Graph Transformation for Topology Modelling

Mathieu Poudret[1,2], Agnès Arnould[2], Jean-Paul Comet[3], and Pascale Le Gall[1,4]

[1] Programme d'Épigénomique, Génopole, F-91000 Évry
pascale.legall@epigenomique.genopole.fr
[2] XLIM-SIC UMR 6172 CNRS, Univ. de Poitiers, F-86962 Futuroscope
{poudret,arnould}@sic.univ-poitiers.fr
[3] I3S, UMR 6070 CNRS, Univ. de Nice-Sophia-Antipolis, F-06903 Sophia-Antipolis
comet@unice.fr
[4] MAS, Ecole Centrale Paris, Grande Voie des Vignes, F-92195 Châtenay-Malabry

Abstract. In this paper we present meta-rules to express an infinite class of semantically related graph transformation rules in the context of pure topological modelling with G-maps. Our proposal is motivated by the need of describing specific operations to be done on topological representations of objects in computer graphics, especially for simulation of complex structured systems where rearrangements of compartments are subject to change. We also define application of such meta-rules and prove that it preserves some necessary conditions for G-maps.

Keywords: topology-based geometric modelling, graph transformation, generalized map.

1 Introduction

Simulation of complex structured systems is a specialised area of topology-based modelling (or topological modelling for short). Topological models deal with the representation of the structure of objects (their decomposition into topological units: vertices, edges, faces and volumes) and with the neighbourhood relations between topological units. Thus topological structures are specific graphs. Among numerous topological models, generalized maps [Lie89, Lie94] (or G-maps) constitute a mathematically-defined model. Intuitively, edges between nodes indicate which nodes are neighbours and edge labels indicate which kind of neighbouring is concerned (i.e. connection of volumes, faces or edges). G-maps are thus a particular class of graphs with labelled edges defined by constraints ensuring that neighbouring relations are consistently organised. Topology-based modellers and simulators aggregate a large number of operations to edit objects. Most operations are designed to be dedicated to some application scopes. Moreover, they are usually implemented by a dedicated algorithm finely tuned in order to optimise its efficiency.

Using the framework of graph transformations [Roz97, EEPT06], we propose in this paper to model topological operations with transformation rules. Thus,

H. Ehrig et al. (Eds.): ICGT 2008, LNCS 5214, pp. 147–161, 2008.

we will be able to develop a simulator as a simple engine of rules applications. In a previous work, we defined transformation rules adapted to G-maps [PCG+07] using the algebraic approach of graph transformation based on labelled graphs and the double-pushout approach. Our first framework contains classical rules defined on an explicit pattern and a first class of meta-rules defined on patterns that carry isomorph topological units (volume, face, edge, vertex). This first proposal was satisfactory in the sense that we defined the four basic operations of G-maps (those from which all others can be defined) in terms of graph transformation rules. Even if we have already used our framework for the simulation of complex biological structured systems [PCLG+08], this first framework was not powerful enough to directly define complex topological operations. However, to facilitate the derivation of efficient simulation algorithms from high-level transformation rules, it becomes essential to be able to describe a large class of complex topological operations directly in term of transformation rules, instead of the composition of elementary topological operations. Indeed, we take advantage of such an approach both by ensuring for free some constraints of G-maps and by directly defining efficient algorithms by means of dedicated coverages of G-maps driven by the form of the considered high-level transformation rules.

In this paper, we present a more general class of meta-rules for G-maps which allows one to directly define a large class of topological operations. Intuitively, our meta-rules are built over graphs whose edges are labelled by new symbols playing the role of variables. The name of the symbols will indicate by which kind of topological units they can be substituted. So, our variables may be perceived as typed variables, each type representing a class of topological units of similar nature as volumes, faces or edges within the framework of G-maps. Thus, our meta-rules are more abstract and expressive than simple transformation rules over G-maps and take advantage of variables to generate a large family of basic transformation rules sharing the same effect according to a topological point of view. The use of variables to abstract graph transformation rules has been previously addressed [Hof05], in particular to model software transformations for refactoring purpose [HJE06]. In [Hof05], variables can be graph variables, attribute variables and cloning variables. In particular, cloning variables are mechanisms for duplicating some scheme extracted from the variable-based transformation, according to a given cardinality. The application of a transformation rule with cloning variables can then be expressed in term of application of simple transformation rules. In a similar approach, the application of our meta-rules will imply a mechanism of scheme cloning. However, our meta-rules will be specialised with respect to the underlying class of graphs on which they are applied, that is, the class of G-maps. Cloning mechanism will allow us to capture the class of all topological units of same nature (as volume, face, ...) which are of different size according to the considered 3D-object. Moreover, as G-maps are strongly constrained graphs, we will give some simple conditions on our meta-rules, ensuring both the dangling condition on all underlying basic transformation rules issued from the meta-rules and some of the constraints characterising G-maps among all labelled graphs.

The paper is organised as follows. Section 2 briefly presents graph transformation rules. Section 3 presents the G-map topological model. In Section 4, we introduce graph transformation meta-rules for modelling high-level topological operations and their application is defined by means of some intermediate cloning steps. In Section 5, we prove that some constraints of G-maps are preserved through the application of graph transformation meta-rules. Section 6 provides some concluding remarks.

2 Preliminaries

Let us first recall some notions and notations concerning graph transformations extracted from [EEPT06].

A graph G with labels in Σ_E is a couple (V, E) such that V is a set of vertices and $E \subset V \times \Sigma_E \times V$ is a set of non-oriented labelled edges. A path in G is a sequence $(v_0, l_1, v_1), (v_1, l_2, v_2), ..., (v_{k-1}, l_k, v_k)$ of E edges. We say that this path links v_0 to v_k and is labelled by the word $l_1 l_2 ... l_k \in \Sigma_E^*$. If $v_0 = v_k$, the path is called a cycle.

We introduce orbit graphs as particular sub-graphs, those which are generated by a vertex and an identified subset of labels. Indeed, these orbit graphs are useful to easily represent and manipulate topological cells (like faces or volumes) in the context of topological modelling.

Definition 1 (orbit). *Let us consider $G = (V, E)$ a graph with labels in Σ_E, $\{l_1, ..., l_k\} \subset \Sigma_E (k \geq 0)$ a set of labels and a vertex v of G.*

We call orbit $< l_1, ..., l_k > (v)$, the subset of V vertices reachable from v with paths labelled by words of $\{l_1, ..., l_k\}^*$. The orbit $< l_1, ..., l_k > (v)$ is said to be adjacent to v.

We call orbit graph $<< l_1, ..., l_k >> (v)$, the subgraph of G with vertices in $< l_1, ..., l_k > (v)$ and with edges in $\{(v', l, v") \in E \ / \ v', v" \in < l_1, ..., l_k > (v)$ and $l \in \{l_1, ..., l_k\}\}$.

A graph morphism $f : G \rightarrow H$ between two graphs G and H with labels in Σ_E, consists of two functions f_V from G vertices to H vertices and f_E from G edges to H edges, such that labelled edges are preserved[1]. Such a morphism is injective (resp. bijective) if both f_V and f_E are injective (resp. bijective). A bijective morphism is named isomorphism. G and H are said isomorphic if there exists an isomorphism $f : G \rightarrow H$.

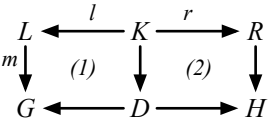
In the sequel, for our purposes, we only consider injective graph morphisms, which formalise the classical inclusion relation. Thus, we present the algebraic graph transformation approach and use the category **Graph** of graphs and graph morphisms (see chapter 2 of [EEPT06]).

A *production rule* $p : L \leftarrow K \rightarrow R$ is a pair of graph morphisms $l : K \rightarrow L$ and $r : K \rightarrow R$. L is the left-hand side, R is the right-hand side and K is the common interface of L and R. The left-hand side L represents the pattern of the

[1] For each edge (v, l, v') of G, $f_E((v, l, v')) = (f_V(v), l, f_V(v'))$.

rule, while the right-hand side R describes the production. K describes a graph part which has to exist to apply the rule, but which is not modified. Intuitively, $L\backslash K$ is the removed part[2] while $R\backslash K$ is the added part.

The rule p *transforms* G into a graph H, denoted by $G \Rightarrow_{p,m} H$, if there are a match graph morphism $m : L \to G$ and two square diagrams which are graph pushouts as in the following figure.



A direct graph transformation can be applied from a production rule p on a graph G if one can find a match m of the left-hand side L in G such that m is an (injective) morphism.

When a graph transformation with a production rule p and a match m is performed, all the vertices and edges which are matched by $L\backslash K$ are removed from G. The removed part is not a graph, in general, but the remaining structure $D := (G\backslash m(L)) \cup m(K)$ still has to be a legal graph (see following dangling condition), *i.e.* no edges should dangle (source and target vertices of all remaining edges should also remain). This means that the match m has to satisfy a suitable gluing condition, which makes sure that the gluing of $L\backslash K$ and D is equal to G (see (1) in the figure). In the second step of a direct graph transformation, D and $R\backslash K$ are glued together to obtain the derived graph H (see (2)).

More formally, we use graph morphisms $K \to L$, $K \to R$, and $K \to D$ to express how K is included in L, R, and D, respectively. This allows us to define the gluing constructions $G = L +_K D$ and $H = R +_K D$ as the pushout[3] constructions (1) and (2) in the figure, leading to a double pushout.

A graph morphism $m : L \to G$ from the left-hand side of a production rule $p : L \leftarrow K \to R$ to a graph G satisfies the *dangling condition* if no edge of $G\backslash m(L)$ is adjacent to a vertex of $m(L\backslash K)$. This dangling condition makes sure that the gluing of $L\backslash K$ and D is equal to G. Intuitively, all edges of G incident to a removed vertex are also removed.

Finally, a *graph transformation*, or, more precisely, a graph transformation sequence, consists of zero or more direct graph transformations.

3 Generalized Maps

The generalized maps (or G-maps) introduced by P. Lienhardt [Lie89, Lie94] define the topology of an n-dimensional subdivision space. G-maps allow the representation of the quasi-varieties, orientable or not. To represent cellular space

[2] The substraction $L\backslash K$ between two graphs $L = (V_L, E_L)$ and $K = (V_K, E_K)$ is defined from the set substraction on vertices and edges $L\backslash K = (V_L\backslash V_K, E_L\backslash E_K)$. Thus $L\backslash K$ may not be a graph.

[3] Let $f : A \to B$ and $g : A \to C$ be two graph morphisms, $D = B +_A C$ is the pushout object of B and C *via* A, or more precisely, *via* (A, f, g).

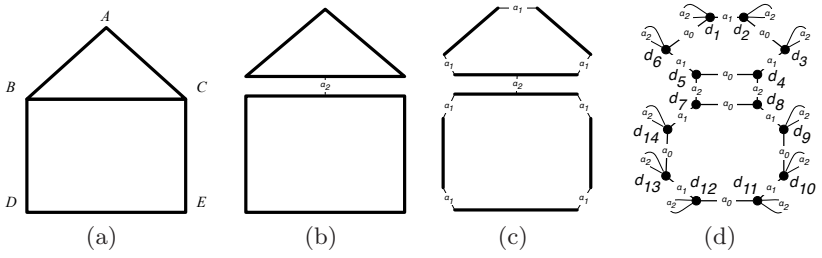


Fig. 1. Decomposition of a 2D object

subdivisions, we can choose other topological representations like combinatorial maps [Tut84]. Nevertheless, G-maps have the advantage of providing a homogeneous definition for all dimensions. Thus, operation specifications are simpler.

Intuitively, the main idea of G-maps is to decompose an object into basic elements, also called darts (graph vertices), which are connected together (with graph edges). The decomposition of a 2D object is shown in Fig. 1. The 2D object is displayed on Fig. 1(a). In Fig. 1(b), the object is split in order to focus on the two faces (topological 2-cells) which compose it. In an n-G-map, $n+1$ kinds of labelled-edges (from α_0 to α_n) allow one to recover the knowledge about neighbourhood relations between the topological cells. Thus, in Fig. 1(b), an α_2 edge makes explicit the adjacency relation which previously exists between faces ABC and $BCDE$. On Fig. 1(c), the faces are decomposed into lower dimension elements: the 1-cells. In the same manner, α_1 edges makes explicit the adjacency relations between the 1-cells. Finally, in the 2-G-map of Fig. 1(d), edges are split into α_0-connected 0-dimensional darts (represented with black dots). We notice that the index i of α_i labelled edges gives the dimension of the considered adjacency relation.

Definition 2 (G-map). *Let $n \geq 0$. An* n-dimensional generalised map *(or n-G-map) is a graph G with labels in $\Sigma_E = \{\alpha_0, \ldots, \alpha_n\}$, such that:*

- *The following $\mathcal{C}_G(\Sigma_E)$ condition is satisfied:*
  each vertex of G has exactly one adjacent l-edge for each label $l \in \Sigma_E$.
- *The following consistency constraint is satisfied:*
  for each pair of labels $\alpha_i, \alpha_j \in \Sigma_E$ such that $i+2 \leq j$, there exist a cycle labelled $\alpha_i\alpha_j\alpha_i\alpha_j$ from each vertex v of G.

The first condition of this definition (which is denoted by $\mathcal{C}$ for short, in the sequel) ensures that each vertex of an n-G-map has exactly $n+1$ adjacent edges labelled by $\alpha_0, ..., \alpha_n$. For example, in Fig. 1(d), the vertex d_4 is α_0-linked with d_5, α_1-linked with d_3, and α_2-linked with d_8. On the border of the objects, some darts do not have all of its neighbours. For instance, on Fig. 1(d) the vertex d_1 is α_0-linked to d_6 and α_1-linked to d_2, but d_1 is not linked to another vertex by an α_2-edge, because d_1 denotes the top corner of the object of Fig. 1(a) and thus is on the border of the 2D object. However, according to the $\mathcal{C}$ condition all

vertices must have exactly one adjacent label for each dimension. Thus, there is an α_2-loop adjacent to vertex d_1.

The second point of the n-G-map definition expresses some consistency constraints on the adjacency relations denoted by the labelled edges. Intuitively, in an G-map, if two i-dimensional topological cells are stuck together then they are stuck along a $(i-1)$-dimensional cell. For instance, on Fig. 1(d), the 2-cell defined by $\{d_1, ..., d_6\}$ is stuck with the 2-cell defined by $\{d_7, ..., d_{14}\}$ along the 1-cell defined by the four vertices $\{d_5, d_4, d_8, d_7\}$. The consistency constraint requires that there is a cycle $\alpha_0\alpha_2\alpha_0\alpha_2$ starting from each vertex of $\{d_5, d_4, d_8, d_7\}$. Thanks to loops, this property is also satisfied on object borders. For example, on the bottom of the object of Fig. 1(d), we have the cycle $\alpha_0\alpha_2\alpha_0\alpha_2$ from d_{11} and d_{12}.

The following definition explains the notion of i-cell in terms of G-map orbits.

Definition 3 (i-cell). *Let us consider G an n-G-map, v a vertex of G and $i \in [0, n]$. The i-cell adjacent to v is the orbit graph (see definition 1) of G $<< \alpha_0, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_n >> (v)$. The i-cell adjacent to v is noted i-cell(v).*

Let us illustrate this definition on the Fig. 1. The 2D geometric object Fig. 1(a) is composed of 0-cells (the geometric points A, B, C, D and E), 1-cells (the geometric segments AB, BC, AC, BD, DE and CE), and 2-cells (the two geometric faces ABC and $BCDE$). The corresponding 2-G-map Fig. 1(d) contains the same cells denoted by the following sub-G-maps. The geometric triangle ABC is denoted by 2-cell(d_1), *i.e.* the orbit graph $<< \alpha_0, \alpha_1 >> (d_1)$ which contains all vertices reachable from d_1 using α_0 and α_1 labelled edges. The geometric segment BC is denoted by the 1-cell(d_5), *i.e.* the orbit graph $<< \alpha_0, \alpha_2 >> (d_5)$ which contains the four vertices d_4, d_5, d_7 and d_8. The geometric points are denoted by 0-cells and their numbers of vertices depend on their numbers of adjacent segments. For example A (denoted by 0-cell(d_1), *i.e.* the orbit graph $<< \alpha_1, \alpha_2 >> (d_1)$), contains the two vertices d_1 and d_2.

We have already seen that applying a production rule on a graph requires to find a matching morphism satisfying the dangling condition. The following proposition shows that in case of graphs verifying the $\mathcal{C}$ condition (see definition 2), the dangling condition only depends on the form of the production rule, and that the derivation then preserves the $\mathcal{C}$ property.

Proposition 1. *Let p: $L \leftarrow K \rightarrow R$ be a production rule, and m: $L \rightarrow G$ be a match morphism on a graph G with labels in Σ_E which satisfies the $\mathcal{C}_G(\Sigma_E)$ condition.*

1. *m satisfies the dangling condition iff $L \backslash K$ satisfies the $\mathcal{C}_{L\backslash K}(\Sigma_E)$ condition*[4]*.*
2. *Moreover, if the rule p satisfies the following $\mathcal{C}_p(\Sigma_E)$ condition, then the derived graph H produced by the direct graph transformation $G \Rightarrow_{p,m} H$ satisfies the $\mathcal{C}_H(\Sigma_E)$ condition.*

[4] The condition $\mathcal{C}_G$ is defined for a graph G but can be extended for a structure which is not a graph. In this case, adjacent edges can dangle.

$\mathcal{C}_p(\Sigma_E)$: $\mathcal{C}_{L\backslash K}(\Sigma_E)$ and $\mathcal{C}_{R\backslash K}(\Sigma_E)$ are satisfied and each preserved vertex of K has the same adjacent labelled edges in L and R (i.e. $\forall v \in K$, $\forall l \in \Sigma_E$, v has an l-edge adjacent in L iff v has an l-edge adjacent in R and if they exist they are unique[5]).

Proof. Let us prove the first point. Let us suppose that m satisfies the dangling condition. By hypothesis, G satisfies the $\mathcal{C}_G(\Sigma_E)$ condition, thus for each deleted vertex v of $m(L\backslash K)$ and for each label $l \in \Sigma_E$ there exists a unique l-edge adjacent to v in G noted (v, l, v'). Thus, thanks to the dangling condition, (v, l, v') is an edge of $m(L)$. Because m is injective, $L\backslash K$ satisfies the $\mathcal{C}_{L\backslash K}(\Sigma_E)$ condition.

Reciprocally, let us suppose that $L\backslash K$ satisfies the $\mathcal{C}_{L\backslash K}(\Sigma_E)$ condition. Since G satisfies the $\mathcal{C}_G(\Sigma_E)$ condition, each edge of G adjacent to a vertex of $m(L\backslash K)$ is an edge of $m(L\backslash K)$. So, the dangling condition is satisfied.

Let us now prove the second point. Let us suppose that G, the removed structure $L\backslash K$ and the created structure $R\backslash K$ satisfy, respectively, $\mathcal{C}_G(\Sigma_E)$, $\mathcal{C}_{L\backslash K}(\Sigma_E)$ and $\mathcal{C}_{R\backslash K}(\Sigma_E)$ conditions and that each preserved vertex of K has the same labelled edges in left-hand side L and in right-hand side R. Thanks to the first point, the dangling condition is satisfied and thus the direct graph transformation $G \Rightarrow_{p,m} H$ exists. Let v be a vertex of H and $l \in \Sigma_E$ a label:

- If v is not a matched vertex, *i.e.* v is a vertex of $G\backslash m(L)$. Thanks to the $\mathcal{C}_G(\Sigma_E)$ condition, there exists a unique l-edge adjacent to v in G, noted (v, l, v'). v' may be a vertex of $m(L)$ or not, but (v, l, v') is not a matched edge, *i.e.* (v, l, v') is not an edge of $m(L)$, because L is a graph. Thus thanks to the direct graph transformation, (v, l, v') is the unique l-edge adjacent to v in H;
- If v is an added vertex, *i.e.* v is not a vertex of G. Thanks to direct graph transformation, there exist a vertex u of R such that the double pushout produces v in H from u. However, thanks to hypothesis, R have exactly one l-edge adjacent to u. Thus H have exactly one l-edge adjacent to v;
- If v is a matched vertex, *i.e.* v is a vertex of $m(K)$. Thanks to the $\mathcal{C}_G(\Sigma_E)$, there exists an unique l-edge adjacent to v in G, noted (v, l, v'). And thanks to the m injectivity, there exists a unique vertex u in L such that $m_V(u) = v$.
  - If there does not exist any l-edge adjacent to u in L, thanks to hypothesis, there does not exist any l-edge adjacent to u in R. Thus, (v, l, v') is an edge of $G\backslash m(L)$ and thanks to direct graph transformation, (v, l, v') is an edge of H. Moreover, thanks to the $\mathcal{C}_G(\Sigma_E)$ condition, (v, l, v') is the unique l-edge adjacent to v in H;
  - If there exists an edge (u, l, u') in L, thanks to hypothesis, it is the unique l-edge adjacent to u in L and there exists an unique l-edge adjacent to u in R noted (u, l, u''). Moreover, thanks to the $\mathcal{C}_G(\Sigma_E)$ condition, the unique l-edge adjacent to v in G is $m_E((u, l, u'))$. Thus the double pushout of the direct graph transformation produces an unique l-edge adjacent to v from the (u, l, u'') edge.

[5] We suppose, without loss of generality, that the morphisms l and r of the double-pushout figure (see section 2) are the identity.

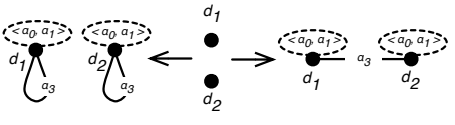
Consequently, there exists a unique l-labelled edge adjacent to v in H. In other words, H satisfies the $\mathcal{C}_H(\Sigma_E)$ condition. □

The proposition 1 ensures that all derivations with an adequate production rule preserve the $\mathcal{C}$ condition of G-maps (see definition 2). Let us notice that like in classical operation definitions (mathematical definitions, algorithms or formal specifications), the G-map consistency constraint (second point of definition 2) has to be verified individually for each production rule.

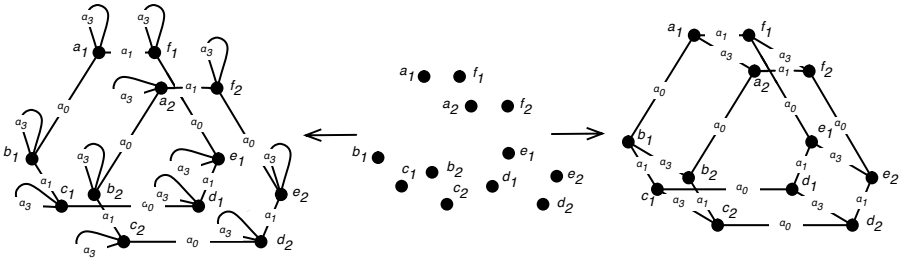
4 Topological Operations in Terms of Graph Transformation

The set of basic topological operations for G-maps has been defined [Lie89] and includes different operations, namely vertex addition, vertex suppression, sew and unsew. In previous works [PCG+07], we have shown that first and second operations can be directly translated into transformation rules satisfying the $\mathcal{C}$ condition and moreover the consistency constraint of G-map (see definition 2).

Nevertheless, both sew and unsew operations are generic and cannot be defined directly in terms of graph transformation rules because they depend on the orbits. To overcome this limitation, we introduced in [PCG+07] a concept of graph transformation meta-rules which abstracts a set of graph transformation rules along an orbit. The idea is to propagate a local transformation pattern (expressed on a few vertices) along an orbit of the graph, independently of the form of this orbit. To specify which part of the local pattern is associated to the elements of the orbit, we introduce an additional label, which denotes the orbit. Graphically, these meta-labelled edges are noted with dotted lines. Thus the 3-sew meta-rule (which aims at sticking two volumes along one face) of Fig. 2(a)

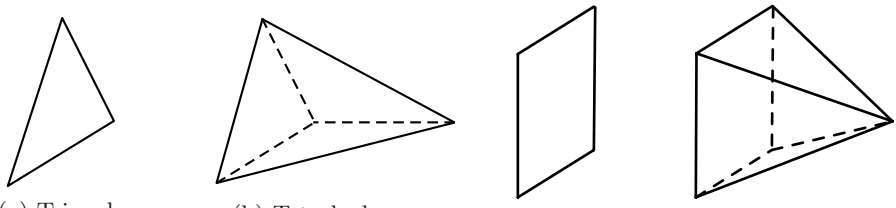


(a) 3-sew meta-rule



(b) 3-sew of triangular faces

Fig. 2. 3-sew rules



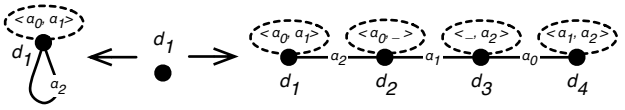
(a) Triangle (b) Tetrahedron (c) Square (d) Pyramid

Fig. 3. Cone operation

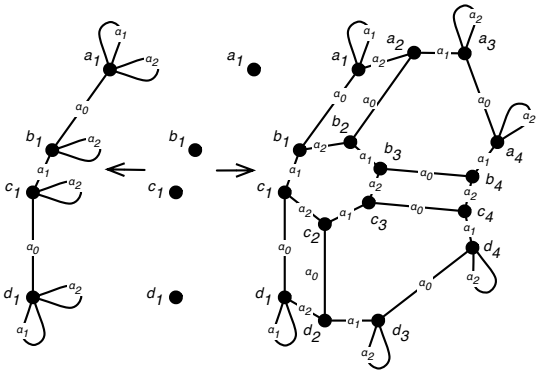
may be applied along a triangular face to define the classical rule of Fig. 2(b), or along any other face orbit. More precisely, a meta-edge $(d_1, < \alpha_0, \alpha_1 >, d_1)$ of a meta-rule specifies a sub-graph labelled on $\{\alpha_0, \alpha_1\}$ and thus matches an orbit graph $<< \alpha_0, \alpha_1 >> (d_1)$ in each classical rule (see Fig. 2(b)). The pattern connected to d_1, compounded of a α_3 classical loop, must be repeated along this orbit. Thus, Fig. 2(b) vertices a_1, ..., f_1, a_2, ..., f_2 have an α_3 loop. Finally, $(d_1, < \alpha_0, \alpha_1 >, d_1)$ and $(d_2, < \alpha_0, \alpha_1 >, d_2)$ must be expended in two isomorphic orbit graphs. Thus, $<< \alpha_0, \alpha_1 >> (d_1)$ and $<< \alpha_0, \alpha_1 >> (d_2)$ are isomorphic faces in Fig. 2(b).

This previous framework is enough to specify basic operations, and thus is complete because all 3-G-map operations may be specified from the basic ones. But, from a user point of view, to specify an operation as a large composition of basic operations is less easy and efficient that specifying it directly. Unfortunately, the previous framework is not general enough to directly specify most of complex operations. Indeed, previous meta-rules are defined along a unique orbit, thus every meta-edges are expended as isomorphic orbit graphs. For example, the four meta-edges of sew rule Fig. 2(a) are expended to four isomorphic triangular faces (see Fig. 2(b)). But, for most operations we need to match (and/or to produce) different kinds of orbit graphs. In the cone operation (which aims at producing a cone-shaped volume from one base face), different kinds of orbit graphs are necessary to produce, for instance, a tetrahedron from a triangular face or a pyramid from a square face (see Fig. 3). This operation cannot be defined from several copies of the base 2-cell. But, it may be defined from copies of base vertices linked together in the right manner. Especially, the top 0-cell of a cone is dual[6] of the base 2-cell. Intuitively, the 2-cells adjacent to the base are also adjacent to the top. The classical rule of Fig. 4(b) defines the cone operation on a face corner. Here, the top orbit graph $<< \alpha_1, \alpha_2 >> (d_4)$ is a copy of the base orbit graph $<< \alpha_0, \alpha_1 >> (d_1)$ with a renaming of links. Thus, when a_1 and b_1 are α_0 linked, a_4 and b_4 are α_1 linked and when b_1 and c_1 are α_1 linked, b_4 and c_4 are α_2 linked. Moreover, when a_1 and b_1 are α_0 linked, a_2 and b_2 are also α_0 linked and when b_1 and c_1 are α_1 linked, b_2 and c_2 are not linked. The α_2 loop of the left-hand side of the rule figure 4(a), means that only isolated faces (which are not linked to another one) can be matched in order to produce cones.

[6] Two topological cells are dual if they are isomorphic up to a renaming of their labels.



(a) Cone meta-rule



(b) Rule of a corner cone

Fig. 4. Cone rules

The following definition allows us to generalise graph and production rules notions by adding meta-edges that denote isomorphic orbit graphs up to a renaming of their edges labels.

Definition 4 (meta-graph and production rule). *Let $\beta = \{\alpha'_1, ..., \alpha'_k\} \subset \Sigma_E$ a subset of labels and Γ_β be the set of all renaming functions $\gamma : \beta \to \Sigma_E \cup \{\_\}$. A renaming function γ is named meta-label and is said full if $\gamma(\beta) \subset \Sigma_E$ (without "_")*[7].

A meta-graph *on β, or meta-graph, is a graph with label in $\Sigma_E \cup \Gamma_\beta$ such that each meta-labelled edge is a loop. A meta-graph is said full if all its meta-labels are full. Graphically, a meta-loop γ is labelled by the renamed orbit $< \gamma(\alpha'_1), ..., \gamma(\alpha'_k) >$.*

A production meta-rule *on β, or meta-rule, is a production rule $p : L \leftarrow K \rightarrow R$ on the full sub-category of generalised meta-graph on β, such that the meta-graph L is full.*

The meta-rule Fig. 4(a) specifies the cone operation. In this example, we can see four different kinds of orbits: a full orbit graph for the base (2-cell $<< \alpha_0, \alpha_1 >> (d_1)$), two partial ones for the side faces[8] (2-cells $<< \alpha_0, \_ >> (d_2)$ and $<< \_, \alpha_2 >> (d_3)$) and another full one for the top (0-cell $<< \alpha_1, \alpha_2 >> (d_4)$). All of them are translated copies of matched 2-cell $<< \alpha_0, \alpha_1 >> (d_1)$, using respectively renaming functions $\gamma_1 : \alpha_0 \mapsto \alpha_0, \alpha_1 \mapsto \alpha_1$, $\gamma_2 : \alpha_0 \mapsto \alpha_0, \alpha_1 \mapsto \_$,

[7] Where $\gamma(\beta)$ names the set $\{\gamma(l) \mid l \in \beta\}$.

[8] Formally, this two subgraphs are not orbits in the sense of definition 1.

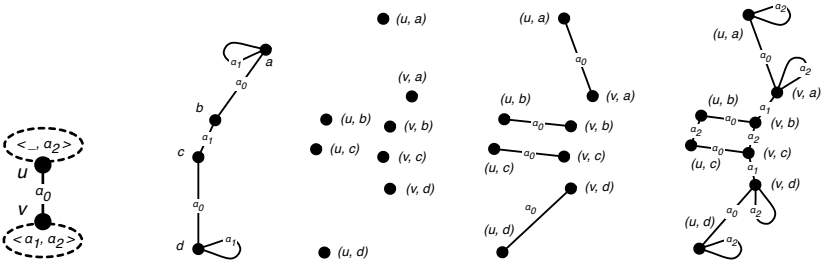


Fig. 5. Expansion of a meta-graph G on $\{\alpha_0, \alpha_1\}$ along a graph O

$\gamma_3 : \alpha_0 \mapsto \_, \alpha_1 \mapsto \alpha_2$ and $\gamma_4 : \alpha_0 \mapsto \alpha_1, \alpha_1 \mapsto \alpha_2$. By lack of space, we do not explain how by means of similar production rules, we can express other topological operations like the extrusion operation (to create box from a face) or the rounding operation (to round angular edges or vertices).

As seen on examples, the semantic of meta-graph patterns is given by expanding the meta-patterns along an orbit on β.

Definition 5 (expansion). *Let $\beta = \{\alpha'_1, ..., \alpha'_k\} \subset \Sigma_E$ be subset of labels and O be a graph with labels in β. The* expansion *of a meta-graph G on β along O is the Cartesian-like product $G \times O$ such that:*

- *The set of vertices is the Cartesian product of vertex sets $\{(u, a) \mid u$ is a vertex of G and a is a vertex of $O\}$;*
- *The set of edges is $\{((u, a), l, (v, a)) \mid (u, l, v)$ is an edge of G and a is a vertex of $O\} \cup$ $\{((u, a), \gamma(l), (u, b)) \mid (u, \gamma, u)$ is a meta-edge of G, $\gamma(l) \in \Sigma_E$ and (a, l, b) is an edge of $O\}$.*

The expansion of a morphism $f : G \to H$ along O, is the morphism $f \times O : G \times O \to H \times O$ which associates the vertex $(f_V(u), a)$ of $H \times O$ to each vertex (u, a) of $G \times O$.

The expansion of a production meta-rule $p : L \xleftarrow{l} K \xrightarrow{r} R$ along O is the production rule $p \times O : L \times O \xleftarrow{l \times O} K \times O \xrightarrow{r \times O} R \times O$.

In Fig. 5, we expand a graph G on $\{\alpha_0, \alpha_1\}$ (see Fig. 5(a)) along a graph O labelled in $\{\alpha_0, \alpha_1\}$ (see Fig. 5(b)). Actually, G is extracted from the right-hand side of the cone meta-rule (as shown in Fig. 4(a)) and O represents corner 2-cell. The first step of the expansion process (see Fig. 5(c)) consists in copying the vertices of G along O (computing $V_G \times V_O$). The next steps consist in, respectively, copying classical edges of G along O (see Fig. 5(d)) and copying renamed edges of O along the G meta-edges (see Fig. 5(e)). Then, Fig. 4(b) is obtained by expansion of cone meta-rule Fig. 4(a) along a face corner pattern.

Proposition 2. *Let $f : G \to H$ be a morphism between the two meta-graphs G and H on β, and O a graph with label in β. The expansion $f \times O$ always exists.*

Proof. For each edge (u, l, v) of G and each vertex a of O, $((u, a), l, (v, a))$ is an edge of $G \times O$ and $((f_V(u), a), l, (f_V(v), a))$ is an edge of $H \times O$.

For each meta-edge (u, γ, u) of G and each edge (a, l, b) of O, if $\gamma(l) \in \Sigma_E$ then $((u, a), \gamma(l), (u, b))$ is an edge of $G \times O$ and $((f_V(u), a), \gamma(l), (f_V(u), b))$ is an edge of $H \times O$.

$G \times O$ has no other edge. □

The following proposition shows that the expansion does not depend on β labels. Its proof is left to the reader.

Proposition 3. *Let β and δ be two subset of labels, $\iota : \delta \to \beta$ a bijective function, G be a meta-graph on β and O a graph labelled in β.*

Let H be the meta-graph on δ obtained from G by renaming each meta γ-loop to a $\gamma \circ \iota$-loop and P be the graph labelled on δ obtained from O by renaming each label of O along ι^{-1}. Then we have $G \times O = H \times P$.

The previous proposition founds the graphical notation of meta-loops with implicit renaming functions.

By definition, if there exist several meta-edges on the left-hand side and on the right-hand side of a production meta-rule, the expansion replaces all these meta-edges with distinct sub-graphs (each of them is isomorphic, up to a renaming of their edges labels, to the β-labelled graph O).

Definition 6 (direct graph meta-transformation). *Let G be a graph labelled on Σ_E and $p : L \leftarrow K \rightarrow R$ a production meta-rule on β.*

The meta-rule p direct meta-transforms *G into a graph H labelled on Σ_E, denoted $G \Rightarrow_{p,O,m} H$, if there are a graph O with labels in β and a match morphism $m : L \times O \rightarrow G$ such that $G \Rightarrow_{p \times O, m} H$ is a direct graph transformation.*

Classically, a *graph meta-transformation*, or more precisely, a graph meta-transformation sequence, consists in zero or more direct graph transformations. We should notice that, a production rule without any meta-edge can be seen as a meta-rule on the empty set. Indeed, such production meta-rules and the corresponding classical production rule allow one to produce the same direct transformed graphs.

5 Consistency of G-Maps and Transformation Rules

We have already seen that applying a production meta-rule on a graph requires to find a matching morphism which satisfies the dangling condition. The following proposition shows that, as in proposition 1, in case of graphs in which each vertex has exactly one adjacent l-edge for each label l (i.e. the condition $\mathcal{C}$), the dangling condition uniquely depends on the form of the production meta-rule, and that the derivation then preserves the $\mathcal{C}$ property. Let us first define the extension of condition $\mathcal{C}$ (see proposition 1) to meta-graphs and meta-rules:

$\mathcal{C}_G(\Sigma_E)$ Let G be a graph on β. For each label $l \in \Sigma_E$, each vertex has exactly one adjacent edge s. t. either it is l-labelled or it is γ-labelled with $l \in \gamma(\beta)$;

$\mathcal{C}_p(\Sigma_E)$ Let p be the rule $L \leftarrow K \rightarrow R$ on β, $\mathcal{C}_{L\setminus K}(\Sigma_E)$ and $\mathcal{C}_{R\setminus K}(\Sigma_E)$ are satisfied and each preserved vertex of K has the same adjacent labelled edges in L and R (in the extended way).

Proposition 4. *Let $p : L \leftarrow K \rightarrow R$ be a production rule on β, O be a graph with labels in β and $m : L \times O \rightarrow G$ be a match morphism on a graph G with labels in Σ_E which satisfies the $\mathcal{C}_G(\Sigma_E)$ condition.*

1. *m satisfies the dangling condition iff O satisfies the $\mathcal{C}_O(\beta)$ condition and $L\setminus K$ satisfies the $\mathcal{C}_{L\setminus K}(\Sigma_E)$ condition.*
2. *Moreover, if the rule p satisfies the $\mathcal{C}_p(\Sigma_E)$ condition, then the derived graph H produced by the direct graph transformation $G \Rightarrow_{p,0,m} H$ satisfies the $\mathcal{C}_H(\Sigma_E)$ condition.*

Proof. Let us first prove the following lemma: for each vertex u of a meta-graph G and each vertex a of O, vertices u in G and (u, a) in $G \times O$ have the same labelled edges (in the extended way).

- If u has an adjacent l-labelled edge (u, l, v) in G, then (u, a) has an adjacent l-labelled edge $((u, a), l, (v, a))$ in $G \times O$;
- If u has an adjacent meta γ-edge in G, and a has an adjacent l-labelled edge (a, l, b) in O, such that $\gamma(l) \in \Sigma_E$, then (u, a) has an adjacent labelled edge $((u, a), \gamma(l), (u, b))$ in $G \times O$; And by definition, $G \times O$ has no other edges.

The proof of the proposition lies directly in this lemma. □

The condition of the proposition 4 ensures that a full γ-edge in the left-hand side of the meta-rule matches a complete $< \gamma(\beta) >$-orbit of the transformed graph and respectively full γ-edges of right-hand side match complete $< \gamma(\beta) >$-orbit of produced graph.

Thanks to proposition 4, a transformation of a G-map along the cone meta-rule of Fig. 4(a) preserves the $\mathcal{C}$ property of G-maps. Since each vertex of the cone meta-rule has exactly three links labelled by α_0, α_1 and α_2 (in extended way), the expanded rule (see Fig. 4(b) for example) has the same property.

Moreover, it is easy to prove that the consistency property of G-maps (see second condition of definition 2) is preserved by application of the cone meta-rule. Indeed, in the left-hand side, because d_1 has a α_2-loop, its α_0-neighbour has also an α_2-loop. Moreover, in all expanded rules along a graph O, because of $\mathcal{C}_O(\{\alpha_0, \alpha_1\})$, each expanded vertex has an α_2-loop and an α_0-neighbour. And thus, each expanded vertex has an $\alpha_0\alpha_2\alpha_0\alpha_2$ labelled cycle. For example, in the cone expansion Fig. 4(b), a_1b_1 and c_1d_1 are two $\alpha_0\alpha_2\alpha_0\alpha_2$ labelled cycles.

In the right-hand side, since d_1 and d_2 are α_2-linked together, their α_0-neighbours are also α_2-linked together. Indeed, because of $\mathcal{C}_O(\{\alpha_0, \alpha_1\})$, each vertex of O has an α_0-edge. And thus, each expanded vertex has an $\alpha_0\alpha_2\alpha_0\alpha_2$ cycle. In the cone rule example Fig. 4(b), $a_1b_1b_2a_2$ and $c_1d_1d_2c_2$ are two $\alpha_0\alpha_2\alpha_0\alpha_2$ labelled cycles. In the same way, d_3 and d_4 are α_0-linked together, thus their α_2-neighbours are also α_0-linked together. In the cone rule example Fig. 4(b), a_3a_4, $b_3b_4c_4c_3$ and d_3d_4 are three $\alpha_0\alpha_2\alpha_0\alpha_2$ labelled cycles.

6 Conclusion and Perspectives

In this paper we focus on the formalisation of complex topological operations on G-maps. Pursuing previous works, we propose a general class of meta-rules for G-maps which allows us to directly define a large class of topological operations, helpful in the context of modelling of complex structured systems, as the cone operation taken as illustration in the paper. We prove that thanks to strong G-maps constraints concerning edge labelling, the dangling condition of a meta-rule can be statically verified independently of the G-map on which it is applied. We will search for sufficient syntactical conditions on rules to ensure G-map consistency constraints.

This rule-based approach to specify topological evolution of objects will be useful for coupling transformations of objects with more classical rule-based approaches for simulating complex systems. In the context of modelling of complex biological systems, such a simulation paradigm has been broadly considered leading to an enormous amount of successfull applications [CFS06, RPS+04, Car05]. In these models, the compartmentalisation captures a static topology (focusing on exchange between compartments and molecular interactions) or simple topological modifications (resulting, for example, from endocytosis or exocytosis). Nevertheless, although biological systems are composed of molecules, the structure of the system and components both play essential roles in the biological functions of the system. Indeed the understanding of biological systems needs to take into account molecular phenomena (possibly abstracted by continuous concentrations), communication channels and space structuring of the cells at a same accuracy level. Thus it is an important challenge to understand the effects of spatial structure on the different concentrations, and reciprocally, the consequences of the evolution of concentrations on the spatial structure.

A general framework for rule-based simulations taking into account both molecular phenomena and subcellular compartment rearrangments would handle embedded G-maps. In previous work [PCG+07], we sketched out embedded G-maps by associating labels with vertices to represent geometric aspects (shapes of objects, distances between them, etc.) and by associating other labels to represent biochemical quantities (protein concentrations, protein fluxes through a subcellular wall, etc.). We have already used such topological transformation rules to simulate the evolution of biological subcellular compartments [PCLG+08]. To apply topological transformation rules, we have first to match the left-hand side of a rule. The pattern-matching problem is recognised as difficult in the general case of general graphs (without any constraint). In the particular case of G-maps, we have applied heuristics derived from usual G-map coverage involved in classical computer graphics operations. Our future works will then focus on the definition of embedded G-maps, and of associated graph transformation rules. Then it will be mandatory to study the conditions which ensure that the application of a transformation rule leads to another embedded G-maps in a coherent way.

References

[Car05] Cardelli, L., Calculi, B.: Interactions of biological membranes. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 257–280. Springer, Heidelberg (2005)

[CFS06] Calzone, L., Fages, F., Soliman, S.: Biocham: an environment for modeling biological systems and formalizing experimental knowledge. Bioinformatics 22(14), 1805–1807 (2006)

[EEPT06] Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. In: Monographs in Theoretical Computer Science. Springer, Heidelberg (2006)

[HJE06] Hoffmann, B., Janssens, D., Van Eetvelde, N.: Cloning and expanding graph transformation rules for refactoring. Electr. Notes Theor. Comput. Sci. 152, 53–67 (2006)

[Hof05] Hoffmann, B.: Graph transformation with variables. In: Kreowski, H.-J., Montanari, U., Orejas, F., Rozenberg, G., Taentzer, G. (eds.) Formal Methods in Software and Systems Modeling. LNCS, vol. 3393, pp. 101–115. Springer, Heidelberg (2005)

[Lie89] Lienhardt, P.: Subdivision of n-dimensional spaces and n-dimensional generalized maps. In: SCG 1989, pp. 228–236. ACM Press, New York (1989)

[Lie94] Lienhardt, P.: n-dimensional generalised combinatorial maps and cellular quasimanifolds. In: IJCGA (1994)

[PCG+07] Poudret, M., Comet, J.-P., Le Gall, P., Arnould, A., Meseure, P.: Topology-based geometric modelling for biological cellular processes. In: LATA 2007, Tarragona, Spain, March 29 - April 4 (2007), `http://grammars.grlmc.com/LATA2007/proc.html`

[PCLG+08] Poudret, M., Comet, J.-P., Le Gall, P., Képès, F., Arnould, A., Meseure, P.: Topology-based abstraction of complex biological systems: Application to the Golgi apparatus. Theory in Biosciences (2008)

[Roz97] Rozenberg, G. (ed.): Handbook of Graph Grammars and Computing by Graph Transformations, Foundations, vol. 1. World Scientific, Singapore (1997)

[RPS+04] Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: an abstraction for biological compartments. Theor. Comput. Sci. 325(1), 141–167 (2004)

[Tut84] Tutte, W.: Graph Theory. Encyclopedia of Mathematics and its Applications, vol. 21. Addison-Wesley, Reading (1984)