling functions are the restrictions of the corresponding functions of G.* ◀



(a) Vertex completion of the edge of e (b) Edge completion of the vertex of e

Figure 10: Completions

Let us take two examples to illustrate the notion of completion. The topological graph of Figure 10(a) is the $\langle 1\ 2 \rangle$-completion of $\langle 0\ 2 \rangle(e)$, *i.e* the vertex completion of the edge adjacent to node e. Symmetrically, the Figure 10(b) presents the $\langle 0\ 2 \rangle$-completion of the vertex $\langle 1\ 2 \rangle(e)$.

Let us now give the consistency constraints that objects defined as G-maps must satisfy.

Definition 4 (Generalized map). *An n-dimensional generalized map, or n-G-map, is a partially labeled n-topological graph $G = (V, E, s, t, l_V, \alpha)$ that satisfies the following topological consistency constraints:*

- Non-orientation constraint*: G is non-oriented,*
- Adjacent arc constraint*: each node is the source node of exactly $n+1$ arcs respectively labeled from 0 to n,*
- Cycle constraint*: for every i and j such that $0 \leq i \leq i+2 \leq j \leq n$, there exists a cycle labeled by $ijij$ starting from each node.* ◀

These constraints ensure that objects represented by embedded G-maps are consistent manifolds [9]. In particular, the cycle constraint ensures that in G-maps, two i-cells can only be adjacent along $(i-1)$-cells. For instance, in the 2-G-map of Figure 8(d), the 0202-cycle constraint implies that faces are stuck along topological edges. Let us notice that thanks to loops (see 2-loops in Figure 8(d)), these three constraints also hold at the border of objects.

3.2. Basic topological transformations

Within a geometric modeler, operations defined on objects are called *topological* (respectively *geometric*) if their main purpose is to change the topological structure (respectively the embedding). Obviously, some operations fall under both aspects, as the rounding operation that consists in the replacement of a vertex or a sharp edge by a curved surface [30].

Roughly speaking, topological operations are applications that allow one to build new generalized maps from generalized maps. The definition of topological operations by graph transformation rules advantageously facilitates the study of stating whether or not the resulting graphs are also generalized maps. To achieve this, rules on generalized maps have to preserve by construction the topological constraints of Definition 4.

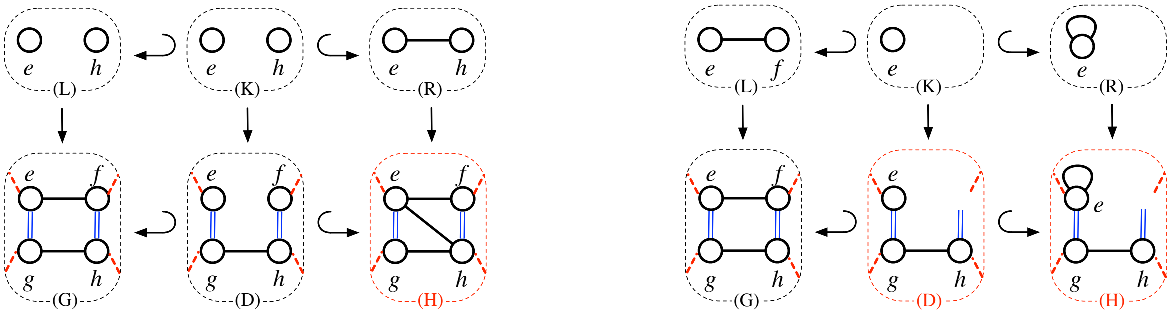
In previous works [11, 6, 12], we elaborated the following syntactic conditions that precisely ensure the preservation of topological consistency:

Theorem 1 (Topological consistency preservation). *Let $r : L \hookleftarrow K \hookrightarrow R$ be a graph transformation rule, G an n-G-map and $m : L \to G$ a match morphism.*

The direct transformation $G \Rightarrow^{r,m} H$ produces an n-G-map H if the following conditions of topological consistency preservation *are satisfied:*

- Non-orientation condition*: the three graphs L, K and R are non-oriented n-topological graphs.*
- Adjacent arcs condition*:*
  - *preserved nodes of K are sources of arcs having the same labels in both the left-hand side L and the right-hand side R;*
  - *removed nodes of $L \backslash K$ and added nodes of $R \backslash K$ must be source of exactly $n+1$ arcs respectively labeled from 0 to n.*
- Cycle condition*: for all couple (i, j) such $0 \leq i \leq i+2 \leq j \leq n$,*
  - *any added node of $R \backslash K$ is the source of an $ijij$-cycle;*
  - *any preserved node of K which is the source of an $ijij$-cycle in L, is also the source of an $ijij$-cycle in R;*
  - *any preserved node of K which is not the source of an $ijij$-cycle in L is source of the same i-arcs and j-arcs in L and R.* ◀

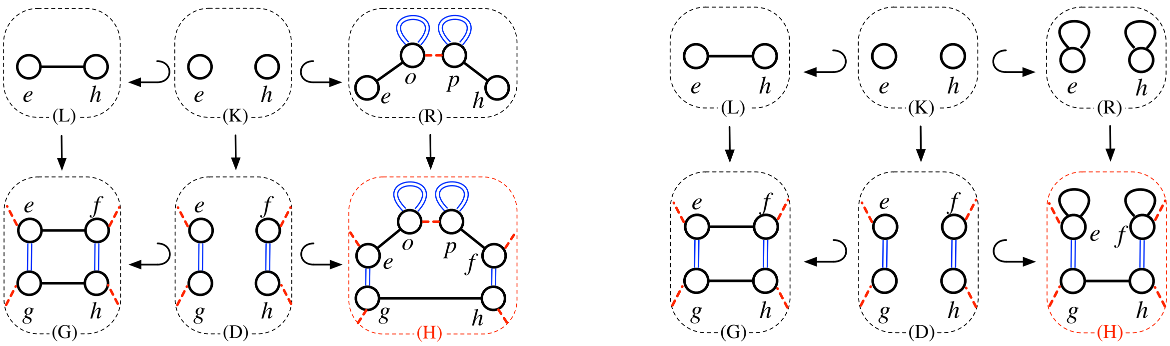
The interested reader can find the proof of this theorem in [6]. In particular, we demonstrated that the dangling condition (see Subsection 2.2) is always ensured when a rule that satisfies those conditions is applied to a G-map.



(a) (b)

Figure 11: Non-respect of adjacent arcs condition

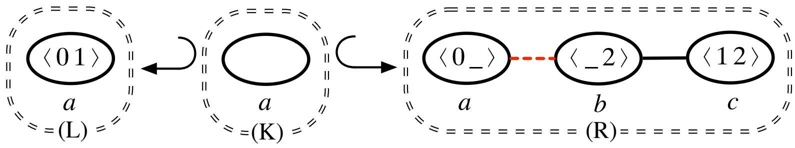
Intuitively, the adjacent arcs condition (combined with the non-orientation condition) ensures that every node concerned by the rule application ends up being the source and the target of exactly one i-arc for each $i \in [0, n]$ in the transformed object. The first point requires that preserved nodes have their adjacent i-arcs in both sides of the rule. Indeed, by construction, i-arcs that are not matched by the rule are preserved in the transformed graph. For example, the rule of Figure 11(a) adds a 0-arc between nodes e and h in R without matching any 0-arc in L. Consequently, in the resulting graph H, nodes e and h have both their original 0-arc issued from G and the 0-arc added in R, and therefore H is not a G-map. The second point requires that added or removed nodes have exactly one i-arc for each $i \in [0, n]$. Indeed, nodes can only be consistently added with all their adjacent arcs. Similarly, removing a node without matching a given i-arc would imply that an i-arc remains in the transformed G-map without one extremity. For example, the rule of Figure 11(b) removes node f without matching its adjacent 1-arc and 2-arc. Consequently, the resulting graph H contains two dangling arcs.



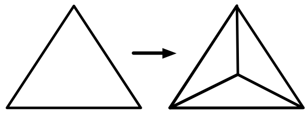
(a) (b)

Figure 12: Non-respect of cycle condition

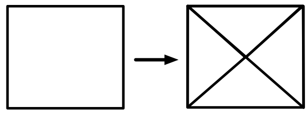
The cycle condition ensures that every node transformed by the rule ends up belonging to an $ijij$-cycle for all $0 \leq i \leq i+2 \leq j \leq n$. The first point requires that added nodes belong to an $ijij$-cycle in R, as all their adjacent arcs belong to R. For example, the rule of Figure 12(a) add two nodes o and p without their 0202-cycle in R. Consequently, they do not belong to such a cycle in the resulting graph H. The second point requires that preserved nodes which belong to an $ijij$-cycle in L also belong to such a cycle in R. Similarly to the previous point, their adjacent i-arcs and j-arcs belong to R, therefore the $ijij$-cycle has to belong to R. The last point requires that preserved nodes which do not belong to an $ijij$-cycle in L have their adjacent i-arcs and j-arcs also preserved. As a matter of fact, any modification of those arcs might break the $ijij$-cycle as it is only partially matched by the rule. For example, in Figure 12(b), by removing the 0-arc between nodes e and h in the rule, we break the 0202-cycle in the resulting graph H.



(a) Rule scheme of the topological triangulation



(b) Triangulation of a triangle



(c) Triangulation of a square

Figure 13: Topological triangulation

3.3. Topological rule schemes

In previous works [11, 6, 12, 26], we introduced special variables, called topological variables, provided with an orbit type in order to abstract any orbit of the given type. Using such a variable, the rule scheme[5] given in Figure 13(a) models the topological triangulation of any face (that is any cell of type $\langle 0\ 1 \rangle$). When applied to a triangle face in Figure 13(b), it subdivides the triangle face in three new triangle faces. In the same way, it subdivides the square face of Figure 13(c) in four triangle faces. In this scheme, the topological variable is $\langle 0\ 1 \rangle$ in the left-hand side L. It indicates which orbit is matched to be transformed. In K and R, $\langle 0\ \_ \rangle$, $\langle \_\ 2 \rangle$ and $\langle 1\ 2 \rangle$ indicate how the arcs of the considered orbit should be modified.

More precisely, the rule scheme can be read as follows:

1. a face is matched in the object under transformation by the node a labeled in L with the orbit type $\langle 0\ 1 \rangle$;
2. in R, each node of the matched face is preserved because of node a, and duplicated twice, one copy for node b and one copy for node c in R;
3. for the matched face, 0-arcs are conserved while 1-arcs are removed as a is relabeled in R by $\langle 0\ \_ \rangle$, in which the label 0 is preserved while the label 1 is replaced by an "empty" label $\_$; similarly, for node b, $\langle \_\ 2 \rangle$ denotes both removal of 0-arcs and 2-relabeling of 1-arcs while for the node c, $\langle 1\ 2 \rangle$ denotes 1-relabeling of 0-arcs and 2-relabeling of 1-arcs,
4. at last, any node a of the matched face is linked to its image in copy b with a 1-arc, and all b and c images of a given node are linked together with a 0-arc.

In [6], we extended the conditions of Theorem 1 to rule schemes. Hence, those schemes provide an efficient implementation of topological modeling operations, as both the topological consistency of transformations and the dangling condition (that usually has a high detection cost) can be ensured by statically analyzing schemes.

Finally, note that the triangulation operation of Figure 13 is purely topological since we did not define a precise position for the created vertex. If one wants to specify that the new vertex should be located at the barycenter of the vertex positions associated to the triangulated face, the operation will become both geometric and topological. In the sequel, each time the topology will be modified, it will be done using rules satisfying conditions of Theorem 1 and thus without topological variables.

[5]Let us recall that rule schemes can be identified by the dotted double circle around L, K, and R.

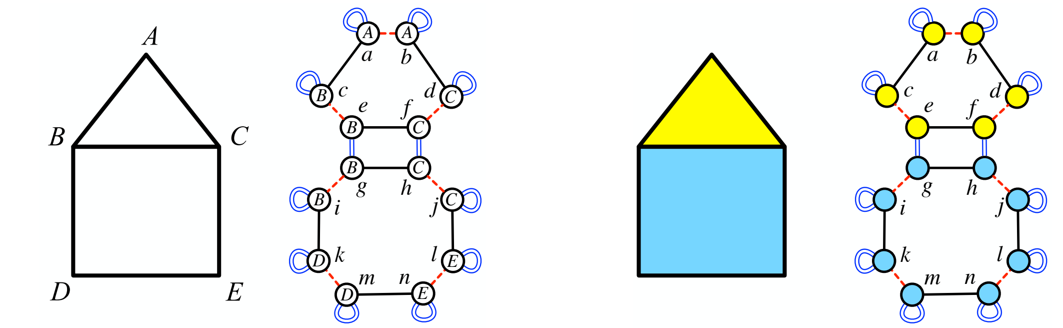
4. Embedded generalized maps and their basic transformations

In the following, we will consider the two embedding data types on 2D objects illustrated in Figure 7(a) in Section 3 (2D points and colors). Most importantly, for pedagogical issue, we will consider them individually: we will either consider 2D geometric points attached to topological vertices or colors attached to faces. However, realistic objects handle simultaneously several data types holding on different topological cells.

4.1. Embedding representation

The topological structures of n-G-maps have ben defined as labeled graphs where the arc label set is $\mathcal{C}_E = [0, n]$. We complete here this definition with node labels to represent the embedding. We already sketched that every dedicated embedding has its own data type and is defined on a particular kind of topological cell: in our example, point are attached to vertices and colors to faces. More accurately, an embedding can be attached to any arbitrary orbit type (*e.g.* a speed to the connected compound orbit).

Consequently, the node labeling function that defines an embedding has to be typed in two ways: by the concerned topological orbit type and the used data type. We characterize such a node labeling function as an *embedding operation* $\pi : \langle o \rangle \to \tau$ where π is the operation name, τ is its data type and $\langle o \rangle$ is its domain given as an n-dimensional orbit type. Hence, for a G-map embedded on an embedding $\pi : \langle o \rangle \to \tau$, the node label set $\mathcal{C}_V$ is the set of values $\lfloor \tau \rfloor$ of type τ and when the profile of π is obvious, the node labeling function is simply noted π. Thus, a G-map provided with a single embedding will be a particular case of partially labeled graph, generically denoted as a tuple $(V, E, s, t, \pi, \alpha)$.



(a) 2-G-map with vertices embedded by 2D points (b) 2-G-map with faces embedded by colors

Figure 14: Two embedding operations

In this article, we consider the following embedding operations:

- $pt : \langle 1\ 2 \rangle \to point\_2D$ the embedding that associates 2-dimensional coordinates (values of type $point\_2D$) with vertices ($\langle 1\ 2 \rangle$-orbits) of 2-G-maps (see Figure 14(a));
- $col : \langle 0\ 1 \rangle \to color$ the embedding that associates colors (values of type $color$) with faces ($\langle 0\ 1 \rangle$-orbits) of 2-G-maps (see Figure 14(b)).

Moreover, as an embedding operation $\pi : \langle o \rangle \to \tau$ is characterized by its domain orbit, it is expected that in an embedded G-map $G = (V, E, s, t, \pi, \alpha)$, all nodes of a common $\langle o \rangle$-orbit share the same label by π, also called π-label. For example, in Figure 14(a), nodes c, e, g and i that belong to the same vertex orbit $\langle 1\ 2 \rangle$ are labeled by the same value B. Similarly, in Figure 14(b), nodes a, b, c, d, e and f that belong to the same face are labeled with the same yellow color. This property is captured by an embedding constraint that embedded G-maps have to satisfy [31].

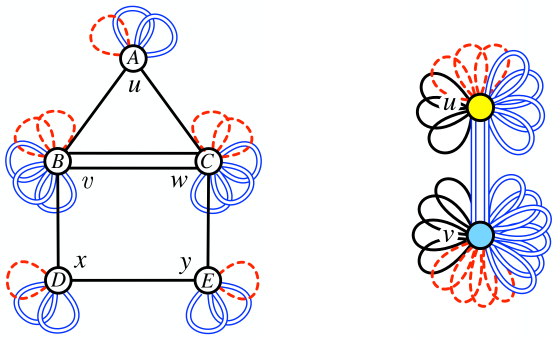
Definition 5 (Embedded graph and embedded generalized map). *Let $\pi : \langle o \rangle \to \tau$ be an embedding operation of dimension $n \geq 0$ with $\langle o \rangle$ an orbit type of dimension n and τ a data type.*

- Embedded graph*: A* graph *embedded on π, or π-embedded* graph*, is an n-topological graph $G = (V, E, s, t, \pi, \alpha)$ where π labels nodes on $\mathcal{C}_V = \lfloor \tau \rfloor$.*
- Embedding consistency constraint*: A π-embedded graph satisfies the* embedding consistency constraint *if for all nodes v and w such that $v \equiv_{\langle o \rangle} w$, if $\pi(v) \neq \perp$ and $\pi(w) \neq \perp$, then $\pi(v) = \pi(w)$.*
- Embedded G-map*: A π-*embedded G-map *is an n-G-map embedded on π satisfying the embedding consistency constraint and such that π is a total function (i.e. $Dom(\pi) = V$).* ◀

Note that the embedding consistency constraint allows an orbit to be partially π-labeled as long as the defined π-labels are equal. This partial orbit labeling will be useful in the sequel to write rules. Conversely, as embedded G-maps are totally labeled, the embedding consistency constraint entails that all nodes of any $\langle o \rangle$-orbit share the embedding same value, *i.e.* for all nodes v and w such that $v \equiv_{\langle o \rangle} w$, $\pi(v) = \pi(w)$ with $\pi(v) \neq \perp$.

Intuitively, the topological structure of any π-embedded graph G can then be highlighted by forgetting node labels. For $G = (V, E, s, t, \pi, \alpha)$, $G_\alpha = (V, E, s, t, \perp, \alpha)$ denotes the underlying topological structure where the everywhere undefined function $\perp$ replaces the node labeling function π.

For π-embedded graph G that satisfies the embedding consistency constraint, the embedding structure can also be highlighted by applying a quotient application along $\langle o \rangle$-orbits. As $\equiv_{\langle o \rangle}$ defines a partition of the set V of nodes, it allows to build a quotient graph of G. For the quotient set of nodes, we consider the set of $\langle o \rangle$-orbits of G. As all nodes of an $\langle o \rangle$-orbit of G share the same π-label, the resulting quotient node directly inherits this shared π-label. For the quotient set of arcs, we consider the set of arcs inherited from G by redefining source and target nodes with their corresponding quotient nodes and by preserving their labels.



(a) Vertices/2D points (b) Faces/colors

Figure 15: Quotients of the embedded 2-G-maps Figure 14

For example, the quotient along $\langle 1\ 2 \rangle$-orbits of the embedded G-map of Figure 14(a) is the graph of Figure 15(a). As nodes c, g, e and i belong to the same vertex orbit, they share the same embedding B and give rise to the B-labeled quotient node v in Figure 15(a). The resulting quotient graph contains 5 nodes, one per $\langle 1\ 2 \rangle$-orbit, with a well-defined π-label. Let us note that by construction, arcs with labels belonging to the orbit type $\langle 1\ 2 \rangle$ become loops. As a consequence, all 1-arcs and 2-arcs adjacent to c, e, g or i are transformed into loops on node v in the quotient graph.

Similarly, Figure 15(b) presents the quotient along $\langle 0\ 1 \rangle$-orbits of the 2-G-map of Figure 14(b). Nodes a, b, c, d, e and f of the triangle face give rise to the yellow quotient node u while the nodes of the square face give rise the blue quotient node v.

Definition 6 (Embedding quotient). *Let $G = (V, E, s, t, \pi, \alpha)$ be a graph embedded on $\pi : \langle o \rangle \to \tau$ that satisfies the embedding consistency constraint.*

*The π-*quotient[6] graph *of G is the graph $G_{/\pi} = (V_{/\pi}, E_{/\pi}, s_{/\pi}, t_{/\pi}, \pi_{/\pi}, \alpha_{/\pi})$ defined by:*

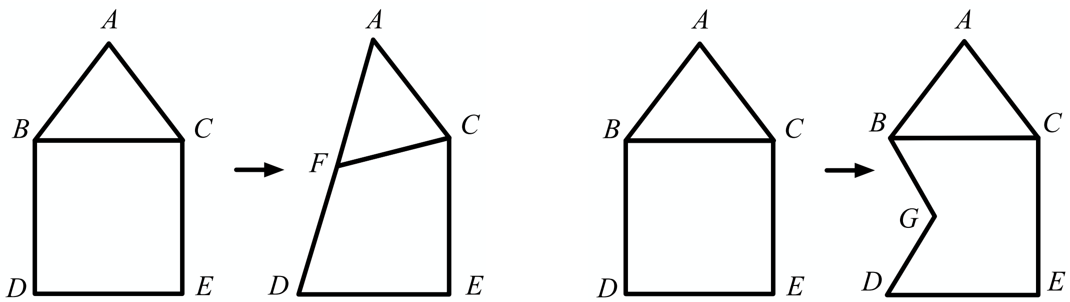
- *$V_{/\pi} = V_{/\equiv_{\langle o \rangle}}$;*
- *$E_{/\pi} = E$ with $\forall e \in E_{/\pi}, \alpha_{/\pi}(e) = \alpha(e), s_{/\pi}(e) = [s(e)]$ and $t_{/\pi}(e) = [t(e)]$;*
- *for $[v]$ in $V_{/\equiv_{\langle o \rangle}}$, $\pi_{/\pi}([v]) = \pi(w)$ if there exists w in $G\langle o \rangle(v)$ such that $\pi(w) \neq \perp$, otherwise $\pi_{/\pi}([v]) = \perp$.*

*The π-*quotient morphism *$q : G \to G_{/\pi}$ is defined by: $\forall v \in V$, $q_V(v) = [v]$ and $q_E = id$.* ◀

Note that as embedded G-maps both satisfy the embedding consistency constraint and are totally labeled, their quotient graphs are also totally labeled.

4.2. Basic embedding transformations

As π-embedded G-maps constitute a particular class of partially labeled graphs, we now investigate how modeling operations on embedded G-maps that modify their geometry using graph transformation rules (see Definition 1). To illustrate this section, we consider the two operations given in Figure 16 that apply on objects with the point embedding $pt : \langle 1\ 2 \rangle \to point\_2D$. The vertex translation of Figure 16(a) in a purely geometric operation as it does not affect the topological structure. Point B is translated to F. Conversely, the edge folding of Figure 16(b) affects both the topological structure and the embedding. An edge of the square face is split into two edges by introducing a new vertex which is embedded by the $point\_2D$ value G.



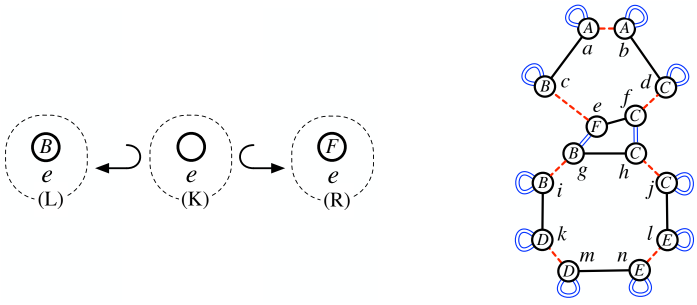
(a) Vertex translation (b) Edge folding

Figure 16: Two operations on the point embedding

In this section, we explore to what extent basic geometric modeling operations designed for a particular embedded G-map can be defined as basic graph transformations as introduced in Definition 1. The key point is to ensure that consistency constraints of embedded G-maps are preserved along rules applications, provided that rules satisfy some syntactic conditions.

In the same way that we gave in Theorem 1 syntactic conditions for the preservation of topological consistency constraints, we here investigate syntactic conditions to ensure that embedded G-maps are transformed into embedded G-maps. Let us take the example of the vertex translation of Figure 16(a). Intuitively, we could consider at first sight the rule of Figure 17(a). Unfortunately, such a rule is not appropriate for our needs. Indeed, by matching node e of the rule of Figure 17(a) with node e of the G-map of Figure 14(a), its application results in the graph given in Figure 17(b). Clearly, this graph does not satisfy the embedding consistency constraint as node e does not have the same label than the other nodes of its vertex orbit (c, g and i).

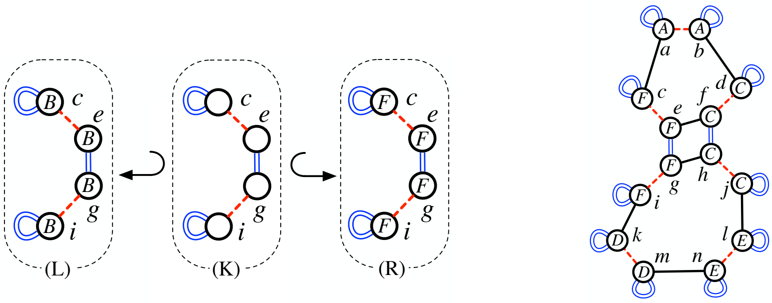
[6]Let X be a set and $\equiv$ an equivalent relation on X, the equivalence class $[x]$ of any element $x \in X$ is defined as $[x] = \{y \in X \mid x \equiv y\}$, and the quotient set $X_{/\equiv}$ is the set $\{[x] \mid x \in X\}$



(a) Translation rule of one node (b) Incoherent application

Figure 17: Incoherent translation

To avoid this, all node labels of an embedding orbit should be modified simultaneously and in the same manner. For example, the rule of Figure 18(a) matches (respectively rewrites) a full vertex orbit in L (respectively in R): indeed, all nodes are connected with both 1-arcs and 2-arcs. In fact, both L and R are full $\langle 1\ 2 \rangle$-orbit. Thus, the application of this rule to the *pt*-embedded 2-G-map of Figure 14(a) along the identity match morphism gives the *pt*-embedded 2-G-map of Figure 18(b). The following theorem introduces syntactic conditions on rules that ensure embedding consistency preservation.



(a) Translation rule of one vertex (b) Coherent application

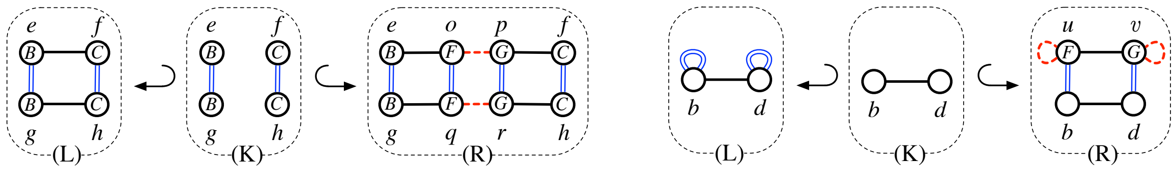
Figure 18: Coherent translation

Theorem 2 (Preservation of the embedding consistency). *Let $\pi : \langle o \rangle \to \tau$ be an embedding operation, $r : L \hookleftarrow K \hookrightarrow R$ be a π-embedded graph transformation rule that satisfies conditions of topological consistency preservation, G a π-embedded G-map and $m : L \to G$ a match morphism.*

The direct transformation $G \Rightarrow^{r,m} H$ produces a π-embedded G-map if the following conditions of embedding consistency preservation *are satisfied:*

- Embedding consistency*: L, K and R satisfy the embedding consistency constraint of Definition 5.*
- Full match of transformed embeddings*: if v is a node of K such that $\pi_L(v) \neq \pi_R(v)$, then every node of $R\langle o \rangle(v)$ is labeled and is the source of exactly one i-arc for each i of $\langle o \rangle$.*
- Labeling of extended embedding orbits*: if v is a node of K and there exits a node w in $R\langle o \rangle(v)$ such that w is not in $L\langle o \rangle(v)$, then there exist v' in K with $v' \equiv_{L\langle o \rangle} v$ and $v' \equiv_{R\langle o \rangle} v$ such that $\pi_L(v') \neq \perp$.* ◀

The first of these conditions is straightforward: it requires that all parts of the rule satisfy the embedding consistency constraint. For example, the rule of Figure 19(a) breaks this condition as it adds to the G-map a new vertex (nodes o, p, q and r) embedded with two different points F and G.



(a) Non-consistent added vertex

(b) Unsafe extension of two vertices

Figure 19: Two non-consistent rules that break conditions of Theorem 2.

The second condition forbids the partial redefinition of the embedding shared by an $\langle o \rangle$-orbit as it would break the embedding consistency. If a preserved node has a transformed embedding, then its $\langle o \rangle$-orbit in R is a totally labeled full orbit. The rule of Figure 17(a) falls in this case as node e has its label changed from B to F without fully matching the topological vertex (1-arc and 2-arc are missing). Hence, an embedding value can only be modified if it is modified for the whole support orbit.

The last condition forbids the extension of an $\langle o \rangle$-orbit (by adding new nodes or merging with another $\langle o \rangle$-orbit) without matching the existing embedding value of the orbit. For exemple, the rule of Figure 19(b) breaks this condition as an half-edge whose embedding is unmatched is added to another half-edge whose vertices are embedded by the two points F and G. Therefore, the application of this rule to the object of Figure 14(a) along the identity morphism would break the embedding consistency: *e.g.* node b would be labeled by A while its added 2-neighbor u would be labeled by F. As this third condition entails that nodes b and d are labeled in L (and thus in R), the rule labeling should be completed. In R, nodes b and d should be respectively labeled by F and G due to the embedding consistency condition. In L, they should also be labeled by F and G as the condition of full match of transformed embeddings prevents to change their labels while the two vertex orbits are not fully matched by the rule (1-neighbors of b and d are not matched).

▶ ***Proof.*** Let $\pi : \langle o \rangle \to \tau$ be an embedding operation, $r : L \hookleftarrow K \hookrightarrow R$ be a graph transformation rule that satisfies the conditions of topological consistency preservation, G a π-embedded G-map and $m : L \to G$ a match morphism. We consider the following direct transformation :

$$\begin{tikzcd} L \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & K \arrow[l, hook'] \arrow[r, hook] \arrow[d, "b"] \arrow[dr, phantom, "(2)"] & R \arrow[d, "c"] \\ G & D \arrow[l, hook'] \arrow[r, hook] & H \end{tikzcd}$$

In double-pushout transformation, each element of H (node, arc, or label) comes either from the right-hand side of the rule R, or from the graph G (through D), or from both. Therefore, to check whether two nodes of H linked with an arc labeled in $\langle o \rangle$ are labeled with the same embedding value, the proof considers all cases for the arc source, target and labelling.

Since the rule r satisfies the conditions of topological consistency preservation of Theorem 1, the direct transformation $G \Rightarrow^{r,m} H$ is well-defined (*i.e.* the dangling condition is satisfied) and the resulting graph H is an n-G-map. Moreover, with the result of [14] stating that total labeling is preserved by rule application, we know that H is totally labeled, in particular that π is defined on every node of H.

It remains to prove that thanks to the conditions listed in Theorem 2, H is a π-embedded G-map.

We then show by exhaustion[7] that for any label i of the orbit type $\langle o \rangle$, and for any i-arc e of H, the source node v of e and the target node w of e have the same defined π-label, *i.e.* $\pi_H(v) = \pi_H(w)$. As this will ease some symmetrical cases, let us note that thanks to the non-orientation preservation (see Theorem 1), e has always a symmetric i-arc with source w and target v in H and also in D (resp. G) if e is also an arc of D (resp. G).

[7]Cases are hierarchically numbered to ease proof commentary.

1 • If e has no antecedent in R, e only comes from G. More precisely e, v and w are respectively arc and nodes of D and G.

1.1 • If both v and w have no antecedent in R. Then v (resp. w) has the same π-label in G ($\pi_G(v) = \pi_H(v)$ and $\pi_G(w) = \pi_H(w)$), D and H. As G is a π-embedded G-map, $\pi_G(v) = \pi_G(w)$. Therefore $\pi_H(v) = \pi_H(w)$.

1.2 • If both v and w have two antecedents v' and w' in R. Because of the adjacent arcs condition of Theorem 1, R has no i-arc neither with source v' nor w'. And thus $R\langle o\rangle(v')$ and $R\langle o\rangle(w')$ are not full orbits. Because of the condition of full match of transformed embeddings, v' and w' are preserved nodes with preserved π-labels, *i.e.* $\pi_L(v') = \pi_R(v')$ and $\pi_L(w') = \pi_R(w')$. Thus v and w have the same defined π-label in G and H, *i.e.* $\pi_G(v) = \pi_H(v)$ and $\pi_G(w) = \pi_H(w)$. As G is a π-embedded G-map, $\pi_G(v) = \pi_G(w)$ and therefore $\pi_H(v) = \pi_H(w)$.

1.3 • If v has an antecedent v' in R, and w has no antecedent in R. According to ***1.1***, w has the same defined π-labels in G and H ($\pi_G(w) = \pi_H(w)$). According to ***1.2***, v has the same defined π-labels in G and H ($\pi_G(v) = \pi_H(v)$). Finally, as G is a π-embedded G-map, $\pi_G(v) = \pi_G(w)$ and therefore $\pi_H(v) = \pi_H(w)$.

1.4 • If v has no antecedent in R and w has an antecedent in R, this case is similar to case ***1.3***.

2 • If e has an antecedent e' in R. Let v' and w' be respectively the source and target nodes of e' in R.

2.1 • If both v' and w' have defined π-labels in R, *i.e.* $\pi_R(v') \neq \perp$ and $\pi_R(w') \neq \perp$. Thanks to the embedding consistency condition, v' and w' have the same defined π-label, $\pi_R(v') = \pi_R(w')$ and therefore $\pi_H(v) = \pi_H(w)$.

2.2 • If both v' and w' have undefined π-labels in R, *i.e.* $\pi_R(v') = \perp$ and $\pi_R(w') = \perp$. Thanks to the rule definition (see Definition 1), v' and w' are nodes of K and thus of L, with $\pi_L(v') = \perp$ and $\pi_L(w') = \perp$. Then v and w also come from G, but not necessarily from the same $\langle o\rangle$-orbit.

2.2.1 • If v' and w' do not come from the same $\langle o\rangle$-orbit, *i.e.* $v' \not\equiv_{L\langle o\rangle} w'$. Thanks to the condition of labeling of extended embedding orbits, there exists $x' \in K$ with $x' \equiv_{L\langle o\rangle} v'$ and $x' \equiv_{R\langle o\rangle} v'$ such that $\pi_L(x') \neq \perp$ and therefore $\pi_R(x') \neq \perp$. Symmetrically, there exists $y' \in K$ with $y' \equiv_{L\langle o\rangle} v'$ and $y' \equiv_{R\langle o\rangle} v'$ such that $\pi_L(y') \neq \perp$ and therefore $\pi_R(y') \neq \perp$. Moreover, as $x' \equiv_{R\langle o\rangle} y'$, the embedding consistency condition ensures that $\pi_R(x') = \pi_R(y')$. Thanks to the condition of full match of transformed embeddings, x' and y' have their π-label preserved by the rule as their $\langle o\rangle$-orbit in R is not totally labeled, and therefore $\pi_L(x') = \pi_L(y')$ (both defined). As $v' \equiv_{L\langle o\rangle} x'$ and $w' \equiv_{L\langle o\rangle} y'$, we have $v \equiv_{G\langle o\rangle} x$ and $v \equiv_{G\langle o\rangle} y$ with x and y the respective images of x' and y' in G. Then because G is a π-embedded G-map, $\pi_G(v) = \pi_G(x)$ and $\pi_G(w) = \pi_G(y)$ and therefore $\pi_H(v) = \pi_H(w)$.

2.2.2 • If v' and w' come from the same $\langle o\rangle$-orbits, *i.e.* $v' \equiv_{L\langle o\rangle} w'$. Then $v \equiv_{G\langle o\rangle} w$ and because G is a π-embedded G-map, $\pi_G(v) = \pi_G(w)$ and then $\pi_H(v) = \pi_H(w)$.

2.3 • If v' has a defined π-label in R but not w', *i.e.* $\pi_R(v') \neq \perp$ and $\pi_R(w') = \perp$. Thanks to the rule definition, w' is a node of K and thus of L with $\pi_L(v') = \perp$. Because of the condition of full match of transformed embeddings, v' can not be an added node and is therefore also a node of K and thus of L with $\pi_L(v') \neq \perp$. The demonstration is then similar to ***2.2*** •, with two cases depending on wether v' and w' come from the same $\langle o\rangle$-orbit, but using directly v' instead of x' as it is labeled.

2.4 • If w' has a defined π-label in R but not v', *i.e.* $\pi_R(w') \neq \perp$ and $\pi_R(v') = \perp$, this case is similar to case ***2.3***.

By transitivity of arcs labeled in $\langle o\rangle$, all nodes v and w of H in the same $\langle o\rangle$-orbit ($v \equiv_{H\langle o\rangle} w$) have the same defined π-label, *i.e.* $\pi_H(v) = \pi_H(w)$. Thus H is a π-embedded G-map. ◀

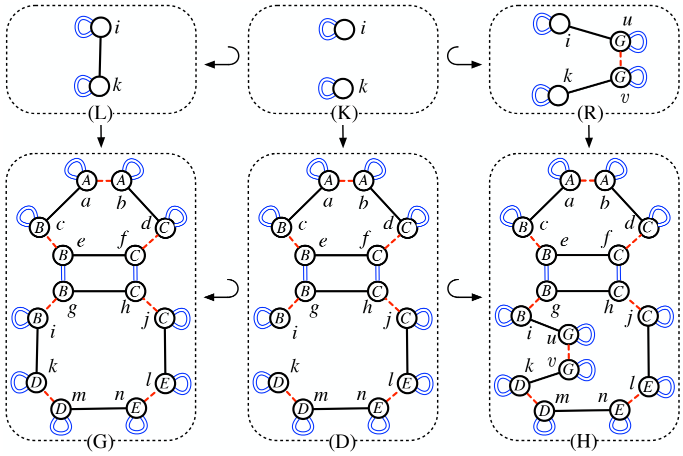


Figure 20: Application of the edge folding

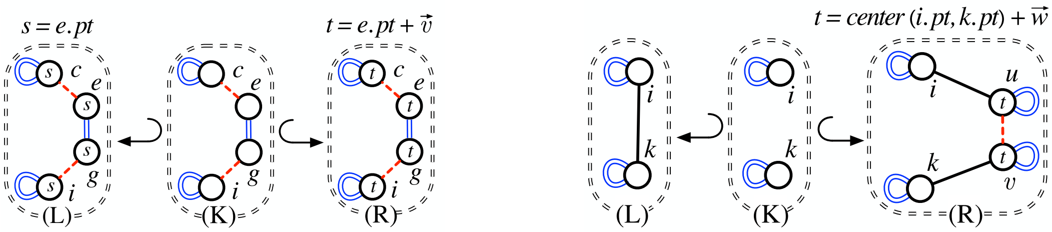
Let us illustrate some of the previous cases with the example of Figure 20 that details the application of the edge folding operation presented in Figure 16(b) on the embedded G-map of Figure 14(a). The 1-arc that links nodes a and b in H falls into the trivial case ***1.1***. The arc and the two nodes belong to object G and are not matched by the rule. As G satisfies the embedding consistency constraint of embedded G-maps, nodes a and b have the same *pt*-label in G and therefore in H. The 1-arc that links nodes i and g in H falls into the case ***1.3***, as the arc and node g are not matched by the rule, conversely to node i. As the vertex orbit that contains both nodes i and g in G is not fully matched, the condition of the full match of transformed embeddings prevents the rule to modify the *pt*-labels of node i. This ensures that nodes i and g have the same *pt*-label in H as they have the same *pt*-label in G. Finally, the 1-arc that links nodes u and v with define labels in H falls into the case ***2.1*** and the embedding consistency condition ensures that their π-labels are equal in R.

5. Rule schemes

This section introduces the rule scheme syntax that allows us to define modeling operations independently from the object embedding values. Following the approach of [15], this syntax includes the use of dedicated variables.

5.1. Node variables

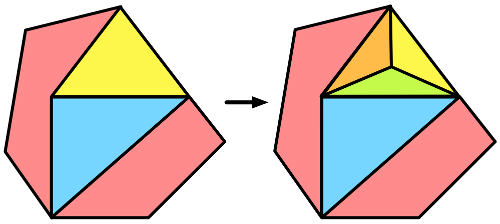
As mentioned in Section 2, attribute variables of [15] do not exactly fit our usage. Computing the new embedding requires both to access the existing embedding (node labels in the transformed object) and to traverse the topological structure (neighboring nodes in the transformed object). Therefore, taking benefit from G-maps regular structures, this article introduces new variables called *node variables* and provides dedicated operators. Instead of defining a new set of variable names, this approach consists in directly using the identifiers of the nodes of L as variables, and therefore variable names freely exist for all transformation.



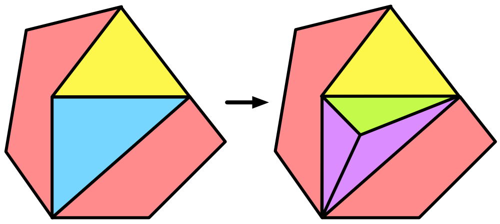
(a) Vertex translation (b) Edge folding

Figure 21: Rule schemes of the operations of Figure 16

The rule schemes of Figure 21 respectively define the translation of a vertex and the folding of an edge, both previously illustrated in Figure 16. Scheme nodes are labelled with terms[8] upon node variables of L, allowing both to match the existing embedding and to express the new embedding computation. In Figure 21(a), the term $e.pt$ in L gives access to the 2D position associated to the matched vertex, while the term $e.pt + \vec{v}$ defines in R the new position fo the translated point[9]. At scheme application, the node variable e of L will be substituted by the node matched by e in the transformed object. The operator $.pt$ (resp. $.\pi$) will then simply grant access to its pt-label (resp. π-label). Similarly in Figure 21(b), $i.pt$ and $k.pt$ are the two positions associated to the matched edge and $center(i.pt, k.pt)$ defines the corresponding center.



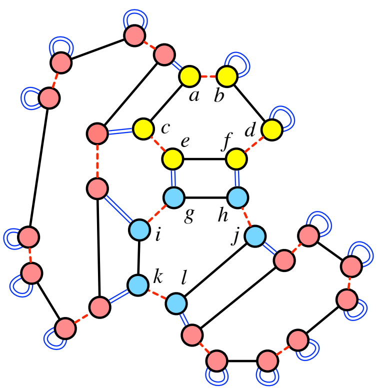
(a)



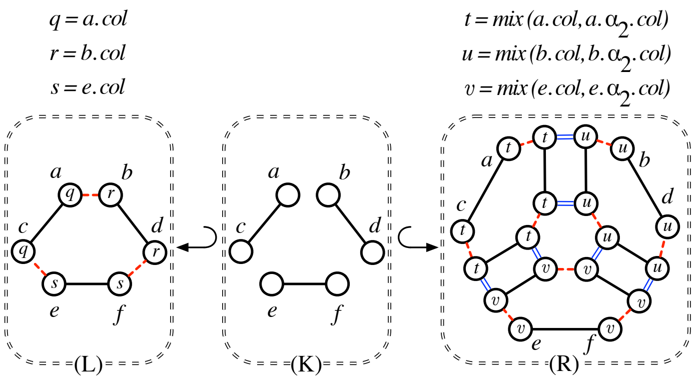
(b)

Figure 22: Face triangulations on the color embedding

As described in Section 3, n-G-maps are highly regular graphs. Every node has $n+1$ neighbors respectively connected by 0, 1, ..., n. Therefore, for all i in $[0, n]$, we can define an $.\alpha_i$ operator on node variables that gives access to their unique i-neighbor. Let us consider the face triangulation of Figure 22 in the case of the color embedding $col : \langle 0\ 1 \rangle \to color$. To smooth face colors, each created triangle is colored by the mix between the original color of the triangulated face and the color of its adjacent face. Using the $.\alpha_2$ operator to access adjacent faces, this operation is defined by the rule scheme of Figure 23(b). In the term $v = mix(e.col, e.\alpha_2.col)$ that defines the color of the bottom face, $e.\alpha_2$ allows the access to the 2-neighbor of e in the transformed object. At application to the object of 23(a) along the identity morphism (as in Figure 22(a)), this neighbor is g and the face color is therefore defined as $mix(e.col, g.col) = mix(●, ●) = ●$. Similarly, if the scheme is applied as in Figure 22(b) with a match morphism that associates node e in the rule with node l in the transformed object, the face color is defined as $mix(l.col, l.\alpha_2.col) = mix(●, ●) = ●$. Finally, note that the case of nodes without adjacent face is covered thanks to the 2-loops - $e.g$ in the first case of Figure 22(a), the term $u = mix(b.col, b.\alpha_2.col)$ is evaluated as $mix(b.col, b.col) = mix(●, ●) = ●$.



(a) Transformed object



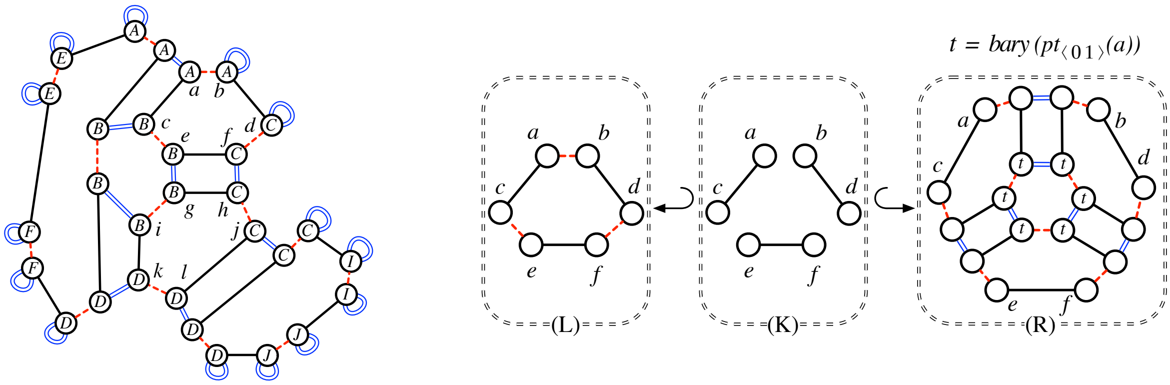
(b) Rule scheme

Figure 23: Face triangulations of Figure 22 on a col-embedded G-map

[8] Note that terms are detailed on top of the rules for readability purposes.
[9] In accordance with Definition 1, rule nodes must be labeled in L in order to relabel them.

5.2. Collect operators

In addition to basic operators ($.\pi$ and $.\alpha_i$), we introduce operators that collect all the embedding values carried by a given orbit. Let us illustrate these operators with the face triangulation, but in the case of the point embedding $pt : \langle 1\ 2 \rangle \to point\_2D$. It is usually expected that the created vertex is positioned at the center of the triangulated face. For example, to triangulate the top triangle of the object of Figure 24(a), the added vertex should be positioned at the barycenter of A, B and C.



(a) Transformed object

(b) Rule scheme

Figure 24: Face triangulation on a *pt*-embedded G-map

To compute this barycenter, the rule scheme of Figure 24 uses the operator $pt_{\langle 0\ 1 \rangle}$ to collect the point values carried by the adjacent face (adjacent $\langle 0\ 1 \rangle$-orbit). At scheme application to the object of Figure 24(a) along the identity match morphism, $pt_{\langle 0\ 1 \rangle}(a)$ will collect the multiset $[\![A, B, C]\!]$. Similarly, its application to the second triangle will result in $[\![B, C, D]\!]$. Intuitively, this operator is based on the quotient representation introduced in Definition 6 that associate each embedding orbit to a single node, and therefore to a single label.

Consequently, each point value appears only once in the resulting multisets regardless of how many times they appears in the object (*e.g* A appears 4 times while B appears 6 times in Figure 24(a)). If a same point was collected multiple times, this would entail that multiple vertices would be embedded with this same point. In the case of geometrical points, we usually do not want two vertices to coincide. However, for most applicative data such as colors, quantities or densities, it is common that the same value appears multiple time in the modeled object. Let us consider the example of the operator $col_{\langle 0\ 1\ 2 \rangle}$ that collects the face colors of the adjacent connected component. The evaluation of $col_{\langle 0\ 1\ 2 \rangle}(a)$ on the colored object of Figure 23(a) results in the multiset $[\![$ ●, ●, ●, ● $]\!]$ in which ● has two occurrences as it labels two faces. More generally, for all embedding π and all orbit type $\langle o \rangle$, we can define an operator $\pi_{\langle o \rangle}$ on node variables that collects the embedding values of the adjacent $\langle o \rangle$-orbit, regardless of embedding orbit sizes.

5.3. Terms and schemes

To sum up, node variables are available straightaway as they are the node identifiers of the left-hand side of the rule scheme L, and they will be substituted by nodes of the transformed G-map at rule scheme application. New embedding values are defined by terms upon these nodes with the introduced G-map operators: the embedding access $.\pi$, the neighbor access $.\alpha_i$, and the collect of orbit embeddings $\pi_{\langle o \rangle}$. Note that in addition to these operators, terms may include various operators and types provided by the user. For example, the translation scheme of Figure 21(a) uses the classical addition between a point and a vector, while the triangulation scheme of Figure 24(b) uses the operator *bary* that defines the barycenter of a point multiset. These operators and types provided by the user are referred in the following as the user signature.

As in Subsection 2.4, we define embedding terms on the user signature extended by the node variables (which dedicated type is denoted *Node* and their operators.

Definition 7 (Terms signature and rule schemes). *Let $\pi : \langle o \rangle \to \tau$ be an embedding operation of dimension n.*

Terms signature. *Let $\Omega_\pi = (S_\pi, F_\pi)$ be a* user signature *with S_π a set of type names including the π-type τ and F_π a set of functions defined on $S_\pi \cup S_\pi^\bullet$.*

$\Omega_{Map} = (S_{Map}, F_{Map})$ is the embedding term signature *extended on G-maps defined as $S_{Map} = S_\pi \cup \{Node\}$ and $F_{Map} = F_\pi \cup F_{Node}$ with F_{Node} the* set of function names *that contains :*

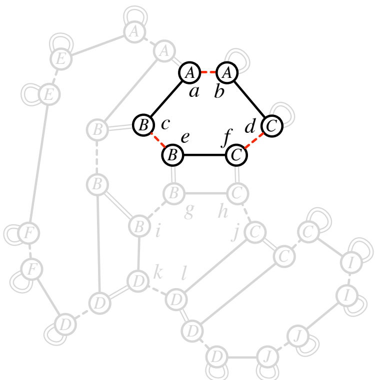
- *$.\pi : Node \to \tau$;*
- *$.\alpha_i : Node \to Node$ for all $i \in [0, n]$;*
- *$\pi_{\langle o' \rangle} : Node \to \tau^\bullet$ for all orbit type $\langle o' \rangle$ of dimension n.*

Graph schemes. *Let X be a set of* node variables. *A* graph scheme *$G = (V, E, s, t, \pi, \alpha)$ on (Ω_π, X) is a graph embedded on $\pi : \langle o \rangle \to T_{\Omega_{Map}}(X)_\tau$.*

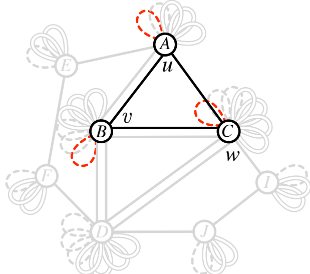
Rule schemes. *A* rule scheme *$r : L \hookleftarrow K \hookrightarrow R$ on Ω_π is a rule on graph schemes on (Ω_π, V_L) with V_L the node set of L.* ◀

For example, the triangulation rule scheme of Figure 23(b) on $\Omega_{\text{col}} = (S_{\text{col}}, F_{\text{col}})$ such that S_{col} includes the type *color* and F_{col} includes the operation $mix : color \times color \to color$. Similarly, the rule scheme of Figure 24(b) on $\Omega_{\text{pt}} = (S_{\text{pt}}, F_{\text{pt}})$ such that S_{pt} includes the type *point_2D* and F_{pt} includes the operation $bary : point\_2D^\bullet \to point\_2D$.

5.4. Evaluation of embedding terms



(a) Orbit $\langle 0\ 1 \rangle(a)$



(b) Quotient $\langle 0\ 1 \rangle(a)_{/\text{point}}$

Figure 25: Evaluation of the multiset of the face points

At rule scheme application, embedding terms have to be evaluated on the embedded G-map under transformation. For example, when the triangulation scheme of Figure 24(b) is applied to the top triangle Figure 24(a), the term $pt_{\langle 0\ 1 \rangle}(a)$ has to be evaluated by the point multiset $[\![A, B, C]\!]$ in order to compute the barycenter. The evaluation of terms on G-maps operators is defined in the following definition as an extension of the algebra provided by the user on the signature Ω_π of his/her sorts and functions. More precisely, given a π-embedded G-map and an Ω_π-algebra, we define the extended Ω_{Map}-algebra on embedding terms (see Section 2.4).

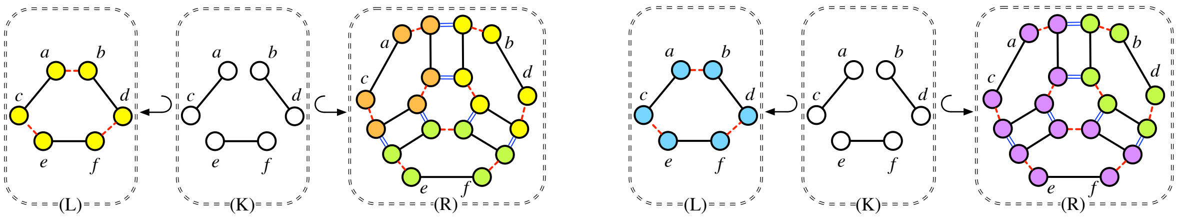
Definition 8 (Algebra extension by a G-map). *Let $G = (V, E, s, t, \pi, \alpha)$ be an n-G-map embedded on $\pi : \langle o \rangle \to \tau$, $\Omega_\pi = (S_\pi, F_\pi)$ be a user signature and an Ω_π-algebra $\mathcal{A}$.*

The extended algebra $\mathcal{A}_G$ from $\mathcal{A}$ by G is the Ω_{Map}-algebra defined as:

- *$(\mathcal{A}_{Map})_s = \mathcal{A}_s$ for $s \in S_\pi \cup S_\pi^\bullet$;*
- *$(\mathcal{A}_{Map})_{Node} = V$;*
- *for each f of F_π, $f^{A_{Map}} = f^A$;*
- *$.\pi^{A_{Map}}$ is the labeling function π;*
- *for all $i \in [0, n]$, for each node $v \in V$, there exists a unique i-arc $e \in E$ such that $s(e) = v$ and the $.{\alpha_i}^{\mathcal{A}_{Map}}$ function associates v to $t(e)$;*
- *for all orbit type $\langle o' \rangle$, for each node $v \in V$, let $G\langle o' \rangle(v)_{/\pi} = (V', E', s', t', \pi', \alpha')$ be the embedding quotient of the orbit graph, the ${\pi_{\langle o' \rangle}}^{\mathcal{A}_{Map}}$ function associates v to the label multiset of the orbit quotient [10] $[\![\pi'(v') \mid v' \in V']\!]$.* ◀

In particular, the evaluation of collect operators is defined with the graph quotient introduced in Definition 6. For example, to evaluate the term $pt_{\langle 0\ 1 \rangle}(a)$ for the object of Figure 25, we construct the quotient $\langle 0\ 1 \rangle(a)_{/\mathrm{pt}}$ of the orbit $\langle 0\ 1 \rangle(a)$. The term evaluation is then defined as the multiset of node labels of that quotient, *i.e.* $[\![A, B, C]\!]$.

Note that an algebra extension from a G-map is well defined. Especially, thanks to G-maps constraints, one node is the source of one and only one i-arc and so $.\alpha_i^{A_G}$ is a well defined function. As a consequence, collect operators are also well defined functions.



(a) (b)

Figure 26: Two evaluations of the rule scheme Figure 23

To be evaluated, a scheme only requires a kernel match as described in Subsection 2.3. In our case, we will use a match morphism of the topological structure of the left-hand-side $m : L_\alpha \to G$ in order to remove variable occurrences with node labels, while still properly matching the structure thanks to arc labels. Practically, the node matching part m_V of this morphism will be directly used for the substitution $\sigma : X \to V_G$. For example, an identity match morphism between the rule scheme and the object of Figure 23 assigns the variables a, b and e to the nodes a, b and e of the object, resulting in the rule Figure 26(a). Similarly, the rule of rule Figure 26(b) result from a match morphism assigning those nodes to the nodes i, g and l of the object.

Definition 9 (Rule scheme evaluation). *Let G be a π-embedded G-map, Ω_π a user signature and $\mathcal{A}$ an Ω_π-algebra.*

Graph scheme evaluation. *Let $S = (V, E, s, t, \pi, \alpha)$ be a graph scheme on (Ω_π, X) and $\sigma : X \to V_G$ an assignment of X. The* evaluated graph *$S^\sigma = (V, E, s, t, \pi^\sigma, \alpha)$ of S along σ is the π-embedded graph such as $\pi^\sigma(v) = \pi(\sigma(v))$ for each node $v \in V$.*

Rule scheme evaluation. *Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule scheme on Ω_π and $m : L_\alpha \to G$ a kernel match morphism. The* evaluated rule *of r along m is the π-embedded rule $r^{m_V} : L^{m_V} \hookleftarrow K^{m_V} \hookrightarrow R^{m_V}$.* ◀

[10] We note $[\![\pi'(v') \mid v' \in V']\!]$ the multiset of type $\tau^\bullet$ such that for all $x : \tau$, the multiplicity of x is equal to the number of node of V' labeled by x.

6. Rule scheme instantiation

In this section, we define how rule schemes are instantiated without considering the consistency preservation which is postponed to Section 7.

6.1. Need for simplicity

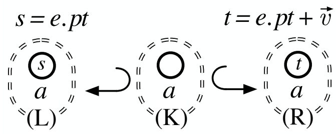
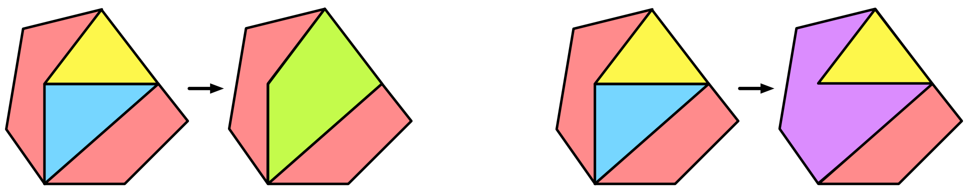


Figure 27: Expected rule scheme of the translation

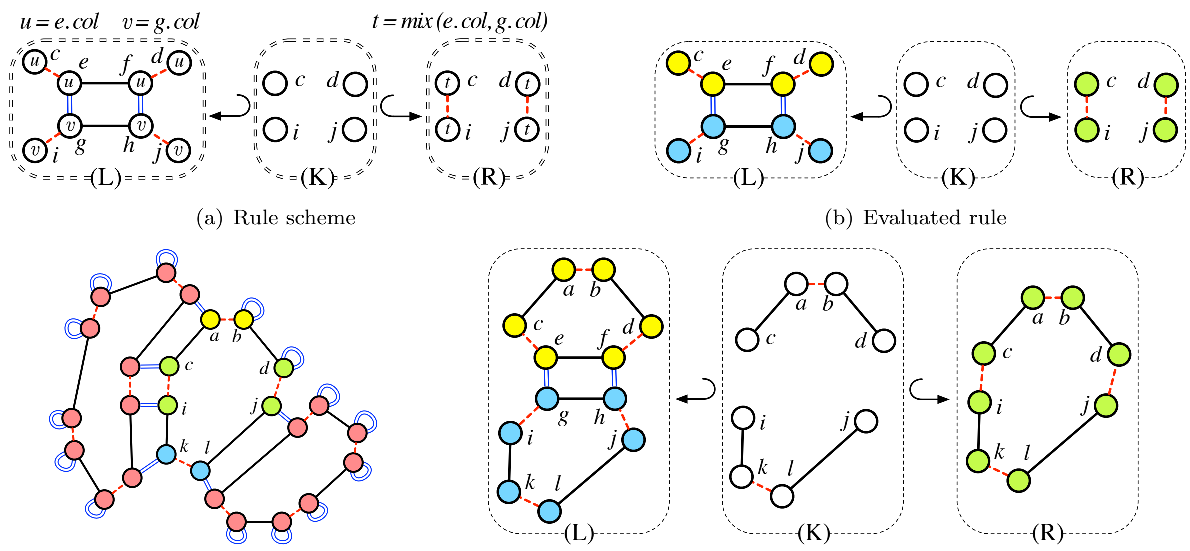
So far, every considered operation has been defined in relation to the specific topological structure of the transformed object. This problem was illustrated in Section 4.2 by the rule of Figure 18 which specifically defines the translation for a vertex adjacent to three edges. This is very restrictive and counter-intuitive from a user-end perspective. On a semantic level, the translation of a vertex has a single meaning, independent from the number of adjacent edges. A user friendly rule scheme should be as simple as in Figure 27 in which a single node relabeling encodes a single embedding transformation.



(a) (b)

Figure 28: Edge removal on the color embedding

Let us take a more significant example with the edge removal of Figure 28. This operation that will be the red line of this section involves both topological and embedding modifications: on the topological aspect, the edge is removed and the two adjacent faces are merged; on the embedding aspect, the color of the resulting face is obtained by mixing the colors of the two original faces.



(a) Rule scheme (b) Evaluated rule

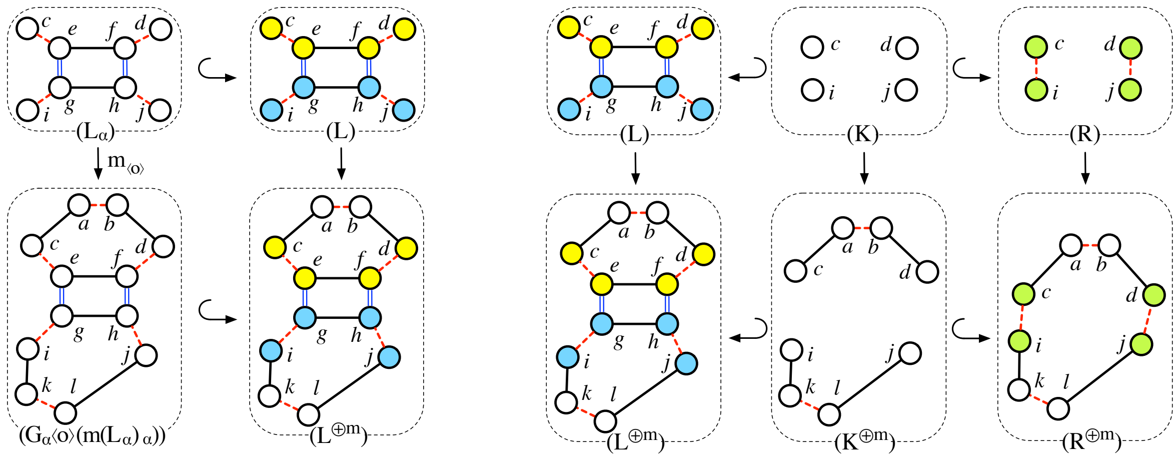
(c) Inconsistent application (d) Correct rule

Figure 29: Rule scheme of the edge removal and its evaluation

Semantically, this operation does not depend on the configurations of the two faces and should be defined by the simple rule scheme of Figure 29(a). But similarly to the translation, the application of the evaluated rule of Figure 29(b) to the object of Figure 23(a) results in the inconsistent object of Figure 29(c). Indeed, the embedding modifications must be propagated to all nodes of the two faces in order to preserve the G-map consistency.

Therefore, it is the task of the instantiation process to extend the evaluated rule to propagate the embedding modification. In our example, the evaluated rule of Figure 29(b) has to be extended into the correct rule of Figure 29(d). This process is divided into two steps: the topological extension that matches all required nodes and the embedding propagation that ensures their consistent relabeling.

6.2. Topological extension



(a) Extension of the left-hand side (b) Application of the evaluated rule

Figure 30: Construction of the topological extension

Intuitively, the topological extension uses the match morphism to complete the partial embedding orbits defined by the the evaluated rule with the actual full orbits of the transformed G-map. First, the extension $L^{\oplus m}$ of the left-hand side is computed in Figure 30(a) by pushout between the topological structure of the $\langle o\rangle$-orbit adjacent to the matched pattern $G_\alpha\langle o\rangle(m(L_\alpha)_\alpha)$, and the left-hand side of the evaluated rule L. The full extended rule $L^{\oplus m} \hookleftarrow K^{\oplus m} \hookrightarrow R^{\oplus m}$ is then computed in Figure 30(b) by application of the evaluated rule on the extended left-hand side.

Definition 10 (Topological extension). *Let $\pi : \langle o\rangle \to \tau$ be an embedding operation and $m : L_\alpha \to G$ be a kernel morphism on a π-embedded G-map G for a rule $r : L \hookleftarrow K \hookrightarrow R$.*

Let $L^{\oplus m}$ be the result of the pushout between $m_{\langle o\rangle} : L_\alpha \to G_\alpha\langle o\rangle(m(L_\alpha)_\alpha)$, the restriction of m to the topological structure of the $\langle o\rangle$-orbit adjacent to the matched pattern, and the inclusion $L_\alpha \hookrightarrow L$:

$$\begin{tikzcd}
L_\alpha \arrow[r, hook] \arrow[d, "m_{\langle o\rangle}"'] & L \arrow[d, "m'"] \\
G_\alpha\langle o\rangle(m(L_\alpha)_\alpha) \arrow[r, hook] & L^{\oplus m}
\end{tikzcd}$$

The topological extension *of r along the match morphism m is the rule $r^{\oplus m} : L^{\oplus m} \hookleftarrow K^{\oplus m} \hookrightarrow R^{\oplus m}$ defined by the following direct transformation:*

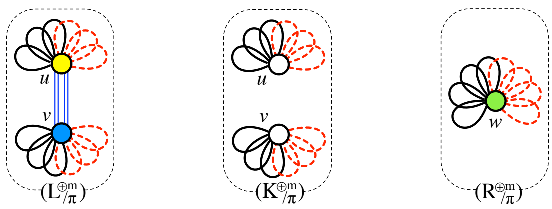
$$\begin{tikzcd}
L \arrow[d, "m'"'] & K \arrow[l, hook'] \arrow[r, hook] \arrow[d] & R \arrow[d] \\
L^{\oplus m} & K^{\oplus m} \arrow[l, hook'] \arrow[r, hook] & R^{\oplus m}
\end{tikzcd}$$

◀

Note that the pushout construction of $L^{\oplus m}$ is well founded since the morphisms $L_\alpha \hookrightarrow L$ and $m : L_\alpha \to G$ meet the conditions given in [14] ensuring the existence of pushouts. Also, note that the resulting rule of Figure 30 would still produce the inconsistent result of Figure 29(c) as extended parts' nodes are not relabeled.

6.3. Embedding propagation

The final step of rule scheme instantiation consists in propagating node labels of the extended rule. For example, for the extended rule of Figure 30(b), node labels have to be propagated in order to obtain the final of rule Figure 29(d). This step is a direct application of the quotient representation. For all graphs of the extended rule, each node is relabeled with the label of its images in the quotient graph. For example, in Figure 31 the three quotient graphs allow the embedding propagation of the extended rule of Figure 30(b) - *e.g.* node a unlabelled in $L^{\oplus m}$ can be labelled with the label of its image u in $L^{\oplus m}_{/\pi}$.



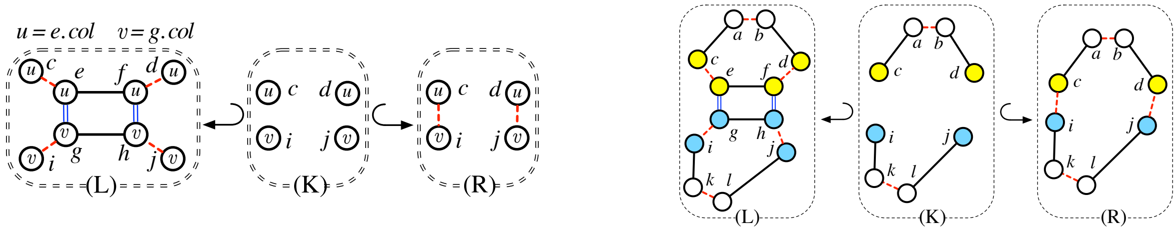
(a) (b) (c)

Figure 31: Quotients for the embedding propagation

Definition 11 (Embedding propagation). *Let $G=(V,E,s,t,\pi,\alpha)$ be a graph embedded on $\pi:\langle o\rangle\to\tau$ such that G satisfies the embedding consistency constraint and $q : G \to G_{/\pi}$ the quotient morphism with $G_{/\pi} = (V_{/\pi}, E_{/\pi}, s_{/\pi}, t_{/\pi}, \pi_{/\pi}, \alpha_{/\pi})$.*

The π-embedding propagation *of G is the π-embedded graph $G^{\odot\pi} = (V, E, s, t, \pi', \alpha)$ such for each node $v \in V, \pi'(v) = \pi_{/\pi}(q_V(v))$.*

For $r : L \hookleftarrow K \hookrightarrow R$ an n-topological π-embedded rule, we note note $r^{\odot\pi}$ the rule $L^{\odot\pi} \hookleftarrow K^{\odot\pi} \hookrightarrow R^{\odot\pi}$. ◀



(a) Rule scheme (b) Extended rule

Figure 32: Inconsistent edge removal

Note that as the quotient existence depends on the satisfaction of the embedding consistency constraint, the embedding propagation only applies to rules for which all parts satisfy the constraint. The extended patterns must contain only one label value per embedding orbit in order for their quotient representation to preserve these unique labels. Let us consider the counterexample of Figure 32. The rule scheme defines the edge removal without consistently relabeling the face colors and therefore the face can be labeled by two different colors in the right-hand side of the extended evaluated rule. As this prevents the quotient existence, the embedding propagation cannot be applied.

Moreover, the satisfaction of the embedding consistency constraint by all parts of a rule scheme does not entail the same property on its extended rule, because of the overlap phenomenon detailed in the next section along the rule scheme conditions of embedding consistency preservation. Therefore, the instantiation definition given in the next subsection depends on the extended rule satisfaction of the embedding consistency constraint but rule scheme conditions will ensure it.

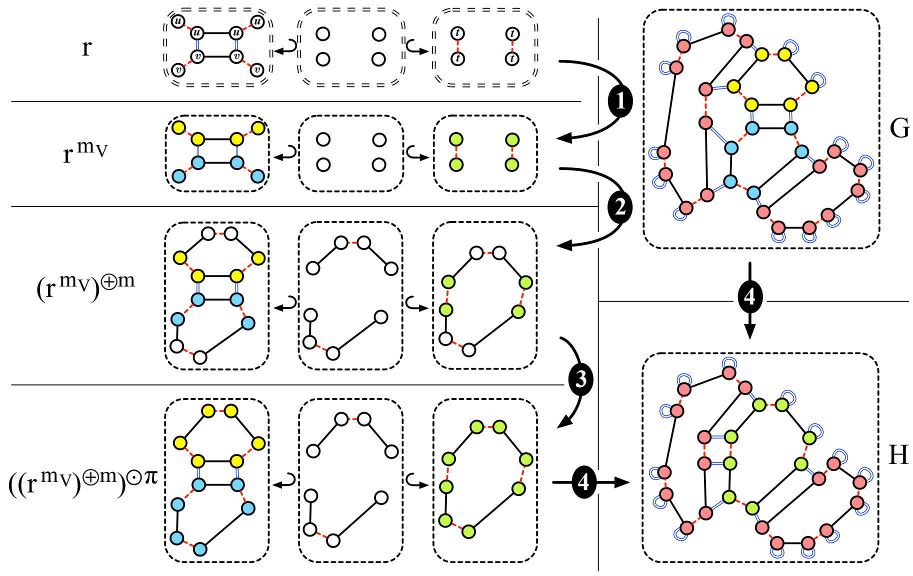


Figure 33: Rule scheme application

Regardless of consistency preservation, the application of a rule scheme r to an object defined as an embedded G-map G along a kernel match morphism m consists in four steps sketched out in Figure 33:

1. the evaluation r^{m_V} of the rule scheme r along m_V to substitute node variables by nodes of G;
2. the topological extension $(r^{m_V})^{\oplus m}$ along m of the evaluated rule r^{m_V};
3. the embedding propagation $((r^{m_V})^{\oplus m})^{\odot\pi})$ along the extended rule $(r^{m_V})^{\oplus m}$;
4. at last the application of the final rule $((r^{m_V})^{\oplus m})^{\odot\pi})$ on the targeted object G.

Note that as the embedding propagation existence depends on the satisfaction of the embedding consistency constraint by all parts of the extended rule. This will be ensured by conditions on rule schemes provided in Section 7 to preserve G-map consistency. Therefore, rule schemes satisfying those conditions can always be instantiated for any kernel match morphism.

Definition 12 (Instantiation and application of rule scheme). *Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule scheme on a user signature Ω_π, and $m : L_\alpha \to G$ a kernel match morphism on a π-embedded G-map G.*

Let $r^{m_V} = L^{m_V} \hookleftarrow K^{m_V} \hookrightarrow R^{m_V}$ be the evaluation of r along m (Definition 9).

Let $(r^{m_V})^{\oplus m}$ be the topological extension of r^{m_V} along[11] *m (Definition 10).*

If all parts of $(r^{m_V})^{\oplus m}$ satisfy the embedding consistency constraint, let $((r^{m_V})^{\oplus m})^{\odot\pi}$ be the π-embedding propagation of $(r^{m_V})^{\oplus m}$ (Definition 11).

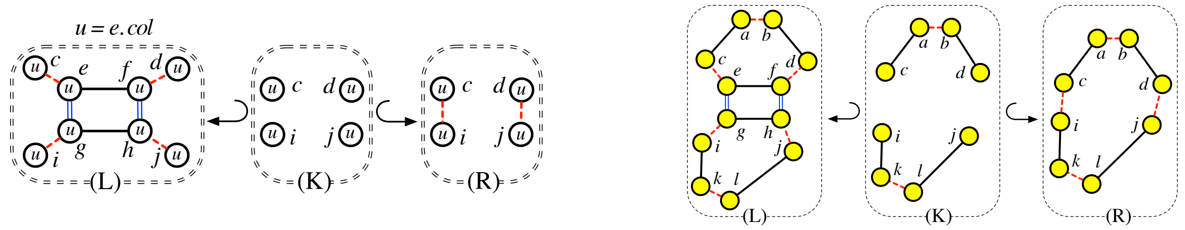
The instantiation *of r along m is $((r^{m_V})^{\oplus m})^{\odot\pi}$ denoted $r^m : L^m \hookleftarrow K^m \hookrightarrow R^m$.*

If there exists a morphism $L^m \to G$ extending m, the application *of r to G along m denoted by $G \Rightarrow_{r,m} H$ is defined by the direct transformation $G \Rightarrow_{r^m,\, L^m \to G} H$.* ◀

Finally, note that similarly to the approach of [15] recalled in Subsection 2.3, the substitution given by the kernel match morphism can not always result in an extended full match of the instantiated rule.

Let us take an example with the operation of edge removal of Figure 34. This time, the rule scheme of Figure 34(a) remove an edge between two faces of the same color *e.col*. The instantiation of the rule scheme along the identity morphism on the object G of Figure 33 results in the rule of Figure 34(b)

[11] Thanks to the definition 9 of graph scheme evaluations, $L_\alpha^{m_V} = L_\alpha$ thus the rule can be directly extended along m.



(a) Rule scheme (b) Instantiated rule

Figure 34: Edge removal between two faces of same color

where the term $e.col$ has been evaluated by yellow. As the extension process rests on the kernel match, the rule can always be extended regardless of the matched object labeling. However, the resulting rule can obviously not be applied to the object as an application match morphism can not be induced because nodes g, h, i and j of the object are blue.

7. Consistency preservation

This section establishes and proves the conditions on rule schemes that ensure the preservation of G-map constraints. Subsection 7.1 adresses the topological consistency while Subsections 7.2 and 7.3 focus on the embedding consistency. More precisely, we show that rule schemes that satisfy some given conditions can always be instantiated and that the instantiated rules satisfy the original conditions of embedding consistency preservation of Definition 2.

7.1. Topological consistency preservation

As the topological extension is the only part of the instantiation that transforms the rule topological structure, let us show that it preserves the conditions of topological consistency preservation of Theorem 1.

Lemma 1 (Topological consistency preservation of topological extension). *Let $r: L \hookleftarrow K \hookrightarrow R$ be a rule embedded on $\pi: \langle o \rangle \to \tau$ and $m: L_\alpha \to G$ a kernel match morphism on a π-embedded G-map G.*

If r satisfies the conditions of topological consistency preservation of Theorem 1, then the topological extended rule $r^{\oplus m}$ also satisfies these conditions.

► ***Proof.*** Let us show the three conditions of topological consistency preservation.

Non-orientation

Because an n-G-map and its $\langle o \rangle$-orbits are non-oriented graphs, the part added by the topological extension is also non-oriented. And because L, K and R are non-oriented graphs, then $L^{\oplus m}$, $K^{\oplus m}$ and $R^{\oplus m}$ are also non-oriented graphs. Consequently, $r^{\oplus m}$ satisfies the non-orientation condition.

Adjacent arcs

As $K^{\oplus m}$ and $R^{\oplus m}$ are computed by application of r on $L^{\oplus m}$, all new nodes added by the topological extension step are preserved nodes of $K^{\oplus m}$. Consequently, those nodes are the sources of the same arcs with the same labels on both sides $L^{\oplus m}$ and $R^{\oplus m}$. Thus all nodes added by the topological extension satisfy the adjacent arc condition. And because r satisfies the adjacent arc condition, $r^{\oplus m}$ also does.

Cycle condition

As already mentioned, all nodes added by the topological extension are preserved nodes of $K^{\oplus m}$ and are the source of the same arcs with the same labels in $L^{\oplus m}$ and $R^{\oplus m}$. Thus, we have multiple cases to consider depending on what portion of a cycle belong to the extended part. Let us prove the three points of the cycle condition of Theorem 1 for all couple (i,j) such $0 \leq i \leq i+2 \leq j \leq n$:

• By definition of the topological extension, any added node v of $R^{\oplus m} \backslash K^{\oplus m}$ comes from $R \backslash K$. And as the rule r satisfies the cycle condition, v is the source of an $ijij$-cycle in R and also in $R^{\oplus m} \backslash K^{\oplus m}$.
• If v is a preserved node of $K^{\oplus m}$ and is the source of an $ijij$-cycle in $L^{\oplus m}$, then either:

- If v is source of a $ijij$-cycle in L, because the rule r satisfies the cycle condition, v the source of an $ijij$-cycle in R, and so in $R^{\oplus m}$.
- If some of the four arcs come from L and some others have been added by the topological extension step. Then, due to the cycle condition on r, the old arcs of L are preserved in R, and thus also in $L^{\oplus m}$ and $R^{\oplus m}$. And, due to topological extension, new arcs are also preserved in $L^{\oplus m}$, $K^{\oplus m}$ and $R^{\oplus m}$. Thus, in this case, the preserved node v of $K^{\oplus m}$ is the source of an $ijij$-cycle in $L^{\oplus m}$, and is also the source of an $ijij$-cycle in $R^{\oplus m}$.
- If the four arcs are added by the topological extension step. Then, as previously, these new arcs are preserved in $L^{\oplus m}$, $K^{\oplus m}$ and $R^{\oplus m}$. And thus the preserved node v of $K^{\oplus m}$ is the source of an $ijij$-cycle in $L^{\oplus m}$, and is also the source of an $ijij$-cycle in $R^{\oplus m}$.

• Finally, if v is a preserved node of $K^{\oplus m}$ and is not the source of an $ijij$-cycle in $L^{\oplus m}$, then, as previously, the i-arc and the j-arc of source v can be either old arcs from r, or new arcs added during topological extension step. In both cases, these arcs are preserved in $R^{\oplus m}$, respectively due to cycle condition of r and topological extension. Consequently, the i-arc and the j-arc of source v are preserved in $R^{\oplus m}$.

Thus, $r^{\oplus m}$ satisfies the cycle condition.

Consequently, $r^{\oplus m}$ satisfies the conditions of topological consistency preservation of Theorem 1. ◀

This result can directly be extended to the whole rule instantiation.

Theorem 3 (Topological consistency preservation of instantiation). *Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule scheme on a user signature Ω_π, and $m : L_\alpha \to G$ a kernel match morphism on a π-embedded G-map G.*

If r satisfies the conditions of topological consistency preservation of theorem 1, then the instantiated rule $r^m = ((r^{m_V})^{\oplus m})^{\odot\pi}$, if it exists, also satisfies these conditions.

▶ ***Proof.*** As r^{m_V} has the same topological structure as r, r^{m_V} satisfies the conditions of topological consistency preservation. Then, according to Lemma 1, $(r^{m_V})^{\oplus m}$ also does. Finally, as the embedding propagation preserves the topological structure, $((r^{m_V})^{\oplus m})^{\odot\pi}$ satisfies these conditions. ◀

7.2. Case of overlap

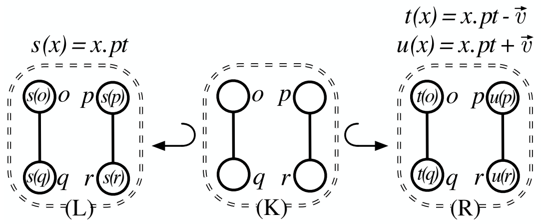
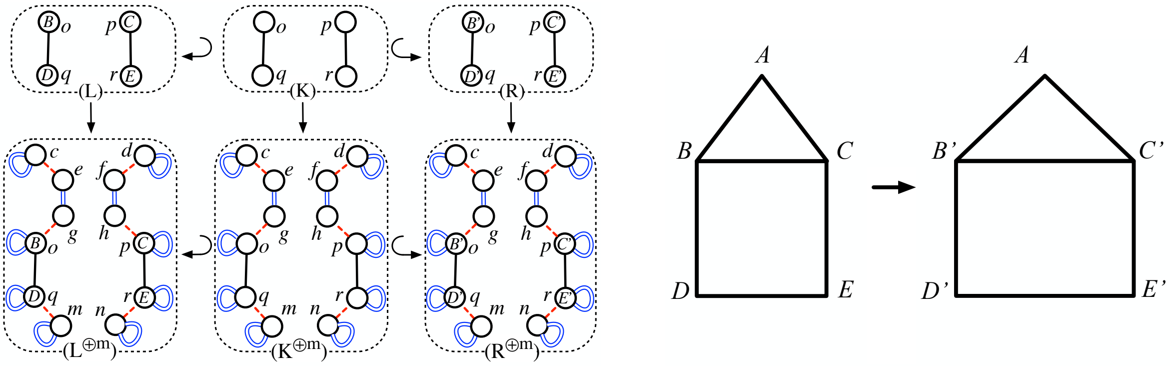


Figure 35: Face stretching rule scheme

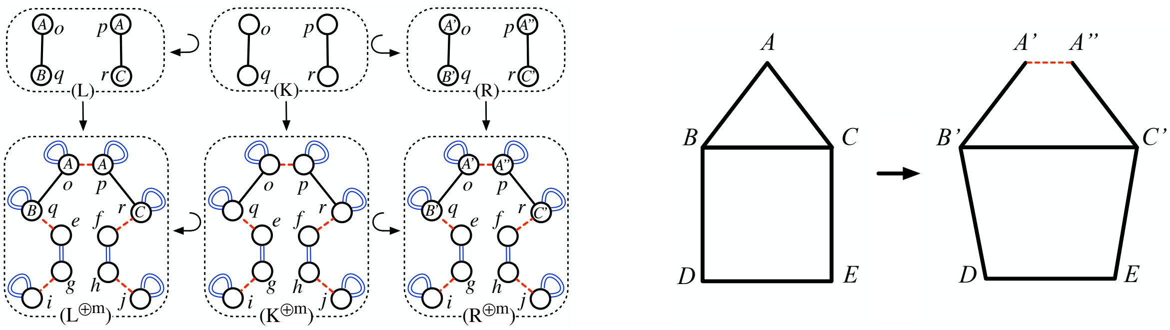
Before we study the embedding consistency preservation, we introduce a risk occurring with topological extension : the overlap of embedding orbits. By allowing a minimal match of the transformed embeddings that relies on the automatic completion of transformed embedding orbits, we are exposed to unexpected merges of different embedding orbits. Let us consider the face stretching defined by the rule scheme of Figure 35. The operation consists in matching two edges to translate their vertices in the two opposed directions $\vec{v}$ and $-\vec{v}$.



(a) Consistent extended rule (b) Intuition

Figure 36: Consistent face streching

When the rule scheme is correctly applied to the square face, the extended rule of Figure 36 contains four vertices in R respectively embedded by $B' = B - \vec{v}$, $D' = D - \vec{v}$, $C' = C + \vec{v}$ and $E' = E + \vec{v}$. Conversely, when the rule is applied to the triangle face, the extended rule of Figure 37 is inconsistent as the top vertex ends up embedded in R with two different values $A' = A - \vec{v}$ and $A'' = A + \vec{v}$. This is a clear case of misapplication as we wanted to match and translate four vertices but only match three. We call an overlap such a situation where different embedding orbits manipulated in the rule end up merged in the extended rule and we define a condition on the kernel morphism that prevent it. This condition can be seen as an extension of the injective condition on the match morphism to the embedding orbits.



(a) Inconsistent extended rule (b) Intuition

Figure 37: Inconsistent face streching

Lemma 2 (Non-overlap). *Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule embedded on $\pi : \langle o \rangle \to \tau$ and $m : L_\alpha \to G$ a kernel match morphism on a π-embedded G-map G.*

We say that the topological extension of r along m produces an overlap *if for v and w two nodes of L (resp. K, R) such that $v \not\equiv_{L\langle o \rangle} w$ (resp. $v \not\equiv_{K\langle o \rangle} w$, $v \not\equiv_{R\langle o \rangle} w$) then $v \equiv_{L^{\oplus m}\langle o \rangle} w$ (resp. $v \equiv_{K^{\oplus m}\langle o \rangle} w$, $v \equiv_{R^{\oplus m}\langle o \rangle} w$).*

The topological extension of r along m does not produce overlap if m satisfies the following condition of non-overlap*: for two nodes v and w of L such as $v \not\equiv_{L\langle o \rangle} w$, $m(v) \not\equiv_{G\langle o \rangle} m(w)$.*

▶ ***Proof.*** Let us show that $L^{\oplus m}$ does not contain overlap. Let suppose that there exist v and w two nodes of L such that $v \not\equiv_{L\langle o \rangle} w$ and $v \equiv_{L^{\oplus m}\langle o \rangle} w$. Then, the overlap comes from the topological extension, *i.e* the node images $m(v)$ and $m(w)$ belong to the same orbit in G, $m(v) \equiv_{G\langle o \rangle} m(w)$. This is contrary to the condition of non-overlap. The proof is similar for $K^{\oplus m}$ and $R^{\oplus m}$. ◀

7.3. Embedding consistency preservation

We now study how the non-overlap condition combined with the conditions of embedding consistency preservations on evaluated rule schemes ensure that the instantiated rules satisfy the conditions of embedding consistency preservation on rules given in Theorem 2. In particular, we will release the condition of full match of transformed embeddings as it was the goal of the automatic orbit completion of transformed embeddings.

We start with the topological extension step. Note that as the nodes added by the topological extension are not labeled, the extended rule is only expected to satisfy a weak version of the full match of transformed embeddings of Theorem 2 that does not require a total labelling of the orbit.

Lemma 3 (Embedding consistency preservation of topological extension). *Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule embedded on $\pi : \langle o \rangle \to \tau$ and $m : L_\alpha \to G$ a kernel match morphism on a π-embedded G-map G.*

If r satisfies the conditions of topological consistency preservation of Theorem 1, the conditions of embedding consistency and of labeling of extended embedding orbits of Theorem 2, and if m satisfies the condition of non-overlap of Lemma 2, then the topological extended rule $r^{\oplus m} : L^{\oplus m} \hookleftarrow K^{\oplus m} \hookrightarrow R^{\oplus m}$ satisfies the following conditions:

- Embedding consistency *of Theorem 2: L, K and R satisfy the embedding consistency constraint of Definition 5.*
- Weak full match of transformed embeddings*: if a preserved node v has a transformed embedding, then $R^{\oplus m}\langle o \rangle(v)$ is a full orbit;* i.e. *if v is a node of $K^{\oplus m}$ such that $\pi_{L^{\oplus m}}(v) \neq \pi_{R^{\oplus m}}(v)$, then every node of $R^{\oplus m}\langle o \rangle(v)$ is the source of exactly one i-arc for each i of $\langle o \rangle$.*
- Labeling of extended embedding orbits *of Theorem 2: if v is a node of K and there exits a node w in $R\langle o \rangle(v)$ such that w is not in $L\langle o \rangle(v)$, then there exist v' in K with $v' \equiv_{L\langle o \rangle} v$ and $v' \equiv_{R\langle o \rangle} v$ such that $\pi_L(v') \neq \perp$.* ◀

▶ ***Proof.*** Let us show the three conditions.

Embedding consistency Let v and w be two nodes of $L^{\oplus m}$ such that $v \equiv_{L^{\oplus m}\langle o \rangle} w$, $\pi_{L^{\oplus m}}(v) \neq \perp$ and $\pi_{L^{\oplus m}}(w) \neq \perp$. Because of the condition of non-overlap, v and w are two nodes of L such $v \equiv_{L\langle o \rangle} w$. As L satisfies the embedding consistency constraint, $\pi_L(v) = \pi_L(w)$ and therefore $\pi_{L^{\oplus m}}(v) = \pi_{L^{\oplus m}}(w)$. The proof is the same for K and R. $r^{\oplus m}$ satisfies the embedding consistency condition.

Weak full match of transformed embedding.

Let v be a node of $R^{\oplus m}$. If v is a preserved node of $K^{\oplus m}$ or an added node of $R^{\oplus m}$, thanks to topological extension step, $R^{\oplus m}\langle o \rangle(v)$ is a complete orbit. Thus v is the source of exactly one i-arc for each i of $\langle o \rangle$. Then $r^{\oplus m}$ satisfies the weak full match of transformed embedding.

Labeling of extended embedding orbits. Let v be a node of $K^{\oplus m}$ and w a node $R^{\oplus m}\langle o \rangle(v)$ such that w is not in $L^{\oplus m}\langle o \rangle(v)$. Because the topological extension definition, w is a node of r. As r satisfies the labeling of extended embedding orbits, there exist v' in K with $v' \equiv_{L\langle o \rangle} v$ and $v' \equiv_{R\langle o \rangle} v$ such that $\pi_L(v') \neq \perp$. Moreover, because of the topological extension definition, $v' \equiv_{L^{\oplus m}\langle o \rangle} v$, $v' \equiv_{R^{\oplus m}\langle o \rangle} v$, and $\pi_{L^{\oplus m}}(v') \neq \perp$. Therefore, $r^{\oplus m}$ satisfies the labeling of extended embedding orbits. ◀

Let us now show that the embedding propagation step restores the original strong embedding consistency conditions.

Lemma 4 (Embedding consistency preservation of the embedding propagation). *Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule embedded on $\pi : \langle o \rangle \to \tau$ and $m : L_\alpha \to G$ a kernel match morphism on a π-embedded G-map G.*

If r satisfies the conditions of topological consistency preservation of Theorem 1 and the conditions of Lemma 3, then the embedding propagated rule $r^{\odot \pi}$ satisfies the conditions of topological consistency preservation of Theorem 1 and the conditions of embedding consistency preservation of Theorem 2.

► ***Proof.*** As previously said, the embedding propagation step does not modify the topological structure, thus this last step preserves the topological consistency conditions of Theorem 1.

In the same way, the conditions of embedding consistency preservation and of labeling of extended embedding orbits are preserved.

Moreover, the π-embedding propagation step propagates each embedding label along the full $\langle o \rangle$-orbit, the weak condition of full match of transformed embed becomes total as all nodes are become labeled. ◄

Finally, we can extend this result to the whole rule instantiation and show that it always exists if the following conditions of embedding consistency preservation are satisfied..

Theorem 4 (Embedding consistency preservation of instantiation). *Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule scheme on a user signature Ω_π and $m : L_\alpha \to G$ be a kernel match morphism on a π-embedded G-map G.*

The instantiated rule $((r^{m_V})^{\oplus m})^{\odot\pi}$ exists and satisfies the conditions of embedding consistency preservation of Theorem 2 if the following conditions are satisfied:

- *r satisfies the condition of embedding consistency of Theorem 2;*
- *r satisfies the condition of labeling of extended embedding orbits of Theorem 2;*
- *m satisfies the condition of non-overlap of Lemma 2.*

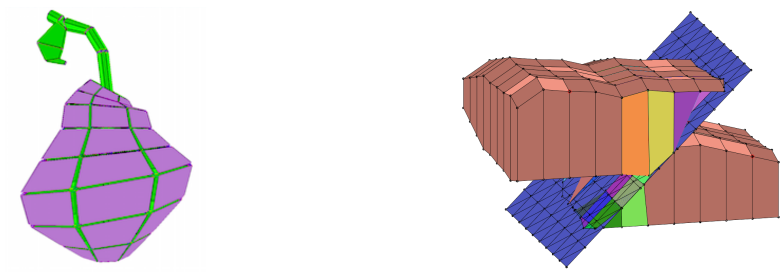
► ***Proof.*** As equal terms are evaluated by equal values, if r satisfies the previous conditions, so does the evaluated rule r^{m_V}. Then, the extended rule $(r^{m_V})^{\oplus m}$ satisfies the condition of Lemma 3, including embedding consistency. Therefore, the propagation $((r^{m_V})^{\oplus m})^{\odot\pi}$ exists. Finally, according to Lemma 4, the instanciated rule $((r^{m_V})^{\oplus m})^{\odot\pi}$ satisfies the conditions of embedding consistency preservation. ◄

Let us note that the properties of Theorem 4 are sufficient but not necessary to ensure the embedding consistency preservation. In practice, it may be useful to relax the embedding consistency condition if several terms can have the same evaluation. For example, algebraic properties of user-defined functions on embeddings could be taken into account.

8. Applications and related works

8.1. Applications

The language of rules introduced in this article has been implemented in the tool set Jerboa [26, 32] that allows a geometric modeler kernel to be generated from a set of rules. It includes a graphical editor that allows both an easy modeler characterization and a fast graphical design of its operations assisted by static verification steps. When the design is over, the Jerboa library produces a full featured modeler kernel that can be used in a final application. Moreover, the generated kernel is highly reliable as rules take benefit from graph transformation techniques to ensure consistency properties. Jerboa has been successfully used in other works, especially for the adaptation of L-Systems with G-map [33] or in a geomodeling context [34] (see Figure 38).

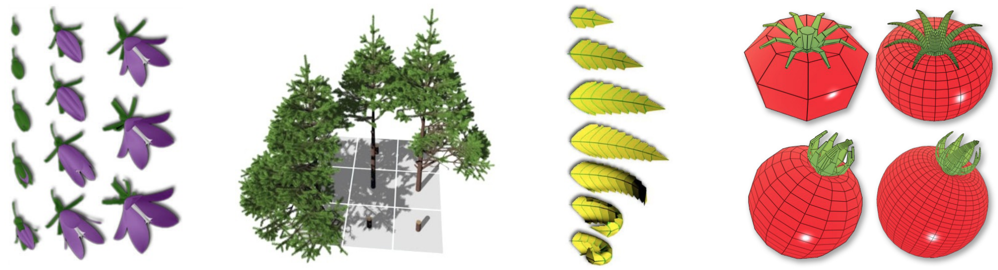


(a) L-System based plant growth (b) Geomodeling of a fault

Figure 38: Two applications of Jerboa

Note that within Jerboa, rule schemes allow to use simultaneously the topological variables (see Subsection 3.3) and the embedding terms introduced in this article. This allows for example to generically define the geometric triangulation of a face whatever its topological size or its geometric shape. Moreover, modeled objects can have several embedding data types (*e.g.* colored polyhedral 2D objects) and therefore multiple expressions can label rule nodes to define the embedding transformation. This multiple embedding aspect has been addressed in [13] by introducing a category of partially I-labeled graphs that handle multiple node labels as an extension of the category defined in [14]. In this category, each kind of embedding (node label) is defined by its own node labeling function, specified on its own orbit type. We extended graph transformations to this category so rules could simultaneously transform multiple embedding.

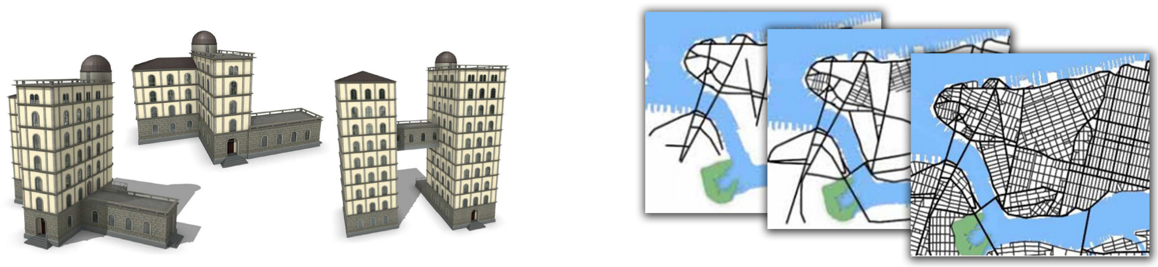
8.2. Related works



(a) Bluebell growth (b) Pines (c) Leaf grothw (d) Fruit structure

Figure 39: Plants modeling based on L-Systems

Formal rule languages have already been used for twenty years in the context of geometric modeling. In particular, L-systems [35] introduced by the biologist Aristid Lindenmayer to model plant growth have been developed for many procedural applications. L-Systems are based on iterated applications of a set of rules until a stop condition is satisfied. Hence, they are suited to represent arborescent structures, like the flowers in Figure 39(a) [36], or the trees in Figure 39(b) [37]. Moreover, L-systems have already been used in a topological-based context in [38] to model leaf growth as in Figure 39(d), in [39] to model flowers, or in [40] to model internal structure of fruits as in Figure 39(d).



(a) Buildings modeling based on shape grammars (b) Street creation system applied to Manhattan

Figure 40: Urban modeling based on L-Systems

Inspired by L-systems, [41] introduced a new type of grammar dedicated to automatically model buildings. In the same way than plants, building designs are derived using grammars that define building shapes. For example, the three buildings in Figure 40(a) from [42] are generated from the same shape grammar. Moreover, 3D models of existing buildings can also be generated from aerial pictures [43] in order to be displayed in navigation applications. For the same purpose, L-systems have also been extended to automatically model street network of cities as represented in Figure 40(b) from [44].

In all these applications, L-systems and graph grammars are defined by a limited set of high level operations such as creating a new branch, inserting a floor or subdividing a district with a road. Conversely to graph transformations, L-system rules do not contain all the low level transformations on the actual object representation and therefore each of these high level operations has then its own dedicated implementation and consistency preservation mechanism. This lack of adaptability and robustness entails that every additional rule requires additional implementation and debugging efforts, which we avoid using graph transformations.

9. Conclusion

In this article, we introduce a new kind of graph transformation variables, called node variables and inspired from the attribute variables of [15], and dedicated to embedding computations in the context of topology-based geometric modeling. Benefiting from the regularity of G-map topological structures, these node variables are provided with operators that allow both to access the existing embedding (node labels in the transformed object) and to traverse the topological structure (neighboring nodes in the transformed object). A rule instantiation mechanism is also provided to propagate the embedding modifications to the concern orbits of the object. The resulting language is generic enough to define any usual embedding transformation, and it is fitted with syntactic conditions that allow an operation implemented as a rule to be statically checked. A single rule application engine may thus be programmed to handle any operation.

Our further work will consist in enhancing the language with new possibilities, while still providing a safe theoretical ground. In particular, we still have to show under which syntactic conditions node variables can be simultaneously used with the orbit variables dedicated to topological transformations in order to define operations independently from both embedding value and topological shape (*e.g.* the triangulation of any sized face of any color). Furthermore, we wish to provide rule scripts in order to compute complex modeling operations, as the boolean operators allowing to combine objects together by intersection, difference or union. Such operations require to search the object and selectively apply rules, following a given strategy. A script language would allow to define these strategies by providing operators such as iterators, loops or conditionals.

References

[1] L. Baresi, R. Heckel, Tutorial introduction to graph transformation: A software engineering perspective, in: Graph Transformation, Springer, 2002, pp. 402–429.

[2] H. Ehrig, G. Rozenberg, H.-J. Kreowski, U. Montanari, Handbook of Graph Grammars and Computing by Graph Transformation: Concurrency, Parallelism, and Distribution, Vol. 3, World Scientific, 1999.

[3] G. Taentzer, Agg: A graph transformation environment for modeling and validation of software, in: Applications of Graph Transformations with Industrial Relevance, Springer, 2004, pp. 446–453.

[4] M. Gyssens, J. Paredaens, J. Van den Bussche, D. V. Gucht, A graph-oriented object database model, Knowledge and Data Engineering, IEEE Transactions on 6 (4) (1994) 572–586.

[5] P. Lienhardt, Topological models for boundary representation: a comparison with n-dimensional generalized maps, Computer-Aided Design 23 (1) (1991) 59–82.

[6] M. Poudret, A. Arnould, J.-P. Comet, P. Le Gall, Graph transformation for topology modelling, in: 4th International Conference on Graph Transformation (ICGT'08), Vol. 5214 of LNCS, Springer, Leicester, United Kingdom, 2008, pp. 147–161.

[7] A. Spicher, O. Michel, J.-L. Giavitto, Declarative mesh subdivision using topological rewriting in mgs, in: Graph Transformations, Springer, 2010, pp. 298–313.

[8] C. Smith, P. Prusinkiewicz, F. Samavati, Local specification of surface subdivision algorithms, in: Applications of graph transformations with industrial relevance, Springer, 2004, pp. 313–327.

[9] P. Lienhardt, N-dimensional generalised combinatorial maps and cellular quasimanifolds, International Journal on Computational Geometry and Applications (IJCGA) 4 (3) (1994) 275–324.

[10] G. Damiand, P. Lienhardt, Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing, A K Peters/CRC Press, 2014.

[11] M. Poudret, J.-P. Comet, P. L. Gall, A. Arnould, P. Meseure, Topology-based geometric modelling for biological cellular processes, in: 1st International Conference on Language and Automata Theory and Applications (LATA 2007), Tarragona, Spain, 2007.

[12] M. Poudret, Transformations de graphes pour les opérations topologiques en modélisation géométrique, application à l'étude de la dynamique de l'appareil de golgi, Thèse, Université d'Évry val d'Essonne, Programme Epigénomique (Oct. 2009).

[13] T. Bellet, A. Arnould, P. Le Gall, Rule-based transformations for geometric modeling., in: 6th International Workshop on Computing with Terms and Graphs (TERMGRAPH 2011), Saarbrucken, Germany, 2011.

[14] A. Habel, D. Plump, Relabelling in graph transformation, in: A. Corradini, H. Ehrig, H.-J. Kreowski, G. Rozenberg (Eds.), ICGT, Vol. 2505 of LNCS, Springer, 2002, pp. 135–147.

[15] B. Hoffmann, Graph transformation with variables, Formal Methods in Software and System Modeling 3393 (2005) 101–115.

[16] H. Ehrig, Introduction to the algebraic theory of graph grammars (a survey), in: Graph-Grammars and Their Application to Computer Science and Biology, Springer, 1979, pp. 1–69.

[17] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, M. Löwe, Algebraic approaches to graph transformation. Part I: basic concepts and double pushout approach, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1997.

[18] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer, Fundamentals of Algebraic Graph Transformation (Monographs in Theoretical Computer Science. An EATCS Series), Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2006.

[19] B. Hoffmann, E. Jakumeit, R. Geiß, Graph rewrite rules with structural recursion, in: Workshop on Graph Computation Models, Leicester, UK, 2008.

[20] F. Orejas, L. Lambers, Symbolic attributed graphs for attributed graph transformation, Electronic Communications of the EASST 30.

[21] F. Orejas, L. Lambers, Lazy graph transformation, Fundamenta Informaticae 118 (1-2) (2012) 65–96.

[22] F. Drewes, B. Hoffmann, D. Plump, Hierarchical graph transformation, Journal of Computer and System Sciences 64 (2) (2002) 249–283.

[23] B. Hoffmann, More on graph rewriting with contextual refinement, GCM 2014.

[24] A. Habel, Hyperedge replacement: grammars and languages, Vol. 643, Springer Science & Business Media, 1992.

[25] A. Habel, H. Radke, Expressiveness of graph conditions with variables, Electronic Communications of the EASST 30.

[26] T. Bellet, M. Poudret, A. Arnould, L. Fuchs, P. Le Gall, Designing a topological modeler kernel: a rule-based approach, in: Shape Modeling International (SMI'10) Shape Modeling International (SMI'10), Aix-en-Provence, France, 2010.

[27] V. Lang, P. Lienhardt, Simplicial sets and triangular patches, in: IEEE (Ed.), Computer Graphics International, 1996.

[28] M. Mantyla, Introduction to Solid Modeling, W. H. Freeman & Co., New York, NY, USA, 1988.

[29] K. Weiler, The radial edge structure: A topological representation for non-manifold geometric boundary modeling, in: Geometric Modeling for CAD Applications: Selected Papers from Ifip Wg 5.2, Elsevier Science, 1988, pp. 3–36.

[30] F. Ledoux, A. Arnould, P. L. Gall, Y. Bertrand, Geometric modelling with CASL, in: Recent Trends in Algebraic Development Techniques, 15th International Workshop (WADT 2001), Vol. 2267 of LNCS, Springer, 2001, pp. 176–200.

[31] T. Bellet, Transformation de graphes pour la modélisation géométrique à base topologique, Ph.D. thesis, Université de Poitiers (2012).

[32] H. Belhaouari, A. Arnould, P. Le Gall, T. Bellet, JERBOA: A graph transformation library for topology-based geometric modeling, in: 7th International Conference on Graph Transformation (ICGT 2014), Vol. 8571 of LNCS, Springer, York, UK, 2014.

[33] V. Gauthier, H. Belhaouari, A. Arnould, Évaluation de la modélisation à base de transformation de graphes avec Jerboa, in: Journées de l'Association Française Graphique (AFIG), Lyon, 2015.

[34] V. Gauthier, A. Arnould, H. Belhaouari, S. Horna, M. Perrin, M. Poudret, J. Rainaud, A topological approach for automated unstructured meshing of complex reservoir, in: 15th European Conference on the Mathematics of Oil Recovery (ECMOR XV), 2016.

[35] P. Prusinkiewicz, A. Lindenmayer, The algorithmic beauty of plants (The Virtual Laboratory), Springer, 1991.

[36] P. Prusinkiewicz, M. Hammel, E. Mjolsness, Animation of plant development, in: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, ACM, 1993, pp. 351–360.

[37] R. Měch, P. Prusinkiewicz, Visual models of plants interacting with their environment, in: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, ACM, 1996, pp. 397–410.

[38] A. Peyrat, O. Terraz, S. Merillou, E. Galin, Generating vast varieties of realistic leaves with parametric 2gmap l-systems, The visual Computer 24 (7) (2008) 807–816.

[39] O. Petrenko, R. J. G. Hernández, M. Sbert, O. Terraz, D. Ghazanfarpour, Flower modelling using natural interface and 3gmap l-systems, in: 12th ACM International Conference on Virtual Reality Continuum and Its Applications in Industry (VRCAI 2013), 2013, pp. 101–108.

[40] E. Bohl, O. Terraz, D. Ghazanfarpour, Modeling fruits and their internal structure using parametric 3Gmap L-systems, The Visual Computer 31 (6-8) (2015) 819–829.

[41] P. Wonka, M. Wimmer, F. Sillion, W. Ribarsky, Instant architecture, Vol. 22, ACM, 2003.

[42] P. Müller, P. Wonka, S. Haegler, A. Ulmer, L. Van Gool, Procedural modeling of buildings, Vol. 25, ACM, 2006.

[43] C. A. Vanegas, D. G. Aliaga, B. Benes, Building reconstruction using manhattan-world grammars, in: Computer Vision and Pattern Recognition (CVPR), 2010, pp. 358–365.

[44] Y. I. Parish, P. Müller, Procedural modeling of cities, in: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM, 2001, pp. 301–308.