Geometric modeling: consistency preservation using two-layered variable substitutions (extended version)

Thomas Bellet, Agnès Arnould, Hakim Belhaouari, Pascale Le Gall

Geometric modeling: consistency preservation using two-layered variable substitutions (extended version)

Thomas Bellet[1], Agnès Arnould[2], Hakim Belhaouari[2], and Pascale Le Gall[1]

[1] MICS, CentraleSupélec, University of Paris-Saclay, France
[2] XLIM UMR CNRS 7252, University of Poitiers, France

Abstract. In the context of topology-based geometric modeling, operations transform objects regarding both their topological structure (*i.e.* cell subdivision: vertex, edge, face, etc.) and their embeddings (*i.e.* relevant data: vertex positions, face colors, volume densities, etc.). Graph transformations with variables allow us to generically handle those operations. We use two types of variables: orbit variables to abstract topological cells and node variables to abstract embedding data. We show how these variables can be simultaneously used, and we provide syntactic conditions on rules to ensure that they preserve object consistency. This rule-based approach is the cornerstone of Jerboa, a tool that allows a fast and safe prototyping of geometric modelers.

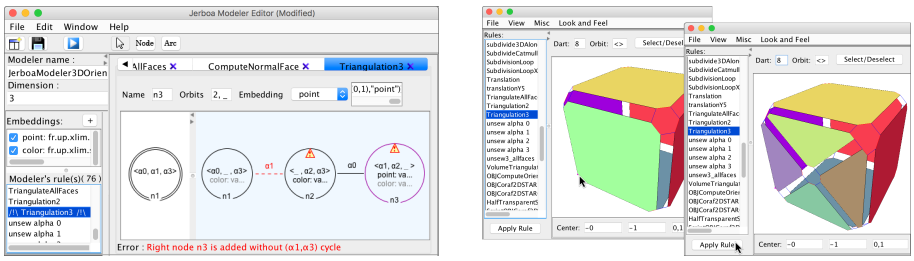
Keywords: DPO graph transformations, labeled graphs, graph variables, topology-based geometric modeling, consistency preservation.

1 Introduction

Context. Geometric modeling concerns mathematical models useful to create, manipulate, modify or display realistic n-dimensional (nD) objects in numerous application domains such as computer-aided design and manufacturing, mechanical engineering, architecture, geology, archaeology, medical image processing, scientific visualization, animated movies or video games. Many modeling tools are therefore developed at expensive costs to fulfill the various application needs. In order to facilitate the prototyping of new modelers, we developed a tool set for designing and generating safe geometric modeler kernels, called Jerboa[3] [1].

The Jerboa tool set. Objects are specified as generalized maps (G-maps) [6] which allow uniform modeling of complex nD objects (*e.g.* 2D surfaces, 3D volumes) by regular graphs. Their topological structure (*i.e.* cell subdivision) is encoded by the graph structure and the arc labels, while their embeddings (*i.e.* geometric or applicative data) are given by the node labels. Their consistency is guaranteed thanks to G-map labeling constraints. Designing a modeler therefore starts by specifying the dimension and the embeddings of manipulated objects (*e.g.* 3D

[3] http://xlim-sic.labo.univ-poitiers.fr/jerboa/



(a) Modeler editor (b) Generated modeler

Fig. 1. Design and application of the face triangulation with Jerboa

objects with vertex positions and face colors in Fig. 1). Modeling operations are then defined as graph transformation rules, using two types of dedicated variables: orbit variables [18] and node variables [3] which respectively abstract topological structures (*e.g.* in order to subdivide a face whatever its number of vertices) and embeddings (*e.g.* in order to compute the barycenter of a face whatever its vertices' positions). The rule editor (see Fig. 1(a)) includes a syntactic analyzer which guides the user while ensuring consistency. Once designed, a modeler can be generated and used right away with the provided generic viewer (see Fig. 1(b)) or integrated into larger tools. End-users of this modeler are not required to understand the rule language as they only pick and apply operations interactively.

Building consistent objects. Jerboa's rule language relies on two key aspects: the instantiation of variables and the syntactic conditions of consistency preservation (*i.e.* conditions on rules that preserve G-map labeling constraints). As long as only one variable type is concerned, these aspects have been proved well-founded. Purely topological operations defined with the sole use of orbit variables (*e.g.* unsew a vertex, sew two faces, etc.) have been discussed in [18, 4], whereas geometric operations defined with the sole use of node variables (*e.g* translate a vertex, swap two face colors, etc.) have been discussed in [2, 3]. Therefore, the previous limitation of Jerboa regarded the simultaneous use of both variable types which come with different instantiation mechanisms and syntactic conditions, making them hard to use together.

Contribution. Following the general approach of DPO graph transformations with variables defined in [10], we propose a two-layered instantiation of variables. Orbit variables are first substituted by cells of the object, thus automatically duplicating the node variables. These can then be substituted if a match morphism exists, thus leading to a classical DPO application of the instantiated rule. To ensure that transformed objects are consistent, we extend the syntactic conditions on rules separetely defined for each variable type to rules with both variable types.

Related work. Formal rule languages are already commonly used in the context of geometric modeling. In particular, L-systems are particularly useful to procedurally model regular objects such as plants or buildings [5, 14]. However, they are

inadequate to design a generic geometric modeler since every new rule requires dedicated implementation efforts. Conversely, graph transformation rules are self-contained and can be applied with a single rule application engine (for a given transformation class). Moreover, they have already been enriched with several variable types with various genericity purposes (*e.g.* label computations [12], labeling constraints [15, 16], structural transformations [11, 9]), which facilitated the definition of variables dedicated to geometric modeling. At last, let us point out that despite the existence of many efficient generic tools (*i.e.* GrGen.NET, Groove, AGG, etc.) [17, 13], we favored the development of a dedicated tool [1] for performance issues. Indeed, as any geometric modeler, modelers designed with the help of Jerboa have to interactively handle objects that can be over a million nodes large, whereas the mentioned tools only allow few thousand nodes.

Paper organization. Section 2 presents G-maps [6] and conditions under which rules without variable define consistent transformations. Section 3 presents orbit variables and node variables and their respective conditions of consistency preservation. In Section 4, we introduce a dedicated two-layered instantiation process to simultaneously use both variable types and we provide new conditions for consistency preservation.

2 G-maps and their transformations

2.1 G-maps

The topological model of G-maps [6] can be directly encoded with labeled graphs: the topological structure is defined by both the graph itself and the arc labels, while the embedding is defined by node labels. To handle multiple embeddings (*e.g.* vertex positions and face colors in Fig. 2(a)), we defined in [2] the category of *Π-graphs* (with Π a finite set of node labels) as an extension of partially labeled graphs [8], in which nodes have $|\Pi|$ labels.

In the sequel, $n \in \mathbb{N}$ will denote the dimension of considered objects and τ will denote a generic data type name with $\lfloor\tau\rfloor$ its set of typed values.

Π-graph. For $\Pi = (\pi :\rightarrow \tau)$ a family of typed names, a *Π-graph* $G = (V, E, s, t, (\pi)_{\pi\in\Pi}, \alpha)$ consists in a set E of arcs, two functions source $s : E \rightarrow V$ and target $t : E \rightarrow V$, a family of partial functions[4] $(\pi : V \rightarrow \lfloor\tau\rfloor)_{\pi\in\Pi}$ that label nodes and a partial function $\alpha : E \rightarrow [0, n]$ that labels arcs.

All examples will be colored G-maps in dimension 2, such as in Fig. 2. Arcs are labeled on $[0, 2]$ to encode topological relations, while nodes are labeled by positions and colors (functions *pos* and *col* respectively) to encode embedding data ($\lfloor\tau_{\text{pos}}\rfloor = [A, B, C, D, \dots]$, $\lfloor\tau_{\text{col}}\rfloor = [$●, ●, ●, ●, $\dots]$).

[4] Given X and Y two sets, a partial function f from X to Y is a total function $f : X' \rightarrow Y$, from X' a subset of X. X' is called the domain of f, and is denoted by $Dom(f)$. For $x \in X\backslash Dom(f)$, we say that $f(x)$ is undefined, and write $f(x) = \perp$. We also note $\perp : X \rightarrow Y$ the function totally undefined, that is $Dom(\perp) = \emptyset$.

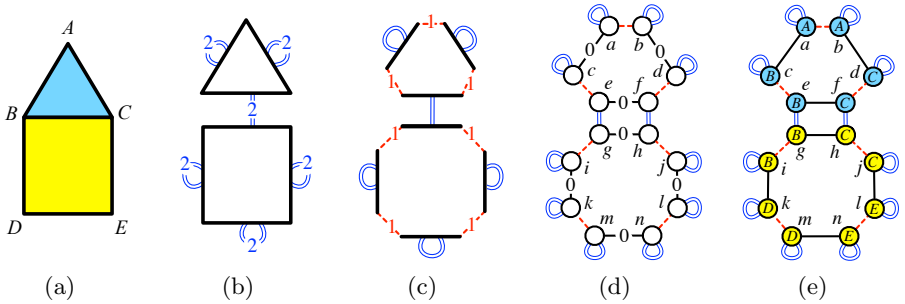


Fig. 2. Decomposition of a geometric 2D object into a 2-G-map

G-maps intuitively result from the object decomposition into topological cells. The 2D object of Fig. 2(a) is first (Fig. 2(b)) decomposed into faces connected along their common edge with a 2-relation and provided with 2-loops on border edges. Similarly, faces are split into edges connected with 1-relations (Fig. 2(c)). At last, edges are split into vertices by 0-relations to obtain the 2-G-map of Fig. 2(d). Nodes obtained at the end of the process are the G-map nodes and the different i-relations are labeled arcs: for a G-map of dimension n, i belongs to $[0, n]$. For readability purpose, we will use the graphical codes of Fig. 2 (black line for 0-arcs, red dashed line for 1-arcs and blue double line for 2-arcs).

Topological cells of G-maps are defined by subgraphs called *orbits* and built from an originating node v and a set $o \subseteq [0, n]$. By denoting $\langle o \rangle$ any ordered list of elements of o (e.g. $\langle 1\,2 \rangle$ or $\langle 2\,1 \rangle$ for $o = \{1, 2\}$), the orbit $\langle o \rangle(v)$ (of type $\langle o \rangle$ adjacent to v) is the subgraph which contains v, the nodes reachable from node v using arcs labeled on o, and the arcs themselves. By definition, embedding data (positions or colors) are shared by all nodes belonging to an orbit of the associated type. In Fig. 2(e), the vertex adjacent to e is the orbit $\langle 1\ 2 \rangle(e)$ which contains nodes c, e, g and i, all labeled with the position B attached to the vertex. Similarly, nodes a, b, c, d, e, f all belong to the same face orbit $\langle 0\ 1 \rangle$ and are all labeled by the same color ●.

Topological graph. An Π-graph G is a *n-topological graph* if the arc labeling function α is a total function on $[0, n]$. G_α, called the topological structure of G, is the graph G, except that all node labeling functions are totally undefined.

Orbit type. An *orbit type* $\langle o \rangle$ is a subset $o \subseteq [0, n]$, and denoted as a word on $[0, n]$ without repetition.

Orbit equivalence. For any orbit type $\langle o \rangle$, $\equiv_{G\langle o \rangle}$ is the *orbit equivalence relation* defined on $V_G \times V_G$ as the reflexive, symmetric and transitive closure built from arcs with labels in o (*i.e.* ensuring for each arc $e \in G$ with $\alpha(e) \in o$, $s_G(e) \equiv_{G\langle o \rangle} t_G(e)$). A graph whose edge labels are in $\langle o \rangle$ is said to be of type $\langle o \rangle$.

Orbit. For $v \in V_G$, the *$\langle o \rangle$-orbit* of G adjacent to v, denoted by $G\langle o \rangle(v)$ (or $\langle o \rangle(v)$), is the subgraph of G whose set of nodes is the equivalence class of v using $\equiv_{G\langle o \rangle}$ and whose set of arcs are those labeled on o between nodes of $G\langle o \rangle(v)$, and whose source, target, labeling functions are the restrictions of functions of G.

Embedding. An *embedding* $\pi : \langle o \rangle \to \tau$ is characterized by a name π, a data type name τ and a support orbit type $\langle o \rangle$.

We will therefore consider $pos : \langle 1\ 2 \rangle \to point$ and $col : \langle 0\ 1 \rangle \to color$. Regarding an embedding $\pi : \langle o \rangle \to \tau$, all nodes of an $\langle o \rangle$-orbit will share the same label by π (called π-label). G-maps are provided with an embedding constraint capturing this property [2]. Besides, G-maps are equipped with constraints relating to the topology. The cycle constraint ensures that in G-maps, two i-cells can only be adjacent along $(i-1)$-cells. For instance, in Fig. 2(d), the 0202-cycle constraint implies that faces are stuck along topological edges.

Definition 1 (G-map [6, 18, 2]). *For $\Pi = (\pi : \langle o \rangle \to \tau)$ a family of embeddings, a G-map embedded on Π (or Π-embedded G-map) is a topological graph $G = (V, E, s, t, (\pi)_{\pi \in \Pi}, \alpha)$ that satisfies the following consistency constraints:*

- Symmetry*: G is symmetric (i.e. $\forall e \in E$, $\exists e' \in E$, such that $s(e') = t(e)$, $t(e') = s(e)$, and $\alpha(e') = \alpha(e)$),*
- Adjacent arcs*: each node is the source node of $n+1$ arcs labeled from 0 to n,*
- Cycles*: $\forall i, j$ such that $0 \leq i \leq i+2 \leq j \leq n$, there exists a cycle[5] labeled by $ijij$ starting from each node.*
- Embedding consistency*: every node labeling function $\pi \in \Pi$ is total and for all nodes v and w such that $v \equiv_{\langle o \rangle} w$, $\pi(v) = \pi(w)$.*

2.2 Consistent G-map transformations using DPO

Morphisms on Π-graphs extend morphisms defined on partially labeled graphs to a set of labels Π. In [2], we extended relabeling graph transformations of [8] to Π-graphs using the double-pushout approach (DPO) [7].

Morphism. For two Π-graphs $G = (V_G, E_G, s_G, t_G, (\pi_G)_{\pi \in \Pi}, \alpha_G)$ and $H = (V_H, E_H, s_H, t_H, (\pi_H)_{\pi \in \Pi}, \alpha_H)$, a *$\Pi$-graph morphism* $m : G \to H$ is defined by two functions $m_V : V_G \to V_H$ and $m_E : E_G \to E_H$ preserving sources, targets and labels - *i.e.* $s_H \circ m_E = m_V \circ s_G$, $t_H \circ m_E = m_V \circ t_G$, $\pi_H(m(v)) = \pi_G(v)$ for all $\pi \in \Pi$ and $v \in Dom(\pi_G)$, and $\alpha_H(m(e)) = \alpha_G(e)$ for all $e \in Dom(\alpha_G)$. If $m(v) = v$ for all $v \in V_G$ and $m(e) = e$ for all $e \in E_G$, m is an *inclusion* and is denoted $m : G \hookrightarrow H$.

We have shown in [2] that the Π-graph category inherits from the partially labeled graph category [8] all classical properties such as the existence of pushouts.

Rule. A rule $r : L \hookleftarrow K \hookrightarrow R$ consists of two inclusions $K \hookrightarrow L$ and $K \hookrightarrow R$ such that:

– for all $v \in V_L$ (resp. for all $e \in E_L$ and all $\pi \in \Pi$), $\alpha_L(v) = \perp$ (resp. $\pi_L(e) = \perp$) implies $v \in V_K$ and $\alpha_R(v) = \perp$ (resp. $e \in E_K$ and $\pi_R(e) = \perp$,),

– for all $v \in V_R$ (resp. for all $e \in E_R$ and all $\pi \in \Pi$), $\alpha_R(v) = \perp$ (resp. $\pi_R(e) = \perp$) implies $v \in V_K$ and $\alpha_L(v) = \perp$ (resp. $e \in E_K$ and $\pi_L(e) = \perp$).

We call L the *left-hand side*, R the *right-hand side* and K the *interface* of r.

[5] A cycle is a sequence $e_1 ... e_k$ of arcs such $t(e_i) = s(e_{i+1})$ for each $1 \leq i < k$ and $t(e_k) = s(e_1)$. The word $\alpha(e_1)...\alpha(e_k)$ is called its label.

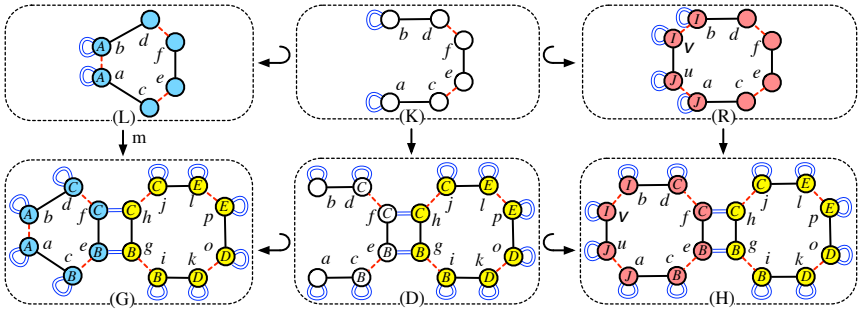


Fig. 3. A direct derivation

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 [8] as in Fig. 3, where $m : L \to G$, called a *match morphism*, is injective[6]. If this derivation exists, we write $G \Rightarrow_{r,m} H$.

For example, the rule $L \hookleftarrow K \hookrightarrow R$ in Fig. 3 matches a blue triangle which has a vertex at position A. The rule transforms the triangle into a square by splitting this vertex into two vertices at positions I and J, while changing the face color to red. According to the rule definition, unmatched labels in L such as the position of node d are also unmatched in R. Secondly, nodes of R with undefined labels are preserved nodes of the rule, with undefined labels in L. Consequently, added nodes of R such as the node v have all their labels defined in R (I and ●). Moreover, changed labels such as the color of d in R (●) are matched in L (●). These rule properties ensure that when the rule is applied to a totally labeled object such as G in Fig. 3, the resulting graph H is also totally labeled.

To ensure that direct derivations preserve G-map constraints of Definition 1, [18] and [2] respectively introduced *topological conditions* and *embedding conditions* on rules that ensure consistency preservation. For example, the rule of Fig. 3 preserves consistency as the added nodes u and v are added with all their adjacent arcs, cycles and embeddings. Similarly, the embedding modifications (● to ● and A to I/J) are consistently defined on all concerned nodes.

Result 1 (G-map consistency preservation using basic rules [18, 2, 3]) *For $r : L \hookleftarrow K \hookrightarrow R$ a graph transformation rule and $m : L \to G$ a match morphism on a Π-embedded n-G-map, the direct transformation $G \Rightarrow^{r,m} H$ produces a Π-embedded n-G-map if r satisfies the following* topological conditions*:*

- Symmetry*: L, K and R satisfy the symmetry constraint.*
- Adjacent arcs*:*
  - *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.*

[6] A morphism g is injective if g_V and g_E are injective.

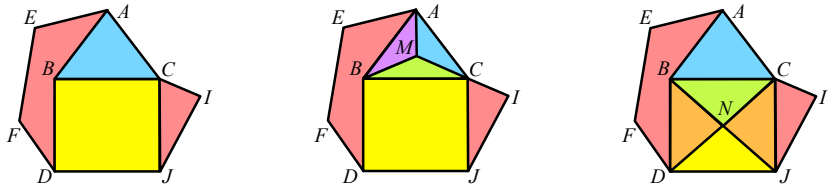
– Cycles*: for all pair* (i, j) *such* $0 \leq i \leq i+2 \leq j \leq n$,
  - *any added node of* $R \backslash K$ *is source of an* $ijij$*-cycle;*
  - *any preserved node of* K *which is source of an* $ijij$*-cycle in* L*, is also source of an* $ijij$*-cycle in* R*;*
  - *any preserved node of* K *which is not source of an* $ijij$*-cycle in* L *is source of the same* i*-arcs and* j*-arcs in* L *and* R*.*

and the following embedding conditions *for all* $\pi : \langle o \rangle \to \tau$ *in* Π*:*

– Embedding consistency*: L, K, R satisfy the embedding consistency constraint.*
– 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 exists a node* w *in* $R\langle o \rangle(v)$ *such that* w *is not in* $L\langle o \rangle(v)$*, then there exists* v' *in* K *with* $v' \equiv_{L\langle o \rangle} v$ *and* $v' \equiv_{R\langle o \rangle} v$ *such that* $\pi_L(v') \neq \perp$*.*

3 Rule variables for geometric modeling

Let us consider the face triangulation operation of Fig. 4. On the topological side, the face is subdivided into as many triangles as it contains edges. On the embedding side, new face colors are computed as the mix of the subdivided face color and the neighboring face color, while the position of the created vertex is set as the barycenter of the transformed face. As the operation depends on the attributes (number of vertices, position, color) of the matched object, its definition by a single generic rule requires the use of variables.

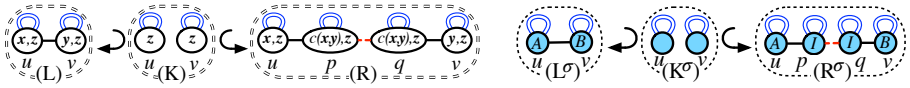


(a) Modeled object (b) Triangle triangulation (c) Square triangulation

Fig. 4. A modeling operation: face triangulation

3.1 Graph transformations with variables

Intuitively, *rule schemes* (rules with variables) describe as many concrete rules as there are possibilities to instantiate variables with concrete elements. In [10], a generic approach has been proposed to deal with variables within graph transformations: roughly speaking, a rule scheme is first applied to a graph along a *kernel match morphism* (in our case a morphism that only matches the topological structure L_α of the left-hand side L). Then, if a substitution σ associating a value

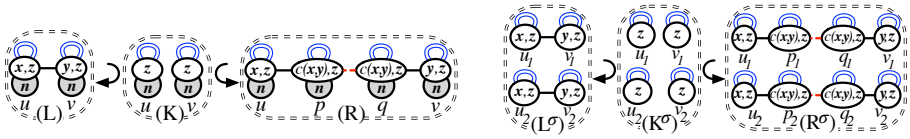


(a) Rule scheme (b) A rule instance

Fig. 5. Embedding label computations with attribute variables

for all variables can be induced from the kernel match morphism, an instance rule, generically denoted $L^\sigma \hookleftarrow K^\sigma \hookrightarrow R^\sigma$ for a rule scheme $L \hookleftarrow K \hookrightarrow R$, can be built and finally, applied to the graph as a classical rule.

Using this principle, [10] introduces three types of variables. Let us briefly present two of them which relate to our dedicated variable types. First, *attribute variables* illustrated in Fig. 5 allow label computation. In the rule scheme of edge subdivision, variables x and y abstract two position labels while z abstracts a color one. In the right-hand side, a topological vertex is added at the center of the existing two positions, using the dedicated center operator $c : point \times point \rightarrow point$, while the color is set to the same face color z. Secondly, *clone variables* illustrated in Fig. 6 allow structural abstraction. Intuitively, in order to subdivide n edges, nodes of the rule scheme which are labeled by the clone variable n are duplicated by as many nodes as the multiplicity value used to instantiate n, while arcs are duplicated accordingly to node duplications and arcs of the rule scheme. Note that as we also use the attribute variables x, y, and z, the rule scheme of Fig. 6(b) still requires to substitute them.

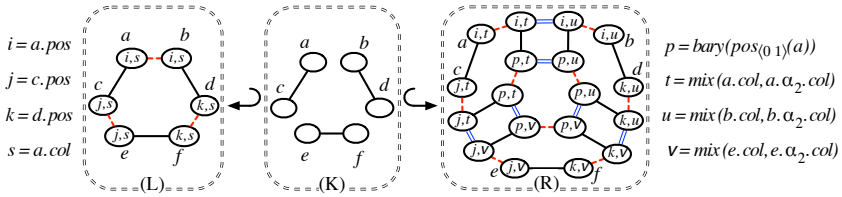


(a) Rule scheme with n as clone variable (b) Instantiation of n by 2

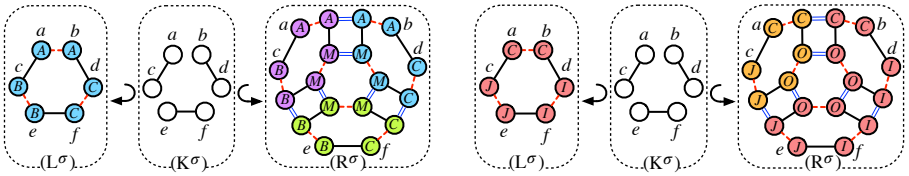
Fig. 6. Expanding computations with clone variables

3.2 Node variables for embedding computation

Embedding computations require to traverse the topological structure of modified objects: *e.g.* the face triangulation of Fig. 4 requires to access the colors of adjacent faces, whether such faces exist or not. In [3], we therefore introduced *node variables* which are similar to attribute variables. By directly using node names of L as variables, we provide operators on node variables to access embedding labels and adjacent nodes, thanks to G-map regularity. For a node variable a, $a.\pi$ gives access to its π-label while $a.\alpha_i$ (with $i \leq n$) gives access to the node connected to node a by an i-arc. Embedding expressions used in a rule scheme $L \hookleftarrow K \hookrightarrow R$ are then terms built over these operators and nodes of L. Thus, for a kernel match morphism $m : L_\alpha \rightarrow G$, a rule instance $L^{m_V} \hookleftarrow K^{m_V} \hookrightarrow R^{m_V}$ can be computed by using the node matching function m_V as variable substitution.



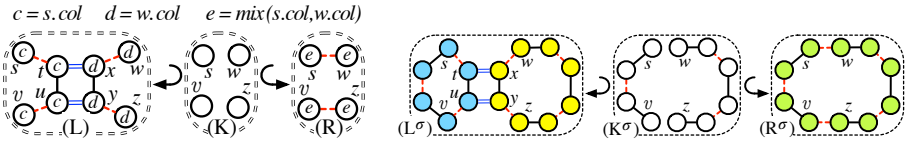
(a) Rule scheme



(b) A rule instance (c) Another rule instance

Fig. 7. Triangle triangulation with embedding terms

From the rule scheme of Fig. 7(a), we can build the two rule instances of Fig. 7(b) and 7(c) respectively corresponding to triangles BAC and JCI of Fig. 4(a). In the left-hand side, the term $a.col$ is respectively evaluated as ● and ●. In the right-hand side, the dedicated operator $mix : color \times color \rightarrow color$ is applied to the face color ($a.col$) and the color of the neighboring face ($a.\alpha_2.col$). At rule application, $a.\alpha_2$ is evaluated as the 2-neighbor of a. Note that in the border, nodes are their self 2-neighbors (*e.g.* node a in Fig. 2(e)). Consequently, some created faces in Fig. 7(b) and 7(c) keep their original color. At last, embedding expressions also include operators in charge of collecting all embedding values carried by a given orbit. In the rule scheme of Fig. 7(a), the term $pos_{\langle 0\ 1\rangle}(a)$ is evaluated as the multiset of positions labeling the face, i.e. of $\langle 0\ 1\rangle$-orbit adjacent to node a. In the instance of Fig. 7(b), this set is $[A, B, C]$ and the added vertex is positioned at $bary([A, B, C]) = M$ using a dedicated barycenter operator.



(a) Rule scheme (b) A rule instance

Fig. 8. Face merge with embedding terms

In order to instantiate a rule scheme built over node variables into a rule that satisfies the conditions of Result 1, we introduced in [3] a completion step. Let us consider the example of Fig. 8 that merges two faces by removing their common edge and mixing their colors. This operation can be defined independently of face shapes by the minimal rule scheme of Fig. 8(a) that only deals with the central edge and the adjacent nodes. The completion step automatically includes in the rule instance all nodes concerned by embedding modifications. In the example of Fig. 8(b) corresponding to an application to the object of Fig. 2(e), this step

completes the rule with the rest of the two faces so that the green color can be attached on the whole new faces.

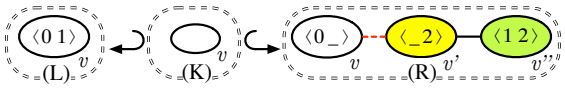
Consequently, rule schemes are exempt from the condition of full match of transformed embeddings of Result 1. However, to prevent misapplication cases in which two different embedding orbits would be matched as a single one, the condition of full match is replaced by a non-overlap condition on match morphism. Note that similarly to the condition of injective match morphism, the non-overlap condition has to be dynamically checked, with no particular difficulty.

Result 2 (G-map consistency preservation using node variables [3])
For $r: L \hookleftarrow K \hookrightarrow R$ a rule scheme with node variables and $m: L_\alpha \to G$ a match morphism on a Π-embedded G-map, if r satisfies the conditions of Result 1, except the full match of transformed embeddings, and satisfies the following non-overlap condition *for all $\langle o \rangle$ occurring as support orbit type in $\Pi = (\pi : \langle o \rangle \to \tau)$, then the instance rule[7] $r^{m_V} : L^{m_V} \hookleftarrow K^{m_V} \hookrightarrow R^{m_V}$ satisfies the conditions of Result 1.*

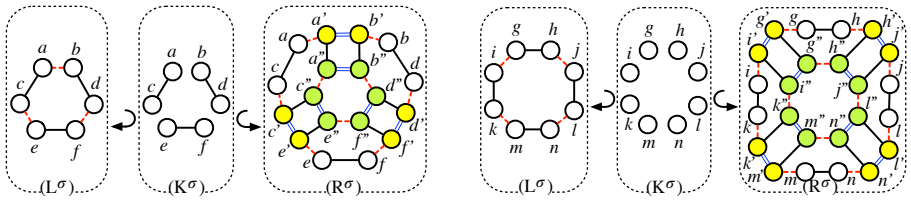
Non-overlap*: for $v, u \in V_L$ such as $v \not\equiv_{L\langle o \rangle} u$, $m(v) \not\equiv_{G\langle o \rangle} m(u)$.*

3.3 Orbit variables for topological rewriting

As existing variable types were unfit to abstract G-map cell transformations, we introduced *orbit variables* in [18]. Intuitively, these are typed by an orbit type $\langle o \rangle$ so that they can abstract any G-map orbit of type $\langle o \rangle$. By rewriting $\langle o \rangle$ into another type $\langle o' \rangle$, we can change arc labels or remove arcs. For example, the rule scheme of Fig. 9(a) models the topological triangulation of any face - *i.e* any orbit of type $\langle 0\ 1 \rangle$. Note that, as for clone variables in Fig. 6, the other node labels, in this case the color labels, are duplicated along the orbit variable instantiation. Note also that these colors have been chosen to help reading of orbit copies, disregarding G-map embedding consistency preservation.



(a) Rule scheme on the orbit type $\langle 0\ 1 \rangle$



(b) Triangle rule instance (c) Square rule instance

Fig. 9. Face triangulation with orbit variable

[7] Note that the instantiation includes the completion step that consist in extending the matched and transformed patterns to the embedding orbits [3].

The two instance rules of Fig. 9(b) and 9(c) are constructed as follows:

1. the node v labeled in L with the orbit type $\langle 0\ 1\rangle$ is substituted by a face, i.e. a $\langle 0\ 1\rangle$-orbit, e.g. a triangle (resp. a square), to build L^σ ;
2. each node of L^σ is kept in R^σ, and duplicated twice, corresponding to the nodes v, v' and v'', labeled by different types in R;
3. for the face matched by node v, 0-arcs are conserved in R^σ while 1-arcs are removed as v is relabeled in R by $\langle 0\ \_\rangle$ (an "empty" label $\_$ replaces 1); similarly, for node v', the type $\langle \_\ 2\rangle$ means both removal of 0-arcs and 2-relabeling of 1-arcs while for the v'' node, $\langle 1\ 2\rangle$ entails 1-relabeling of 0-arcs and 2-relabeling of 1-arcs;
4. at last, as indicated by the arcs of R between v, v' and v'' nodes, any node of the matched face v is connected to its image in copy v' with a 1-arc, and all v' and v'' images of a given node are connected with a 0-arc.

Topological rewriting. A *topological rewriting* $\langle\omega\rangle$ of an orbit type $\langle o\rangle$ is defined by a word ω on $[0,n]\cup\{\_\}$, of same length as o, and such that for all i in $[0,n]$, there is at most one occurrence of i in ω. We write $\langle o\rightarrow\omega\rangle$ for the *type rewriting function* that associates each label $o_i \in [0,n]$ at position i in $\langle o\rangle$ to its images ω_i at the same position in $\langle\omega\rangle$ - *i.e.* $\langle o\rightarrow\omega\rangle(o_i) = \omega_i$. For a graph $G = (V, E, s, t, (\pi)_{\pi\in\Pi}, \alpha)$ of type $\langle o\rangle$, we denote $G_{\langle o\rightarrow\omega\rangle} = (V, E', s, t, (\pi)_{\pi\in\Pi}, \alpha')$ the rewritten graph with $E'\subset E$ such that $\forall e \in E$, $e \notin E'$ if $\langle o\rightarrow\omega\rangle(\alpha(e)) = \_$, and $e \in E'$ otherwise with $\alpha'(e) = \langle o\rightarrow\omega\rangle(\alpha(e))$. At last, for convenience, $\langle o\rangle$ denotes both an orbit type and the identity type rewriting function $\langle o\rightarrow o\rangle$.

Rule scheme with orbit variable. For an orbit type $\langle o\rangle$, a *rule scheme* $L\hookleftarrow K\hookrightarrow R$ *with orbit variable* $\langle o\rangle$ is such that all nodes of L, K and R are labeled by topological rewritings of $\langle o\rangle$ and at least one node of L is labeled by $\langle o\rangle$.

Orbit variable instantiation. For $r : L \hookleftarrow K \hookrightarrow R$ a rule scheme with orbit variable $\langle o\rangle$ and O a graph of type $\langle o\rangle$, we denote $r^O : L^O \hookleftarrow K^O \hookrightarrow R^O$ the instantiated rule[8] [18]. The functions that respectively associate instance nodes with their originating nodes in graphs of r or in O are respectively denoted $\uparrow^L_{L^O}: V_{L^O}\rightarrow V_L$, $\uparrow^L_{K^O}: V_{K^O}\rightarrow V_K$, $\uparrow^R_{R^O}: V_{R^O}\rightarrow V_R$ and $\uparrow^O_{r^O}: (V_{L^O}\cup V_{K^O}\cup V_{R^O})\rightarrow V_O$. In particular, for all $\pi \in \Pi$ and all node v of r^O, we have $\pi(v) = \pi(\uparrow^O_{r^O}(v))$.

Since nodes of rule schemes contain topological rewritings as special labels used to match topological graphs, they do not belong to the same Π-category than the underlying Π-embedded G-maps. However, as in Fig. 9, these extra labels disappear after variable instantiation. Additionally, the syntactic conditions that preserve G-map topological consistency have been adapted to handle both the explicit arcs of rule graphs and the implicit arcs of topological rewritings that label nodes. For example, node v' in Fig. 9(a) is added with all its adjacent arcs as both the 0-arc and 1-arc are explicit in R, while the 2-arc is implicit in the orbit rewriting $\langle \_\ 2\rangle$. Similarly, v' is added with an half-implicit 0202-cycle as v' and v'' are connected with an explicit 0-arc while 2 is at the same position in their topological rewritings $\langle \_\ 2\rangle$ and $\langle 1\ 2\rangle$.

[8] L^0, K^0, and R^0 are the Cartesian product graphs of O and resp. graphs L, K and R, by keeping tracks of arc relabelings and arc removals.

Result 3 (Topological consistency preservation using orbit variable [18]) *For $r: L \hookleftarrow K \hookrightarrow R$ a rule scheme with orbit variable $\langle o \rangle$ and a graph O of type $\langle o \rangle$, the instance rule $r^O: L^O \hookleftarrow K^O \hookrightarrow R^O$ satisfies the topological conditions of G-map consistency preservation of Result 1 if r satisfies the same conditions extended to implicit arcs and cycles such as for v a node of L, K or R labeled by $\langle \omega \rangle$:*

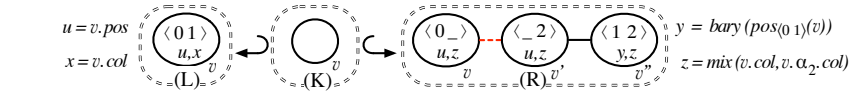
- *for $i \in [0, n]$, v is source of an* implicit i-arc *if $i \in \langle \omega \rangle$; for $v \in K$, this implicit i-arc is preserved if i is at the same position in $\langle \omega_L \rangle$ and $\langle \omega_R \rangle$ the respective labels of v in L and R* - i.e. $\langle \omega_L \rangle \rightarrow \langle \omega_R \rangle(i) = i$;
- *for all (i, j), v is source of an* implicit $ijij$-cycle *if $i \in \langle \omega \rangle$, $j \in \langle \omega \rangle$, and there exists a node v' in L labeled by $\langle \omega' \rangle$ and source of an $i'j'i'j'$-cycle such that $i' = \langle \omega \rangle \rightarrow \langle \omega' \rangle(i)$ and $j' = \langle \omega \rangle \rightarrow \langle \omega' \rangle(j)$;*
- *for all (i, j), v is source of an* half-implicit $ijij$-cycle *if $i \in \langle \omega \rangle$ (resp. $j \in \langle \omega \rangle$) and either v is source of an j-loop (resp. i-loop) or v is connected by an j-arc (resp. i-arc) to a node v' labeled by $\langle \omega' \rangle$ such that i (resp. j) is at the same position in $\langle \omega \rangle$ and $\langle \omega' \rangle$* - i.e. $\langle \omega \rangle \rightarrow \langle \omega' \rangle(i) = i$ *(resp. $\langle \omega \rangle \rightarrow \langle \omega' \rangle(j) = j$).*

As orbit variables abstract multiple nodes (and arcs), their combined use with node variable requires some care in order to instantiate both variable types.

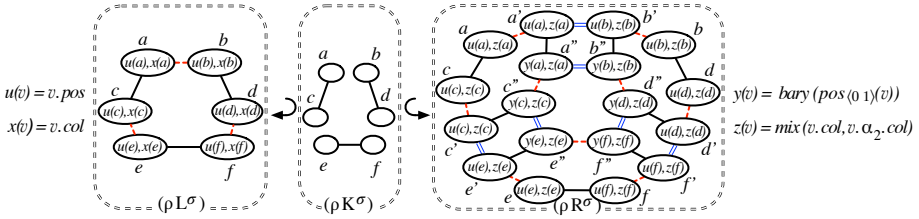
4 Rule schemes for specifying modeling operations

4.1 Combining orbit variables and node variables

Let us consider the rule scheme of Fig. 10(a) that defines the face triangulation of Fig. 4 by combining the two variable transformations of Fig. 7 and 9. Intuitively, as the orbit variable $\langle 0\ 1 \rangle$ defines the topological structure of the rule instance, it has to be substituted first in order to provide all the node variables required to match the embedding of the face. When the orbit variable is instantiated in Fig. 10(b) by a triangle face, the terms $v.pos$ and $v.col$ are duplicated on all instantiated nodes. However, they are rewritten by respectively replacing v by the new node names a, b, ..., f. For example, the term $mix(v.col, v.\alpha_2.col)$ has been rewritten on c'' by $mix(c.col, c.\alpha_2.col)$ as c is the corresponding new variable.



(a) Rule scheme on the orbit type $\langle 0\ 1 \rangle$



(b) After orbit variable substitution and term rewriting

Fig. 10. Face triangulation with both variables types

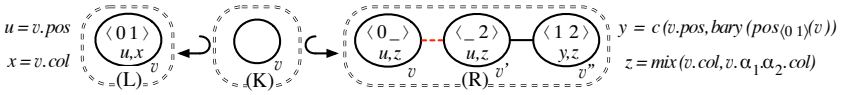
Definition 2 (Term rewriting). *Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule scheme with orbit variable $\langle o \rangle$ and node variables. Let r^O be a rule scheme with node variables resulting from a substitution $\langle o \rangle$ by a graph O of type $\langle o \rangle$.*

The rewritten rule scheme $r(O)$ results from the respective application of the following term functions ρ_v for each node v of r^O to the node labels of r^O, such that ρ_v extends the following variable substitution: for every node variable u of V_L, $\rho_v(u) = u'$ in which u' is the unique node variable of V_{LO} such that $\uparrow^L_{LO}(u') = u$ and $\uparrow^O_{rO}(u') = \uparrow^O_{rO}(v)$.

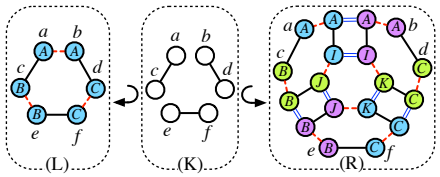
Similarly to the rule scheme of Fig. 7(a), the rewritten one of Fig. 10(a) only requires to substitute node variables to produce the instance rules of Fig. 7(b) and 7(c), but in this case multiple terms define the same value. For example, the barycenter is successively defined as $bary(pos_{\langle 0\ 1\rangle}(a))$, $bary(pos_{\langle 0\ 1\rangle}(b))$, ..., $bary(pos_{\langle 0\ 1\rangle}(f))$ which will all result in the same position. Therefore we must adapt the conditions of consistency preservation.

4.2 Consistency preservation

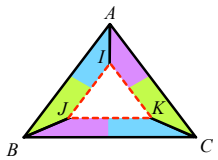
Let us consider the rule scheme of Fig. 11(a) that still defines the triangulation, but with different embedding computations. In particular, the center is positioned right between the corner and the barycenter ($c(v.pos, bary_{\langle 0\ 1\rangle}(v))$) and the color of created faces is defined as the mix between the original color and the color of the adjacent face around the corner ($v.\alpha_1.\alpha_2.col$). Note that this rule satisfies the conditions of embedding consistency preservation as defined in Result 2. However, most instances of this scheme surely break the embedding consistency. Fig. 11(b) and 11(c) respectively present the rule instance and the intuitive representation for its application to the triangle ABC of Fig. 4(a). This rule breaks the added vertex consistency since three different positions are computed: $I = c(A, M)$, $J = c(B, M)$, $K = c(C, M)$ with $M = bary([A, B, C])$. Similarly, each created face is embedded with two different colors since the original color is mixed with the colors of the faces around the two corners.



(a) Rule scheme on the orbit type $\langle 0\ 1 \rangle$



(b) A rule instance



(c) Intuitive representation

Fig. 11. Break of G-map consistency

To prevent inconsistencies, conditions on rule scheme must take into account the pattern expanding and the term rewriting due to the orbit variable substitution. For this purpose, considering rule schemes without orbit variable, we first define the equivalence between two embedding terms that ensures embedding equality on rule application to a G-map.

Definition 3 (Term equivalence). *Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule scheme with node variables.*

For any terms t and t' of r that define an embedding $\pi : \langle o \rangle \to \tau$, the equivalence *between terms, denoted $t \equiv_{L\langle o \rangle} t'$, is the smallest equivalence relation extending the equivalence between nodes of L such that:*

- *for all dimension $i \in [0, n]$, let $\langle o' \rangle$ the sub-orbit of $\langle o \rangle$ containing all dimensions j of $\langle o \rangle$ such $j + 2 \leq i$ or $i + 2 \leq j$, if $t \equiv_{L\langle o' \rangle} t'$ then $t.\alpha_i \equiv_{L\langle o \rangle} t'.\alpha_i$,*
- *for all dimension $i \in \langle o \rangle$, if $t \equiv_{L\langle o \rangle} t'$ then $t \equiv_{L\langle o \rangle} t'.\alpha_i$ and $t.\alpha_i \equiv_{L\langle o \rangle} t'$,*
- *for all embedding $\pi' : \langle o' \rangle \to \tau'$, if $t \equiv_{L\langle o' \rangle} t'$ then $t.\pi' \equiv_{L\langle o \rangle} t'.\pi'$,*
- *for all orbit $\langle o' \rangle$, if $t \equiv_{L\langle o' \rangle} t'$ then $\pi_{\langle o' \rangle}(t) \equiv_{L\langle o \rangle} \pi_{\langle o' \rangle}(t')$,*
- *for all user function $f : s_1 \times ... \times s_m \to s_{m+1}$ and all terms $t_1, t'_1 : s_1, ..., t_m, t'_m : s_m$, if $t_1 \equiv_{L\langle o \rangle} t'_1$ and ... and $t_m \equiv_{L\langle o \rangle} t'_m$ then $f(t_1, ..., t_m) \equiv_{L\langle o \rangle} f(t'_1, ..., t'_m)$.*

Intuitively, in the rule scheme of Fig. 10(b), the two terms $bary(pos_{\langle 0\ 1 \rangle}(a))$ $bary(pos_{\langle 0\ 1 \rangle}(b))$ are equivalent because a and b belong to the same $\langle 0\ 1 \rangle$-orbit in L. Similarly, $mix(a.col, a.\alpha_2.col)$ and $mix(c.col, c.\alpha_2.col)$ are equivalent because: (i) $a.col$ and $c.col$ are equivalent as they belong to the same $\langle 0\ 1 \rangle$-orbit carrying the color embedding; (ii) $a.\alpha_2.col$ and $c.\alpha_2.col$ are equivalent as the α_2-access ensures that $a.\alpha_2$ and $c.\alpha_2$ belong to the same $\langle 0\ 1 \rangle$-orbit and thus have the same color.

Theorem 1 (Evaluation equality along term equivalence). *Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule scheme with node variables and $m : L_\alpha \to G$ a kernel match morphism on a Π-embedded n-G-map G.*

If two terms t and t' of r defining an embedding $\pi : \langle o \rangle \to \tau$ are equivalent, i.e. $t \equiv_{L\langle o \rangle} t'$, then their interpretations along m_V are equal, i.e. $t^{m_V} = t'^{m_V}$.

Proof. Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule scheme with embedding terms (without orbit variable) and $m : L_\alpha \to G$ a kernel match morphism on a Π-embedded n-G-map G. Let us show by induction on the structure of t and t', that if two terms t and t' of r defining an embedding $\pi : \langle o \rangle \to \tau$ are equivalent, *i.e.* $t \equiv_{L\langle o \rangle} t'$, then their interpretations along m_V are equal, *i.e.* $t^{m_V} = t'^{m_V}$.

If $t \equiv_{L\langle o \rangle} t'$, then by term equivalence definition 3, one of the following case is verifyed:

- $t = u$ and $t' = u'$ are two node variables of L. Then, by match morphism definition, $q^{m_V} \equiv_{G\langle o \rangle} q'^{m_V}$.
- $t = p.\alpha_i$, $t' = p'.\alpha_i$ with $i \in [0, n]$, p and p' two node termes, and $p \equiv_{L\langle o' \rangle} p'$ with $\langle o' \rangle$ the sub-orbit of $\langle o \rangle$ containing all dimensions j of $\langle o \rangle$ such that $j + 2 \leq i$ or $i + 2 \leq j$. Then by induction hypothesis, $p^{m_V} \equiv_{G\langle o' \rangle} p'^{m_V}$. And thanks to the cycle consistency constraint on the G-map G (see Definition 1), t^{m_V} and t^{m_V} are two nodes of the same $\langle o' \rangle$-orbit and then of the same $\langle o \rangle$-orbit , *i.e.* $t^{m_V} \equiv_{G\langle o \rangle} t'^{m_V}$.

- $t = p.\alpha_i$ (resp. $t' = p'.\alpha_i$) with $i \in \langle o \rangle$ and p (resp. p') is a node term, and $p \equiv_{L\langle o \rangle} t'$ (resp. $t \equiv_{L\langle o \rangle} p'$). Then by induction hypothesis ${p}^{m_V} \equiv_{G\langle o \rangle} {t'}^{m_V}$ (resp. ${t}^{m_V} \equiv_{G\langle o \rangle} {p'}^{m_V}$). And thanks to the embedding consistency constraint on the G-map G (see Definition 1), ${t}^{m_V} \equiv_{G\langle o \rangle} {t'}^{m_V}$.
- $t = p.\pi'$ and $t' = p'.\pi'$ with $\pi' : \langle o' \rangle \to \tau'$ an embedding of Π, p and p' two node termes, and $p \equiv_{L\langle o' \rangle} p'$. Then, by induction hypothesis ${p}^{m_V} \equiv_{G\langle o' \rangle} {p'}^{m_V}$. And by definition of embedded G-maps, ${p}^{m_V}$ and ${p'}^{m_V}$ have the same π'-embedding ($\pi'({p}^{m_V}) = \pi'({p'}^{m_V})$), and therefore ${t}^{m_V} = {t'}^{m_V}$.
- $t = \pi'_{\langle o' \rangle}(p)$ and $t' = \pi'_{\langle o' \rangle}(p')$ with $\pi' : \langle o' \rangle \to \tau'$ an embedding of Π, p and p' two node termes, and $p \equiv_{L\langle o' \rangle} p'$. Then by induction hypothesis ${p}^{m_V}$ and ${p'}^{m_V}$ are two nodes of the same collected $\langle o' \rangle$-orbit, *i.e.* ${p}^{m_V} \equiv_{G\langle o' \rangle} {p'}^{m_V}$. Consequently, ${t}^{m_V}$ and ${t'}^{m_V}$ are the same set of τ' values, *i.e.* ${t}^{m_V} = {t'}^{m_V}$.
- $t = f(p_1, \ldots, p_m)$ and $t' = f(p'_1, \ldots, p'_m)$ with a user function $f : s_1 \times ... \times s_m \to s_{m+1}$, and $p_1, \ldots, p_m, p'_1, \ldots, p'_m$ terms of type s_1, ..., s_m respectively, and $p_1 \equiv_{L\langle o' \rangle} p'_1$, ..., $p_m \equiv_{L\langle o' \rangle} p'_m$. By induction hypothesis, because s_1, ..., s_m are user types distinct from nodes, ${t_1}^{m_V} = {t'_1}^{m_V}$, ..., ${t_m}^{m_V} = {t'_m}^{m_V}$. Then, $f({t_1}^{m_V}, \ldots, {t_m}^{m_V}) = f({t'_1}^{m_V}, \ldots, {t'_m}^{m_V})$, and therefore ${t}^{m_V} = {t'}^{m_V}$.

Consequently, equivalent embedding terms have equal interpretations. □

We now define a condition on rule schemes with orbit variable that ensure term equivalence, and therefore stability along the instantiation process. Intuitively, this condition ensures that when a term labels a node that is also labeled by a topological rewriting, the rewritten terms should be equivalent for all expanded embedding orbits. This can be predicted by comparing the term itself with terms capturing the relabeling of concern arcs - *i.e.* that belong to the embedding orbit.

Let us consider the term $bary(pos_{\langle 0\ 1 \rangle}(v))$ that labels node v'' in the rule scheme of Fig. 10(a). As node v'' is labeled by $\langle 1\ 2 \rangle$ and will be expanded, we must ensure equivalence for all labels of the $\langle 1\ 2 \rangle$-subset of $\langle 1\ 2 \rangle$ (vertex orbit carrying the position embedding), therefore in this case for both 1 and 2. As node v appearing in the term is originally labeled $\langle 0\ 1 \rangle$ in L, we have to consider the respective reverse relabeling $\langle 1\ 2 \to 0\ 1 \rangle(1) = 0$ and $\langle 1\ 2 \to 0\ 1 \rangle(2) = 1$. Consequently, the term must be equivalent to both $bary(pos_{\langle 0\ 1 \rangle}(v.\alpha_0))$ and $bary(pos_{\langle 0\ 1 \rangle}(v.\alpha_1))$. This is indeed true as $v.\alpha_0$ and $v.\alpha_1$ both belong to the collected orbit $\langle 0\ 1 \rangle(v)$.

Similarly, let us consider the term $mix(v.col, v.\alpha_2.col)$ labeling v''. We must ensure equivalence for all labels of the $\langle 1\ 2 \rangle$-subset of the $\langle 0\ 1 \rangle$ (face orbit carrying the color embedding), therefore in the case only 1. As node v appearing in the term is originally labeled by $\langle 0\ 1 \rangle$ in L, we consider $\langle 1\ 2 \to 0\ 1 \rangle(1) = 0$ and show that the term is equivalent to $mix(v.\alpha_0.col, v.\alpha_0.\alpha_2.col)$. As α_0 belong to the face orbit $\langle 0\ 1 \rangle$ carrying the color embedding, $v.col$ and $v.\alpha_0.col$ are equivalent. Similarly, because of the 0202-cycle constraint of G-maps, $v.\alpha_2$ and $v.\alpha_0.\alpha_2$ belong to the same face, therefore $v.\alpha_2.col$ and $v.\alpha_0.\alpha_2.col$ are equivalent terms.

Definition 4 (Condition of term stability). *Let $r : L \hookleftarrow K \hookrightarrow R$ be a rule scheme with orbit variable and node variables, and v a node of L, K or R, such v is labeled by an embedding term t defining an embedding $\pi : \langle o \rangle \to \tau$ and by a topological rewriting $\langle \omega_v \rangle$.*

*The term t is stable along instantiation if for all label i of $\langle o' \rangle$ the $\langle o \rangle$-subset of $\langle \omega_v \rangle$, t is equivalent to the rewritten term t_i (*i.e. $t \equiv_{L\langle o \rangle} t_i$*) in which any occurence of a variable $x \in V_L$ is replaced by $x.\alpha_j$, in which $j = \langle \omega_v \to \omega_x \rangle(i)$* [9] *with $\langle \omega_x \rangle$ the topological rewriting that labels x in L.*

Finally, rule schemes containing both node variables and orbit variables preserve G-map consistency if they satisfy all conditions previously introduced, including term stability. Note that the non-overlap condition of Result 2 is still to be additionally checked on the application morphism.

Theorem 2 (G-map consistency preservation using both variable types). *For a rule scheme $r : L \hookleftarrow K \hookrightarrow R$ with orbit variable and node variables, any rule instance resulting from the orbit variable substitution satisfies the conditions of Result 2 if r satisfies the topological conditions of Result 3 and the following embedding conditions:*

- Embedding consistency *of Result 1;*
- Labeling of extended embedding conditions orbits *of Result 1;*
- Term stability *of Definition 4.*

Proof. For $r : L \hookleftarrow K \hookrightarrow R$ a rule scheme with orbit variable and node variables, we have to show that any rule instance resulting from the orbit variable substitution satisfies the conditions of Result 2 (G-map consistency preservation using node variables) if r satisfies the topological conditions of Result 3 (G-map consistency preservation using orbit variable) and the following embedding conditions:

- Embedding consistency and labeling of extended embedding conditions orbits of Result 1 (G-map consistency preservation using basic rules);
- Term stability of Definition 4.

Result 3 showed that if a rule scheme with orbit variable satisfies the topological conditions of Result 1 up to arcs and cycles within orbit variable, instances resulting from the orbit variable substitution fully satisfy the same conditions. As nodes variable substitution does not affect the topological structures of rules, instances resulting from this second step also satisfy these conditions. We still have to show that rule instances satisfy the embedding conditions of Result 1.

[3] already showed that if a rule scheme with orbit variable satisfy the non-overlap condition of Result 2, instance resulting from node variable substitution satisfy the full match of transformed embeddings of Result 1.

Let us show that the embedding consistency condition and of the term stability condition ensure that rule instances satisfy the embedding consistency condition. Let us note $r(O) : L(O) \hookleftarrow K(O) \hookrightarrow R(O)$ the rule scheme resulting from orbit

[9] Note that if $\langle \omega_v \to \omega_x \rangle(i) = \_$, t_i does not exist and therefore t is not stable.

substitution and term rewriting. For any embedding operation $\pi : \langle o \rangle \to \tau$, let us consider v and v' two nodes of $R(O)$ (resp. $L(O)$, $K(O)$) connected by an α_i-arc e with $i \in \langle o \rangle$, respectively labeled by the rewritten term t and t'. Two cases exist:

- v and v' have a same antecedent u in R (resp. L, K), e is abstracted by an implicit arc of the topological rewritings labeling u. In this case, t and t' comes from the same term s, which has been rewritten along the substituted orbit. Therefore, any node variable x of L occurring in s has been respectively substituted by two variables of the substituted pattern L^O with the same image in the substituted orbit, *i.e.* two variable x_t and $x_{t'}$ such as $\uparrow^O_{r^O}(x_t) = \uparrow^O_{r^O}(v)$ and $\uparrow^O_{r^O}(x_{t'}) = \uparrow^O_{r^O}(v')$. Let us note $\langle \omega_u \rangle$ and $\langle \omega_x \rangle$ the respective labels of u and x. Because of term equivalence, the arc e has an antecedent e' in L^O such that its label is $j = \langle \omega_u \to \omega_x \rangle(i)$ and therefore, $x_{t'} = x_t.\alpha_j$. Because the condition of term stability of Definition 4 ensure that the term s must equivalent using either x or $x.\alpha_j$, therefore t and t' are equivalent. As equal terms produce equal values, v and v' satisfy the embedding consistency.
- v and v' have two respective antecedents u and u' in R (resp. L, K), e is an arc of R (resp. L, K). As r satisfy the embedding consistency condition, t and t' originate from a same term s. Moreover, as v and v' are connected with a rule arc, they are at the same position in the instance orbit, *i.e.* $\uparrow^O_{r^O}(v) = \uparrow^O_{r^O}(v')$. Therefore, t and t' have been rewritten with the same node variables at the same position, *i.e* for all variable $x \in V_L$ occurring in s, x is rewritten by the same variable $y \in V_{L^O}$ in t and t' as $\uparrow^O_{r^O}(v) = \uparrow^O_{r^O}(v') = \uparrow^O_{r^O}(y)$. The two terms are therefore equal, and as equal terms produce equal values, v and v' satisfy the embedding consistency. □

5 Conclusion

In this paper, we have presented a rule-based language for geometric modeling involving two types of variables, node variables and orbit variables, and provided them with a two-layered variable substitution mechanism. Orbit variables are first substituted, therefore defining the resulting topological structure and generating new node variables; these node variables are then substituted to compute the new embeddings. Moreover, rules written in a DPO style are provided with syntactic conditions ensuring the consistency preservation of embedded G-maps - *i.e.* by construction, transformed objects are also embedded G-maps. In particular, by introducing the conditions of term stability and terms equivalence, the syntactic conditions of embedding consistency preservation have been adapted to handle the new node variables generated by the orbit variable instantiation. This language is the core of Jerboa, a tool set for designing and generating geometric modelers.

References

1. Belhaouari, H., Arnould, A., Le Gall, P., Bellet, T.: JERBOA: A graph transformation library for topology-based geometric modeling. In: 7th International Conference

on Graph Transformation (ICGT 2014). LNCS, vol. 8571, pp. 269–284. Springer, York, UK (Jul 2014)
2. Bellet, T., Arnould, A., Le Gall, P.: Rule-based transformations for geometric modeling. In: 6th International Workshop on Computing with Terms and Graphs (TERMGRAPH 2011), Part of ETAPS. p. 20p. Saarbrücken, Germany (Apr 2011)
3. Bellet, T., Arnould, A., Le Gall, P.: Constraint-preserving labeled graph transformations for topology-based geometric modeling. Research report, XLIM (Feb 2017), `https://hal.archives-ouvertes.fr/hal-01476860`
4. Bellet, T., Poudret, M., Arnould, A., Fuchs, L., Le Gall, P.: Designing a topological modeler kernel: a rule-based approach. In: Shape Modeling International Conference (SMI). pp. 100–112. IEEE (2010)
5. Bohl, E., Terraz, O., Ghazanfarpour, D.: Modeling Fruits and Their Internal Structure Using Parametric 3Gmap L-systems. The Visual Computer 31(6-8), 819–829 (Jun 2015)
6. Damiand, G., Lienhardt, P.: Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing. A K Peters/CRC Press (Sep 2014)
7. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Monographs on Theoretical Computer Science, Springer (2006)
8. Habel, A., Plump, D.: Relabelling in graph transformation. In: Proceedings of the First International Conference on Graph Transformation. pp. 135–147. ICGT '02, Springer-Verlag, London, UK (2002)
9. Habel, A., Radke, H.: Expressiveness of graph conditions with variables. Electronic Communications of the EASST 30 (2010), Graph and Model Transformation 2010
10. Hoffmann, B.: Graph transformation with variables. In: Formal Methods in Software and Systems Modeling: Essays Dedicated to Hartmut Ehrig on the Occasion of His 60th Birthday. LNCS, vol. 3393, pp. 101–115. Springer, Berlin, Heidelberg (2005)
11. Hoffmann, B.: More on graph rewriting with contextual refinement. In: Echahed, R., Habel, A., Mosbah, M. (eds.) Graph Computation Models Selected Revised Papers from GCM 2014. Electronic Communications of the EASST, vol. 71 (2015)
12. Hoffmann, B., Jakumeit, E., Geiß, R.: Graph rewrite rules with structural recursion. In: Mosbah, M., Habel, A. (eds.) 2nd Intl. Workshop on Graph Computational Models (GCM 2008). pp. 5–16 (2008)
13. Jakumeit, E., Buchwald, S., Wagelaar, D., Dan, L., Hegedüs, Á., Herrmannsdörfer, M., Horn, T., et al.: A survey and comparison of transformation tools based on the transformation tool contest. Science of computer programming 85, 41–99 (2014)
14. Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.: Procedural modeling of buildings. In: ACM SIGGRAPH 2006 Papers. pp. 614–623. SIGGRAPH '06, ACM, New York, NY, USA (2006)
15. Orejas, F., Lambers, L.: Symbolic attributed graphs for attributed graph transformation. Electronic Communications of the EASST 30 (2010), Graph Computation Models 2010
16. Orejas, F., Lambers, L.: Lazy graph transformation. Fundamenta Informaticae 118(1-2), 65–96 (Jan 2012)
17. Pérez, J., Crespo, Y., Hoffmann, B., Mens, T.: A case study to evaluate the suitability of graph transformation tools for program refactoring. International Journal on Software Tools for Technology Transfer 12(3-4), 183–199 (2010)
18. Poudret, M., Arnould, A., Comet, J.P., Le Gall, P.: Graph Transformation for Topology Modelling. In: 4th International Conference on Graph Transformation (ICGT'08). LNCS, vol. 5214, pp. 147–161. Springer, Leicester, UK (Sep 2008)