Global Graph Transformations

Luidnel Maignan and Antoine Spicher

University Paris-Est Créteil, LACL
61 avenue du Général de Gaulle
F-94015 Créteil Cedex, France
{luidnel.maignan,antoine.spicher}@u-pec.fr

Abstract. In this paper, we consider *Global Graph Transformations* where all occurrences of a set of predefined local rules are applied altogether synchronously so that each part of the original graph gives rise to a part of the result graph, without any reference to the original one. The particularity here is that our framework is deterministic. This is achieved by incorporating a notion of mutual agreement between its local rules. Our proposition is first motivated and illustrated on existing problems coming from different domains. It is then formalized as a categorical construction which is finally compared to more usual algebraic constructions, in particular to the strongly related Amalgamation Theorem. Applications of this work include the generalization of cellular automata and the clarification of some frameworks of complex systems modeling where the usual mutual exclusion of rule applications can be replaced by a concept of mutual agreement.

1 Introduction

The framework proposed herein has been designed with the need to model *deterministic* dynamical systems by graph transformations. The state of such a system is represented by a graph and its *global* dynamics is specified through a set of *local* evolution rules.

One such example is the simple framework of cellular automata (CA). The state of a CA is usually represented by a labeled regular graph where the nodes are the cells, the labels encode the cell states, and the edges represent the neighborhood relation between cells. A global evolution step consists of a synchronous update of all the labels relying on a local evolution function, the structure of the graph remaining unchanged. From another point of view, each patch of neighbor cells (e.g. triple of consecutive cells in 1D CA with radius 1, square of 9 cells in 2D Moore CA) gives rise to a new node with the updated label. All these new nodes are then connected together to form the next state graph representation which is independent from the current one. Recently, different formalizations based on this point of view have been proposed to generalize CA to arbitrary graphs with dynamic structures [1,2]. This paper is an attempt to show that an algebraic approach can provide an interesting alternative to those formalisms.

Other work was already devoted to the simulation of the so called *Dynamical Systems with Dynamical Structure* [5,20]. In these approaches, a rewriting of cell complexes (an extension of graphs to higher dimensions, see Section 3) has been developed to simulate concurrent interaction rules. The resulting programming language, called MGS, has been shown as a unification of many computation models including CA, Lindenmayer systems, membrane computing systems, etc. So far, MGS parallel rule application strategies rely on a maximal-parallel property: only mutually exclusive matchings can be applied simultaneously. This leads to some non-determinism since there might be different maximal sets of mutually exclusive matchings. This paper arises as an attempt to model systems where the natural mutual agreement between the local rules applications can be used to overcome the difficulties introduced by the concept of mutual exclusion.

As a result of these two motivations, the transition mechanism of the modeled system is described in our setting by a set of rules that do not send rooted graphs to nodes as in CA, nor nodes to graphs as it is often the case in graph transformation [7], but that send graphs to graphs. Moreover, the reconstruction of the resulting global state from the local applications of these rules on the current state, relies on the following coherence property of the rules: when two rule matchings overlap on an input graph, the local behavior of the common part has to be shared by the two rules. This intuition can be seen as a compound of two instances of a simpler situation: any time a first matching includes a second matching, the result of the first matching has to include the result of the second matching. Because the proposed formalism is a direct expression of the coherence property, we believe that this framework allows to model very intuitively any desired deterministic system, and can be easily adapted to any particular need (*e.g.*, non-determinism, presence of terminals/non-terminals). Of course, as for any modeling framework, it certainly asks for some getting-used-to to users already accustomed with other modes of thinking.

The proposed framework reminds some existing ones. The statement of the coherence above is the same leading to the concept of sub-rule and amalgamation in the double-pushout approach [3]. It is also reminiscent of the connecting or gluing mechanisms used in node or edge based parallel graph grammars (see Sect. 2). This very simple inclusion intuition can be applied in various settings, as cell complexes in the following. In section 5, it is expressed categorically; this allows a short and intrinsic formalization with possible instantiations to different kinds of objects. Finally the evolution described by a total function in the CA setting simply becomes functors on a full subcategory in our setting.

Organization of the Paper. The rest of the paper is organized as follows. Section 2 provides some comparison with existing approaches of parallel graph transformation. Section 3 gives some formal preliminaries. Section 4 considers the example of triangular mesh refinement in order to expose the idea of the proposed framework informally. This example has been chosen because it encompasses many considerations about the proposed framework. Section 5 formalizes the concept of global transformation. It is then compared with the double-pushout approach and the strongly related concept of amalgamation of productions. Section 6 discusses the remaining aspect of the work and concludes.

2 Related Work

In this section, we describe briefly existing approaches of parallel rewriting systems with a final comparison to global transformations.

One of the first parallel rewriting systems are *Lindenmayer Systems* (LS), which are parallel string rewriting in contrast with sequential Chomsky grammars. A $\langle k, l\rangle$ LS ($k, l \in \mathbb{N}$) is a context-sensitive system gathering productions in a transition table, so that each sub-word of length $k + 1 + l$ (*i.e.*, a letter with a left context of length k and a right context of length l[1]) is associated with (possibly many) words. The parallel rewriting of a word is done as follows: each letter is substituted accordingly to its left and right contexts by one of its associated words in the transition table. Obviously, the LS is deterministic if the transition table is a function (*i.e.*, associates a unique word with each entry). LS have been extensively studied for their expressive powers and compared to the classic Chomsky hierarchy of formal languages [19]. They have also been used in the modeling of unidimensional and tree-like dynamical systems [16]. LS can be seen as a special case of parallel graph transformation, restricted to linear/sequential graphs. Other special cases of graphs can be addressed. For example, Paŭn Systems (roughly speaking, nested parallel multiset rewriting systems [15]) can be considered as complete graph transformations. Such systems derive naturally from LS by considering rewriting modulo associativity and commutativity: any two symbols of a string representing a multiset can be made neighbors by permuting the letters of the word. In this setting, a production is a metaphor of a chemical reaction. The left hand side (l.h.s.) of a production designates a sub-multiset which is entirely replaced by its associated right hand side (r.h.s.) (in contrast with LS where each letter is replaced independently). To avoid the consumption of the same symbol by two different reactions, the maximal-parallel strategy is considered leading to non-determinism.

Extension of LS to arbitrary graph transformation is not an easy task [7]. A first idea consists in encoding the graph using sets, multisets, sequences or terms, and their associated well-known and rich techniques. As an example, [17] bridges graph rewriting to set rewriting by considering a graph as a set of (hyper-)edges, an hyper-edge being a sequence of vertices. On the other hand, some works address the issue of a direct rewriting of graphs. Classical work in graph grammars includes node-rewriting and hyperedge-rewriting graph grammars [18]. These works have some interesting relation with our framework when they are used in a parallel setting, that is, when each node (resp. edge) chooses a production rule to apply. In this case, all of them provide a resulting graph and these graphs are connected (resp. glued) together in some way or another by an *embedding mechanism*. In the node setting, edges are used to specify the connection between the graphs while, in the hyperedge setting, the nodes specify the gluing between the graphs. In fact, any deterministic instances of these systems can easily be represented in our framework.

[1] An extra dummy symbol, the *marker*, is used to deal with boundaries.

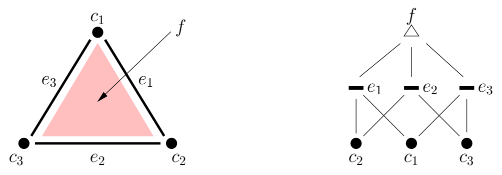


Fig. 1. On the left, a cell complex composed of three 0-cells (c_1, c_2, c_3), of three 1-cells (e_1, e_2, e_3) and of a single 2-cells (f). On the right, the Hasse diagram of its incident relationship.

The previous examples are set-theoretic approaches: graph transformations are expressed in terms of sets and set operations. Graph transformations can also be represented algebraically using the double-pushout and the simple-pushout approaches which formalize the idea of local replacement in a categorical manner [18]. The double-pushout approach is inherently local so that it needs to be extended to deal with parallel applications leading to the concepts of *parallelism* [4] and *amalgamation* [3]. Roughly speaking, parallelism allows to apply many mutually exclusive matchings simultaneously. Amalgamation provides a more general setting where the set of productions is augmented with sub-productions that handle some kind of overlaps. Therefore, many matchings can be applied together as long as sub-productions are chosen to deal with the overlapping sub-parts which is reminiscent of the coherence property stated above. Multi-amalgamation [6] is an extension of amalgamation to consider maximal matchings. However, the amalgamation theorem makes clear that a compound production can be applied only when each of its parts can be applied. This constraint makes (multi-)amalgamation unusable straightforwardly in the cases where the transformation of matchings only makes sense when taken altogether.

3 Formal Preliminaries and Notations

Since the present article follows naturally work introduced in [20], we consider *cell complexes*, a more general setting than graphs. Like a graph, a cell complex is a formal construction that builds a space in a combinatorial way through more basic objects called *topological cells.* Each topological cell abstractly represents a part of the whole and is characterized by a natural integer called *dimension.* A topological cell of dimension d is called d-cell. The structure of the whole space, corresponding to the partition into topological cells, is considered through the *incidence relationships*, relating two “neighbor” cells in the partition. A graph can then be seen as a cell complex composed of 0-cells and 1-cells, respectively the nodes and the edges, so that the incidence relationship of the complex coincides with the usual notion of incidence in graphs. More generally, a cell complex using only two dimensions is an undirected multi- and hyper-graph. There exist many possible formal definitions of cell complexes coming from different fields (algebraic topology [12], digital topology [8], geometric modeling, etc.). We consider here the definition of [20].

Let $\mathcal{L}$ be an arbitrary set of symbols. A *labeled abstract cell complex* $\mathcal{K}$ *with labels in* $\mathcal{L}$ is given by a tuple $\langle \mathrm{C}_\mathcal{K}, \prec_\mathcal{K}, \dim_\mathcal{K}, \mathrm{l}_\mathcal{K} \rangle$ where $\mathrm{C}_\mathcal{K}$ is the set of *abstract topological cells*, $\prec_\mathcal{K}$ is a locally finite[2] strict partial order relation over $\mathrm{C}_\mathcal{K}$, $\dim_\mathcal{K} : \langle \mathrm{C}_\mathcal{K}, \prec_\mathcal{K} \rangle \to \langle \mathbb{N}, < \rangle$ is a strictly monotonic map assigning the dimension to each cell, and $\mathrm{l}_\mathcal{K} : \mathrm{C}_\mathcal{K} \to \mathcal{L}$ is a map assigning a label to each cell. An example of an abstract cell complex is shown on Fig. 1. We denote Acc the set of abstract cell complexes. In the following, we use the term *cell complex* (resp. *cell*) for abstract cell complex (resp. topological cell) since there is no possible ambiguity.

A *cell complex morphism* $h : \mathcal{K} \to \mathcal{K}'$ from a cell complex $\mathcal{K}$ to a cell complex $\mathcal{K}'$ is given by a strictly monotonic map $\mathrm{C}_h : \langle \mathrm{C}_\mathcal{K}, \prec_\mathcal{K} \rangle \to \langle \mathrm{C}_{\mathcal{K}'}, \prec_{\mathcal{K}'} \rangle$ such that $\mathrm{l}_\mathcal{K} = \mathrm{l}_{\mathcal{K}'} \circ \mathrm{C}_h$, and $\dim_\mathcal{K} = \dim_{\mathcal{K}'} \circ \mathrm{C}_h$. A *cell complex inclusion* $i : \mathcal{K} \to \mathcal{K}'$ from a cell complex $\mathcal{K}$ to a cell complex $\mathcal{K}'$ is a cell complex morphism from $\mathcal{K}$ to $\mathcal{K}'$ such that C_h is injective.

We consider the category $\mathbf{AccM}_\mathcal{L}$ of cell complexes and cell complex morphisms between them on one hand, and the sub-category $\mathbf{AccI}_\mathcal{L}$ of cell complexes and cell complex inclusions between them on the other hand, together with the associated inclusion functor $\mathrm{U}_\mathcal{L} : \mathbf{AccI}_\mathcal{L} \to \mathbf{AccM}_\mathcal{L}$. In the following, the subscript $\mathcal{L}$ is omitted, the set of labels is clear from the context and we never consider different sets of labels simultaneously. For the categorical discussion, we use the concepts of categories, functors, natural transformations, pushouts, colimits and comma categories. For formal definitions of these concepts, we refer to [10].

4 Specification of a Triangular Mesh Refinement

Mesh refinement is an approach used in geometrical modeling to generate smooth surfaces from an initial set of control points. Mesh refinement algorithms consist generally in iteratively generating new control points from current ones by applying a set of creation rules. Such procedures are commonly used in numerical resolution schemes and can be specified through graph transformations [14]. In [20], the declarative expression and implementation of these algorithms as a maximal-parallel cell complex rewriting are discussed.

Although mesh algorithms are intuitively described by local graphical schemes, they operate on the whole mesh and then turn out to be *global* and *synchronous*. Let us illustrate this issue by considering the Loop subdivision procedure [11] which is one of the simplest algorithms for refining triangular meshes (cell complexes where all 2-cells respect the incidence given in Fig. 1). It relies on the polyhedral subdivision where each triangle of the original mesh is substituted by four triangles as shown on Fig. 2a. This rule is quite informal. In particular, notice the dashed edges: they are definitively not part of the local transformation

[2] For any elements $x, y \in \mathrm{C}_\mathcal{K}$, the interval $[x, y] = \{\ z \mid x \prec_\mathcal{K} z \prec_\mathcal{K} y\ \}$ is finite.

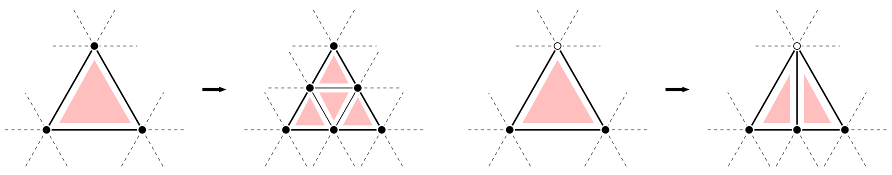

(a) Inner Rule (b) Boundary Rule

Fig. 2. Polyhedral Subdivision Scheme

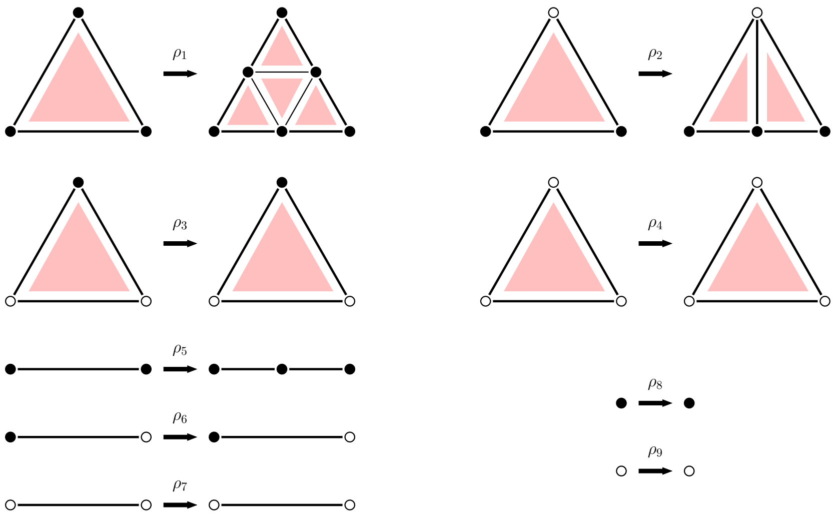
of the triangle but express how the new pattern has to be merged with the hypothetical applications of the same rule on some neighbor triangles. However, this detail —in the sense that the primary role of the rule is to specify the refinement of one triangle— has serious consequences on the algorithm since it forbids to consider the application of the rule on only one triangle and worse it only allows the transformation of the whole mesh. The reason of such a constraint comes from the considered class of cell complexes, *i.e.*, the mesh must remain triangular. This particularity makes mesh refinement an excellent candidate to illustrate our approach. In this section, we first specify a set of transformation rules for polyhedral subdivision, and then we detail how these rules are applied in the context of a global transformation.

4.1 Rules Specification

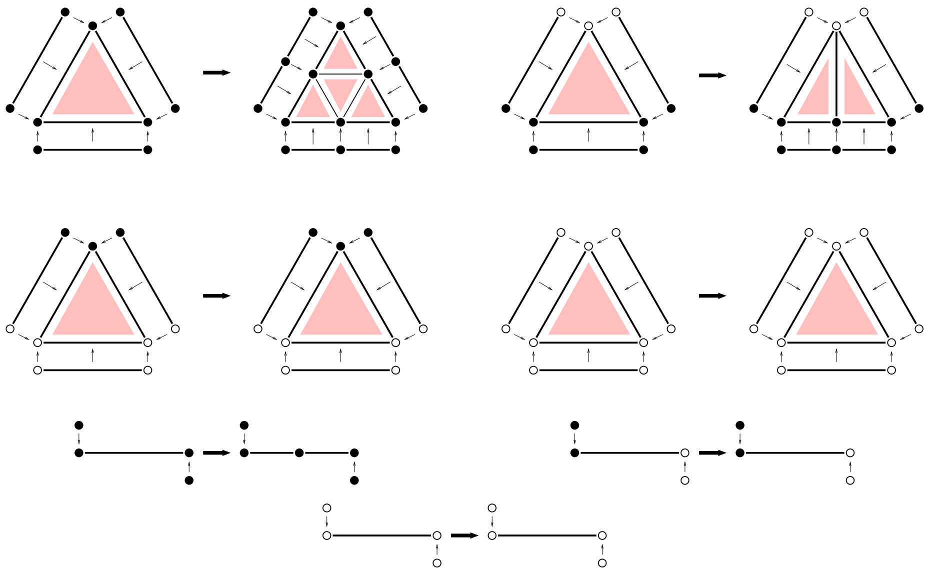
For the sake of illustration, let us consider that the refinement is restricted to a sub-part of the mesh. The region is specified by white or black labeled nodes so that the subdivision only occurs on triangles incident to three black nodes. Solutions exist to stop the natural propagation of the procedure over the whole mesh. Here, we consider an additional rule (see Fig. 2b) which deals with triangles located on the boundary of the region to be refined, *i.e.*, incident to exactly two black nodes. Other triangles are left unchanged.

The use of dashed contexts in Fig 2 is not accurate. Let us design a complete set of rules allowing the refinement of a triangular mesh in one global step. As a starting point, we specify the transformation of any single triangle of the mesh by considering two subdivision rules, ρ_1 and ρ_2 (corresponding to Fig. 2 without dashed context), and two more additional rules, ρ_3 and ρ_4, for unmodified triangles. These rules are shown on top of Fig. 3a.

Obviously, there is a lack of specification since connections between triangles are not taken into account. For instance, let us consider two triangles with only black nodes connected by a common edge as shown on the left of Fig. 4. Two matchings of the l.h.s. of rule ρ_1 are clearly identified. Therefore, the resulting cell complex should be composed of two instances of the r.h.s. of rule ρ_1. However, nothing specifies the way to built it. Forgetting mesh refinement for a moment, many possible constructions are conceivable: leaving the r.h.s. instances isolated,



(a) Local Transformation Rules



(b) Inclusion between Rules

Fig. 3. Polyhedral Subdivision Rules

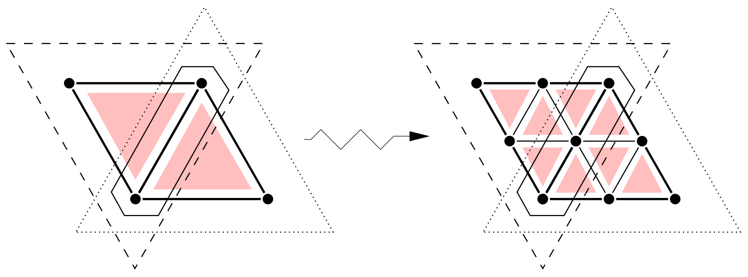


Fig. 4. Overlapping between two triangles

or merging them by a vertex or an edge, or even identifying them in only one instance, and so on and so forth.

In the case of mesh refinement, when two triangles share an edge, the two corresponding edges on the r.h.s. should also be shared on the resulting cell complex. There are two ways to specify such a behavior: one union-based method where a super-rule is designed for pairs of triangles, and another dual intersection-based method where an agreement is given about the transformation of the common edge. Here, we choose the latter method focusing on the evolution of an edge incident to two black nodes. Two things are required: firstly rule ρ_5 of Fig. 3a specifies the subdivision of that edge; secondly for each edge found on the l.h.s. of rule ρ_1, the associated divided edge is identified in the r.h.s. of ρ_1 as shown on Fig. 3b.

With this information at hand, the previous example of two side-by-side triangles is clarified: there are two matchings of ρ_1 and one matching of ρ_5 included in the two former matchings. The result cell complex is the unique cell complex where we can see two instances of the r.h.s. of ρ_1 and one instance of the r.h.s. of ρ_5 with inclusions between them being exactly the ones dictated by the inclusion rule. This can be observed in Fig. 4 where the inclusions between the l.h.s. matchings and the correspondence with the associated r.h.s. inclusions between the corresponding r.h.s. instances are depicted.

Iterating this design process for all possible intersections between rules, we obtain all the rules of Fig. 3a and all the inclusions given in Fig. 3b. Rules ρ_6 and ρ_7 describe the conservation of edges incident to a white node and the conservation of nodes is specified in ρ_8 and ρ_9. The latter are necessary for triangles sharing only one node. After transformation, their r.h.s. must also be connected by a node[3].

[3] As a byproduct of this intersection-based methodology, rules ρ_5-ρ_9 are used to manage isolated edges and nodes, which seems coherent with the refinement process. If these effects are not desired then the union-based methodology is applicable, leading to more transformation rules, namely 32 pair-rules since there are 4 types of triangles and two triangles are either incident by an edge or a node. This allows to stay strictly concerned with triangles.

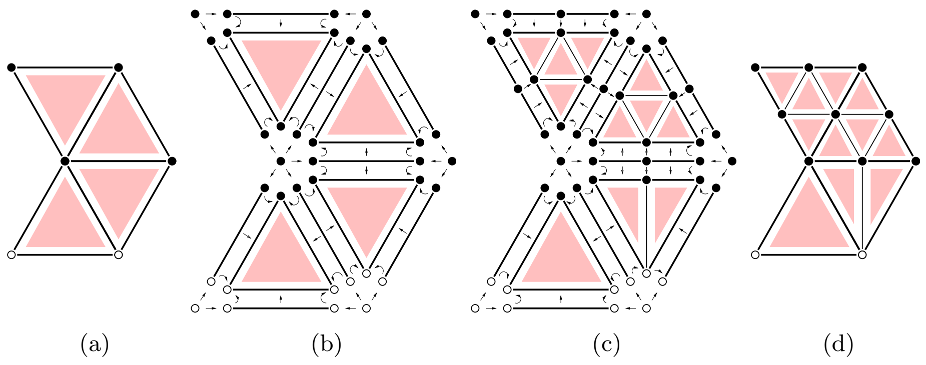

(a) (b) (c) (d)

Fig. 5. Application Steps

As for the rule ρ_5, all inclusions between the l.h.s.'s of these new rules are found and the associated r.h.s.'s are identified. This association of inclusion has to be seen as a set of rules too, that we call *inclusion rules*: for each inclusion of one l.h.s. in another l.h.s., an inclusion rule specifies an associated inclusion between the two corresponding r.h.s. Put in another way, theses inclusion rules express a kind of *mutual agreement* between the rules. When a rule indicates a transformation on a l.h.s., and this l.h.s. includes the l.h.s. of another rule, the former rule must achieve the transformation required by the latter rule, and this is materialized by an inclusion rule indicating where this required transformation can be found. So no mutual exclusion is necessary. This is reminiscent of the notion of sub-production in amalgamation. The comparison with this concept is delayed to Sect. 5.3.

4.2 Rules Application

The construction procedure described above about the transformation of two side-by-side triangles can be generalized to any mesh as follows:

1. *Pattern matching* (Fig. 5a $\Rightarrow$ 5b): the original mesh is split into all the matchings of the local rules l.h.s. together with all the inclusion information between these matchings. Any unmatched part of the mesh is lost.
2. *Local rule application* (Fig. 5b $\Rightarrow$ 5c): each l.h.s. instance is locally replaced by its corresponding r.h.s. The inclusion information is also updated: each l.h.s. inclusion is replaced by its corresponding r.h.s. inclusion.
3. *Reconstruction* (Fig. 5c $\Rightarrow$ 5d): the inclusion information are finally used to merge the different r.h.s. giving rise to the transformed mesh.

Fig. 5 illustrates the computation of a global transformation step on a mesh composed of four triangles. Note that Figs. 5b and 5c only show more explicitly for the four-triangles case the inclusion structures already described in Fig. 4 for the two-triangles case.

This construction of the polyhedral subdivision based on *mutual agreement* has to be compared with the equivalent construction in [20] using *mutual exclusion*. Mutual exclusion prevents the computation of refinement in one global step. The solution of [20] consists of a two-step procedure: a first transformation focuses on the subdivision of all edges and a second transformation splits the obtained hexagons and squares into triangles. Exclusion holds since the l.h.s. of each rule consists of only one element (a 1-cell for the first step, a 2-cell for the second step). Obviously the results of both approaches are the same. However the exclusion based approach suffers from two main drawbacks:

1. A marking is required in the facets subdivision step to identify the nodes generated by the edge subdivision step. For example, in the substitution of an hexagon by four triangles, the three newly created edges surrounding the central triangle are incident to new nodes.
2. The intermediate cell complex contains hexagons and squares and then is not a regular triangular mesh. An implementation of this approach requires the use of a data structure allowing the representation of such a complex. This is beyond the ability of data structures classically used in geometric modeling which strongly rely on the triangular nature of the meshes.

5 Global Transformations

In this section, we begin by formalizing the constructions described in the example using categorical concepts. Then, we compare the double-pushout (DPO) approach and global transformations, showing in particular some correspondence in the particular case where some conservation rules are specified. We also compare global transformations to the Amalgamation Theorem. The similarity is explained, and some differences are identified.

5.1 Categorical Characterization

Here, we want to give a formal specification of the objects and constructions presented in Section 4. A *transformation rule* ρ is a pair of two cell complexes, the l.h.s. and the r.h.s. A set of transformation rules gives rise to a function $R_0 : L_0 \to \mathrm{Acc}$, where $L_0 \subset \mathrm{Acc}$ denotes the set of l.h.s., that maps each l.h.s. to its corresponding r.h.s. An *inclusion rule* between two transformation rules ρ and ρ' associates an inclusion of the l.h.s. of ρ' in the l.h.s. of ρ with an inclusion of the r.h.s. of ρ' in the r.h.s. of ρ. Let us denote L_1 the set of all inclusions between any pair of l.h.s. of L_0, the set of inclusion rules can be interpreted as a function R_1 mapping any element of L_1 to its corresponding inclusion in Acc. A set of inclusion rules is said to be *complete* if the associated function is total. A set of inclusion rules is said to be *consistent* if all the inclusions agree on composition. In that sense, the set of inclusion rule given in Fig. 3b for the mesh refinement example is complete and consistent[4]. It is trivial to see that these two properties induce the determinism of a global transformation since any case that could be encountered is taken into account by a mutual agreement (completeness) and there is no contradiction between these agreements (consistency).

This situation can be nicely summarized in terms of categorical concepts. Considering that L_0 and L_1 form a sub-category $\mathbf{L}$ of the category $\mathbf{AccI}$, R_0 and R_1 form a functor R from $\mathbf{L}$ to $\mathbf{AccI}$. While consistency holds since R, as a functor, respects the morphism composition in $\mathbf{AccI}$, completeness is grabbed when $\mathbf{L}$ is a *full* sub-category where all possible inclusions are considered. This leads to the following definition:

Definition 1 (Global Transformation). *A* global transformation T *is given by a tuple* $\langle \mathbf{L}_T, \mathrm{L}_T, \mathrm{R}_T \rangle$ *where* $\mathbf{L}_T$ *is a full subcategory of* $\mathbf{AccI}$ *corresponding to the l.h.s. with* $\mathrm{L}_T : \mathbf{L}_T \to \mathbf{AccI}$ *its associated inclusion functor and* $\mathrm{R}_T : \mathbf{L}_T \to \mathbf{AccI}$ *is a functor from* $\mathbf{L}_T$ *to* $\mathbf{AccI}$.

We now formalize how a global transformation T is applied on a cell complex $\mathcal{K}$ to build the associated result $T\mathcal{K}$. Following the decomposition presented in Section 4.2, we get:

1. *Pattern matching*: all the matchings of the l.h.s. of T in $\mathcal{K}$ and the way they are included one in each other exactly correspond to the objects and morphisms of the comma category $\mathrm{L}_T/\mathcal{K}$. Objects of $\mathrm{L}_T/\mathcal{K}$ are pairs $\langle l \in \mathbf{L}_T, i : \mathrm{L}_T l \to \mathcal{K} \rangle$ where l is a l.h.s. and i an inclusion of this l.h.s. in $\mathcal{K}$. Morphisms of the comma category are inclusions *between* the matchings, *i.e.*, a morphism from a matching $\langle l, i \rangle$ to a matching $\langle l', i' \rangle$ is an inclusion $j : l \to l'$ in $\mathbf{L}$ such that $i = i' \circ \mathrm{L}_T j$.
2. *Local rule application*: in order to get the r.h.s. corresponding to each matching, we first use the projection functor $\mathrm{Proj}_{\mathrm{L}_T/\mathcal{K}} : \mathrm{L}_T/\mathcal{K} \to \mathbf{L}_T$ defined on matchings as $\mathrm{Proj}_{\mathrm{L}_T/\mathcal{K}} = \langle l, i \rangle \mapsto l$, and on matching inclusions as $\mathrm{Proj}_{\mathrm{L}_T/\mathcal{K}} = j \mapsto j$. The functor R_T then maps each l.h.s. to the corresponding r.h.s., and each l.h.s. inclusion to the corresponding r.h.s. inclusion. So the result of this step is the compound functor $\mathrm{R}_T \circ \mathrm{Proj}_{\mathrm{L}_T/\mathcal{K}}$.
3. *Reconstruction*: all the r.h.s. instances are finally glued together w.r.t. $\mathrm{R}_T \circ \mathrm{Proj}_{\mathrm{L}_T/\mathcal{K}}$. The resulting cell complex $T\mathcal{K}$ could be obtained as the colimit of this functor. However, colimits are only guaranteed[5] in $\mathbf{AccM}$ since the universality may require a non-inclusion morphism to hold. So we use the forgetful functor U to pass from $\mathbf{AccI}$ to $\mathbf{AccM}$.

[4] Only a subset of the inclusion rules is drawn but all other rules can be retrieved using composition of inclusion.

[5] A way to show that all the colimits exist consists in exhibiting an initial object and all pushouts. Here, the initial object is simply the empty cell complex. The pushout of a span of cell complexes can be obtained from the pushout in **Set** of their cells sets by adding the unique possible dimension function and the smallest possible strict partial order relation, which happens to necessarily exist thanks to the dimensions.

A global transformation application is summarized as follows:

Definition 2 (Application of Global Transformation). *Given a cell complex $\mathcal{K}$ and a global transformation T, the result $T\mathcal{K}$ of the application of T on $\mathcal{K}$ is the cell complex $T\mathcal{K} = \mathrm{Colim}(\mathrm{U} \circ \mathrm{R}_T \circ \mathrm{Proj}_{\mathrm{L}_T/\mathcal{K}})$.*

Note that since we consider all the inclusions between the l.h.s. graphs, we also consider their automorphisms, *i.e.*, their symmetries. This means that, in our previous examples, each matching is *really matched* as many times as it has symmetries. The final result remains correct because all the results of these symmetric matchings are glued together appropriately due to the functorial nature of L_T and R_T. Considering these automorphisms is in fact meaningful as they allow to prevent incoherent specifications that intuitively lead to symmetry breaking. Unfortunately, the size of this paper does not allow us to enter into a more detailed discussion about these features.

5.2 Double-Pushouts and Global Transformations

One of the consequences of our construction of the transformation result is that all parts of the original cell complex which are not matched are not conserved by any means. This is why we do not consider any morphism relating l.h.s. and r.h.s. If something has to be kept, this must be specified explicitly as we did for the not-refined triangles in Section 4. This is in plain contrast with the DPO approach where the default behavior is to conserve unmatched parts of the graph and deletion has to be specified explicitly.

However, if a global transformation states that some patterns have to be conserved, then it is possible to see the elements of the DPO occurring in a different light. This relation appears because conservation rules have identical l.h.s. and r.h.s. So despite the fact that all considered inclusions are either strictly between l.h.s. or strictly between r.h.s., these conservation rule allows to think inclusions linking both l.h.s. and r.h.s. worlds. To express this a bit more visually, let us consider a derivation $G \Rightarrow G'$ via a production $p = L \leftarrow K \rightarrow R$ based on an inclusion $i : L \rightarrow G$ and try to lay it out in a global transformation application way. The elements of the derivation are given in the following diagram.

$$\begin{tikzcd}
L \arrow[d, "i"'] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "k"'] & R \arrow[d, "o"'] \\
G & G^- \arrow[l, "l'"'] \arrow[r, "r'"] & G'
\end{tikzcd}$$

In the global transformation framework, G^- is a compound of many conservation rules, K is a compound of many small conservation rules included in both the conserved and the modified parts, and L and R are all the l.h.s. and r.h.s. of the transformation rules. The result G' is obtained as the colimit of K, G^- and R, which is precisely a pushout. This gives the following layout:

rules

$$\begin{tikzcd}
K \arrow[r, mapsto] \arrow[d, bend right, "k"'] \arrow[dd, bend right=60, "l"'] & K \arrow[d, bend left, "k"] \arrow[dd, bend left=60, "r"] \\
G^- \arrow[r, mapsto] & G^- \\
L \arrow[r, mapsto] & R
\end{tikzcd}$$

matchings

$$\begin{tikzcd}
K \arrow[d, bend right, "k"'] \arrow[dd, bend right=60, "l"'] \arrow[dr, "i \circ l"] & \\
G^- \arrow[r, "l'"] & G \\
L \arrow[ur, "i"] &
\end{tikzcd}$$

r.h.s. and colimit

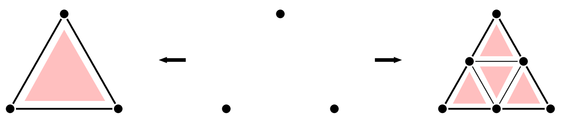
$$\begin{tikzcd}
K \arrow[d, bend right, "k"'] \arrow[dd, bend right=60, "r"'] \arrow[dr, "o \circ r"] & \\
G^- \arrow[r, "r'"] & G' \\
R \arrow[ur, "o"] &
\end{tikzcd}$$

Notice that no notion of pushout complement is required in our construction. The existence of a pushout complement for a production $L \leftarrow K \rightarrow R$ based on a morphism $m : L \rightarrow \mathcal{K}$ imposes that the parts of L which are not in K should not be incident to any cell outside of m for a derivation to be feasible. Contrary to DPOs, all matchings are necessarily applicable in our setting. This gives more flexibility in the design of transformation rules and allows to work exclusively with inclusions. All of this are consequences of the fact that we are in the restricted case where nothing from an input cell complex is considered as conserved.

5.3 Amalgamation and Global Transformations

There is a strong relation between the proposed formalism and the amalgamation of productions available in the DPO approach. Indeed, both consider rules and a notion of inclusions between rules. Moreover, amalgamation of two productions via a sub-production is a pushout of productions and it is well-known that a colimit can be viewed as being mainly a compound of pushouts. If we consider all the matchings of a set of productions and if we have enough sub-productions to handle all possible overlaps, it is clearly possible to synchronize all the productions into one production that operates in a single step as required.

However there are some differences. We have already discussed the default behavior on the unmatched part for both approaches: amalgamation conserves it while global transformation drops it. Assuming that there is no unmatched part, a strong difference remains because of applicability. Indeed, in the DPO approach the amalgamation theorem states that anytime an amalgamation of productions can be applied, each production can be applied individually, with the restriction imposed by the existence of a pushout complement explained earlier. Let us consider the following production that is a straightforward translation of rule ρ_1 of Fig 3a:



Since the three 1-cells in L are not part of K, the rule can only be used on triangles whose edges are incident to no other 0-cells nor 2-cells, which is not the common case in a triangular mesh. Indeed, in mesh refinement, every cells (except the 0-cells) have to be transformed and the main part of them are incident to each other. This makes impossible in general the design of productions isolating the cells to be transformed, except by considering the whole black region in one global production. As a consequence, there is no simple way to use amalgamation alone to build a single global production from many local ones.

Again, an important property of our approach is that there is no applicability condition over the cell complex to transform. The only constraint comes from the completeness and the consistency of the set of inclusion rules which ensure to get a well defined and unique result.

6 Conclusion and Future Work

In this article, we have presented the Global Graph Transformations, an original categorical framework to deal with deterministic graph transformation with a maximal application strategy. Global transformations are based on the notion of *mutual agreement* (as opposed to *mutual exclusion*) which allows overlapping matchings to agree on the transformation of the shared part. Analogously to amalgamation of productions, mutual agreement is realized by considering additional rules specifying the behavior of the intersection parts. Global transformations are useful for expressing the global rewriting of a graph where no interface is explicitly conserved between the l.h.s. and the r.h.s. Taking a point of view different from substitution processes, a global transformation is a way to construct a set of constraints: local transformation constraints imply that for each l.h.s. matching the corresponding r.h.s. has to appear in the result; inclusion constraints state how r.h.s. have to be glued from l.h.s. inclusions. The resulting graph corresponds to the solution of these constraints.

In this paper, the formalization of global transformations relies on the cell complexes category. We made this choice for bridging with other related works. Firstly, our running example about mesh refinement is definitively more clear without any encoding in usual graphs. Secondly, this approach is in line with the MGS programming language based on the rewriting of topological collections [20]. In this context, labeled cell complexes are reminiscent of the MGS topological collections and global transformations correspond to a new rule application strategy. We plan to integrate this strategy in the current implementation of the language. Finally, *causal graphs dynamics* proposed in [1] have been recently extended to *combinatorial manifolds* [2], a specific class of cell complexes. Similarly to these works, we are interested in understanding the notion of locality and causality in global transformations and to relate these notions to a kind of topological continuity of the transformation. In this context, using cell complexes seems more natural since they have been the subject of many developments in algebraic and digital topology.

From a pure algebraic point of view, the use of cell complexes is a detail. For example, the global transformation framework can be moved seamlessly from cell complexes to graphs. One can then ask what kind of general theory supports global transformations in the same way that the essential properties of graphs regarding DPOs have been identified leading to several axiomatic frameworks (*e.g.*, adhesive [9] and High-Level Replacement [13] categories). There also seems to be interesting possibilities in integrating the notion of nested application condition in the context of global transformations too, to specify for example that a sub-rule is only considered as the intersection of some super-rules and not others. All these issues are part of future work.

Acknowledgments

This research is partially supported by the French ANR grants "SynBioTIC" ANR-10-BLAN-0307 and "TARMAC" ANR-12-BS02-0007, and by the University Paris-Est Créteil.

References

1. Arrighi, P., Dowek, G.: Causal graph dynamics. CoRR abs/1202.1098 (2012)
2. Arrighi, P., Martiel, S., Wang, Z.: Causal dynamics of discrete surfaces. In: DCM 2013, Buenos Aires, Argentina, 26 August 2013. pp. 30–40 (2014)
3. Boehm, P., Fonio, H.R., Habel, A.: Amalgamation of graph transformations: A synchronization mechanism. Journal of Computer and System Sciences 34(2–3), 377 – 408 (1987)
4. Ehrig, H., Rosen, B.K.: Parallelism and concurrency of graph manipulations. Theoretical Computer Science 11(3), 247 – 275 (1980)
5. Giavitto, J.L., Michel, O., Cohen, J., Spicher, A.: Computation in space and space in computation. In: UPP'04. LNCS, vol. 3566, pp. 137–152. Spinger, Le Mont Saint-Michel (Sep 2005)
6. Golas, U., Habel, A., Ehrig, H.: Multi-amalgamation of rules with application conditions in m-adhesive categories. Mathematical Structures in Computer Science 24(4), 1 – 68 (2014)
7. Janssens, D., Rozenberg, G., Verraedt, R.: On sequential and parallel node-rewriting graph grammars. Computer Graphics and Image Processing 18(3), 279–304 (1982)
8. Kovalevsky, V.A.: Geometry of Locally Finite Spaces. Editing House Dr. Baerbel Kovalevsky (2008)
9. Lack, S., Sobociński, P.: Adhesive categories. In: FOSSACS 2004, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings. pp. 273–288. Springer (2004)
10. Lane, S.M.: Categories for the working mathematician, vol. 5. Springer Science & Business Media (1978)
11. Loop, T.L.: Smooth subdivision surfaces based on triangle. Master's thesis, University of Utah (Aug 1987)
12. Munkres, J.: Elements of Algebraic Topology. Addison-Wesley (1984)
13. Padberg, J.: Survey of high-level replacement systems. Tech. rep. (1993)
14. Paszyńska, A., Grabska, E., Paszyński, M.: A graph grammar model of the hp adaptive three dimensional finite element method. part i. Fundamenta Informaticae 114(2), 149–182 (2012)

15. Paun, G.: Computing with membranes. Journal of Computer and System Sciences 1(61), 108–143 (2000)
16. Prusinkiewicz, P., Lindenmayer, A., Hanan, J.S., et al.: The Algorithmic Beauty of Plants. Springer-Verlag (1990)
17. Raoult, J.C., Voisin, F.: Set-theoretic graph rewriting. Tech. Rep. RR-1665, INRIA (April 1992)
18. Rozenberg, G., Ehrig, H.: Handbook of graph grammars and computing by graph transformation, vol. 1. World scientific Singapore (1999)
19. Rozenberg, G., Salomaa, A.: Lindenmayer Systems. Springer, Berlin (1992)
20. Spicher, A., Michel, O., Giavitto, J.L.: Declarative mesh subdivision using topological rewriting in MGS. In: ICGT 2010. LNCS, vol. 6372, pp. 298–313 (2010)