ISSN 2464-4617 (print)
ISSN 2464-4625 (online)
Computer Science Research Notes - CSRN 3401
http://www.wscg.eu
WSCG 2024 Proceedings

Reevaluation in Rule-Based Graph Transformation Modeling Systems

Maxime Gaide[1] David Marcheix[2] Agnès Arnould[3]
Xavier Skapin[3] Hakim Belhaouari[3] Stéphane Jean[2]
[1]ISAE-ENSMA Poitiers, Université de Poitiers, LIAS, Poitiers, France
[2]Université de Poitiers, ISAE-ENSMA Poitiers, LIAS, Poitiers, France
[3]Université de Poitiers, Univ. Limoges, CNRS, XLIM, Poitiers, France
firstname.lastname@{[1,2]ensma, [3]univ-poitiers}.fr

Abstract

In this paper, we widen the naming problem studies to the rule-based graph 3D transformation modeling systems. We propose a persistent naming method taking advantage of the generalized maps' and graph transformation rules' formalization of simple operations. It enables a unique and homogeneous characterisation of entities in all dimensions. Most existing methods require tracking numerous topological entities and consider the persistent naming problem only from the parameters' modifications of a parametric specification standpoint. With our solution, not only the naming problem is tackled within the usual framework of parameters edition, but we also take the specification edition into account (addition, deletion and displacement of operations). Moreover, our solution makes use of directed acyclic graphs to represent the histories of topological entities and to track only the entities used in the parametric specification and the ones they originate from.

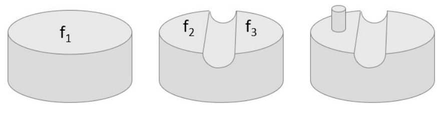
Keywords

Topology-based modeling; Graph transformation rules; Persistent Naming; Reevaluation; Generalized maps;

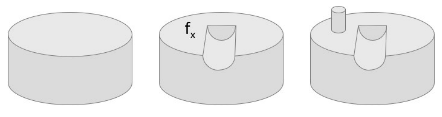
1 INTRODUCTION

The ability to generate multiple variants of an object during a construction process is becoming increasingly frequent in many application areas. Most of the time, tools and operations used to create those variants are dedicated to specific fields and the construction process is often both tedious and time-consuming. For example, in the field of Archaeology, remaining data found on the working field often represent only vestiges of ancients buildings. By means of 3D reproduction, archaeologists painstakingly develop a number of hypotheses they expect to test while being able to quickly model and visualize them [QB15]. CAD uses parametric history-based systems; such systems can be thought as dual structures with, on the one hand, the geometric model corresponding to the modelled object and, on the other hand, the successive operations (and their parameters) recorded during the construction process. This process can then be reevaluated after some slight modifications upon the operations, without starting all over again from the beginning. Nevertheless, creating complex objects always requires a substantial amount of time. Modeling buildings is also of interests for architects. Grammar-based procedural methods are commonly used to generate variations of constructions [HMV09; Mül+06]. But those grammars are based on a specific corpus of information which is difficult to transpose to other case studies.

In this paper, we propose to use a rule-based graph 3D transformation formalism, and more specifically the Jerboa software [Bel+14], to make the development of dedicated modelers for specific applications easier. Rule-based languages for modeling are commonly used in a number of fields such as plant growth with L-systems [Lin74; BTG15], wood's internal structure [Ter+09], or virtual cities 3D models [ESR; Bei+10]. Contrary to other approaches, Jerboa is independent from any specific application field and does not require *ad hoc* operations to be manually coded. Simple operations are formally defined as rules within the Jerboa interface, thereby facilitating their rapid development. Furthermore, it guarantees the topological consistency of the underlying geometric model, regardless of the applied operations [Arn+22]. Jerboa is based on the generalized maps (or G-maps) topological model [Lie91; DL14]. This model represents a specific class of labelled graph and allows the homogeneous modeling of quasi-manifolds in any dimension. Number of applications already make use of Jerboa and/or G-maps in fields such as plant growth



(a) Initial evaluation



(b) Reevaluation

Figure 1: Parametric specification

[BTG15], architecture [Hor+09; ALS15], geology [HH00] or physics-based modeling [Ben+17]. Despite its advantages, Jerboa does not provide the mechanisms required to quickly reevaluate variations from a base model. Conversely, history-based parametric systems take advantage of the construction process recording to make reevaluations as fast and accurate as possible. Thus, our objective is to extend the capabilities of Jerboa by incorporating the mechanisms inherent in parametric systems. Any reevaluation of the parametric specification entails modifying the parameters of the operations. Those parameters are either geometric (such as the length of a groove) or references topological entities (vertices, edges, faces and so on) defined at an earlier stage of the modeling process. Modifying some operation parameters requires ensuring that the subsequent operations are still valid, even if their own parameters have been updated. This issue, known as *persistent naming*, is illustrated in Fig. 1. The initial parametric specification consists of three constructive operations (Fig. 1a): `1-Create-Cylinder(geo_param1); 2-RoundedGroove(f1, geo_param2); 3-CylindricalProtrusion(f2, geo_param3)`. When identifiers or pointers (*i.e.* concrete names) to the topological parameters of a geometric model (*e.g.* the identifier of the face `f2` as a parameter of the `CylindricalProtrusion` operation) are used as topological parameters, the issue of the persistence of these references at reevaluation comes up. For example, in Fig. 1b, the rounded groove's length is reduced. The face `f1` is not split anymore, unlike during the initial evaluation. Thus, neither face `f2` nor face `f3` are created: identifiers and pointers to the entities are obviously different and, therefore, the cylindrical protrusion can no longer be re-applied onto `f2`. Hence the necessity to use persistent identifiers as operations parameters, which make possible to unambiguously characterize entities and find their match at reevaluation. In Fig. 1, using a persistent name to characterize `f2` during the initial evaluation allows matching it with the face `fx` at reevaluation. Although persistent naming has been studied for decades in the CAD's field [Wu+01; MH05; Mar06; Bab10; Xue+16; FH18; Saf+20; CBS23; DZ24], to our knowledge two preliminary approaches have attempted to use graph transformation rules to tackle this issue [Car+19; Gai+23b]. In [Car+19], the authors propose to use *History Records* (HRs) to represent the history of any topological entity designated in parametric specification and *Matching Trees* (MT) to match this entity during reevaluation. This is an interesting initial theoretical approach based on graph transformation, but the history represented in HR is limited. In particular, no distinction is made between the entities at the origin of the designated entity and the evolution of the designated entity itself. Furthermore, some elements are omitted in HRs (such as the history of entities at the origin of the designated entity), which can lead to mismatches during reevaluation. In the concise poster paper [Gai+23b], the authors propose a full persistent naming mechanism based on graph transformation rules. They also propose to complete the histories of topological entities by taking their origins into account and integrate them in a reevaluation mechanism. In this paper, we base our work on [Gai+23b] to integrate the complete history of topological entities. Our contribution is twofold. First, we widen the naming problem studies to the rule-based graph transformation modeling systems. Second, we integrate the mechanisms of reevaluation for parametric systems into Jerboa.

We propose a persistent naming method taking advantage of the rule-based formalization of operations and their ability to precisely describe the history of topological entities, such that these entities are uniquely and homogeneously characterized for all dimensions. Most existing methods require tracking numerous topological entities and consider the persistent naming problem only through the prism of parameters modifications from a parametric specification standpoint[CH95; Wu+01]. Our solution tracks only the entities used in the parametric specification and the ones they originate from. Moreover, not only the naming problem is tackled within the usual framework of parameter edition, but we also take the specification edition (*i.e.* adding, deleting and reordering of operations) into account.

In section 2, we present the necessary concepts to carry out persistent naming mechanisms within the framework making use of graph transformation rules. We focus on G-maps, Jerboa's rules, and on how to automatically detect topological changes (creation, deletion, split, merging, modification) that may occur upon any rule application. Section 3 is dedicated to the data structures used by parametric specifications and persistent naming. Section 4 describes how a parametric specification is evaluated or reevaluated through directed acyclic graphs which track the evolutions of topological entities and the ones they originate from. Section 5 presents the matching process between evaluated and reevaluated entities. We conclude in section 6 and present the main directions of our future works.

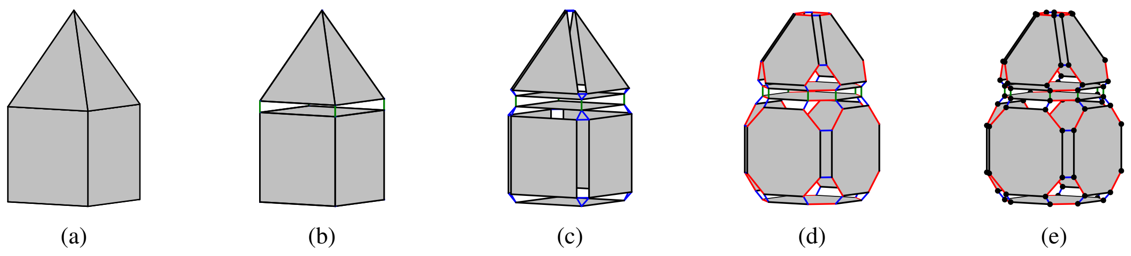


Figure 2: Cell decomposition of a geometric 3D object

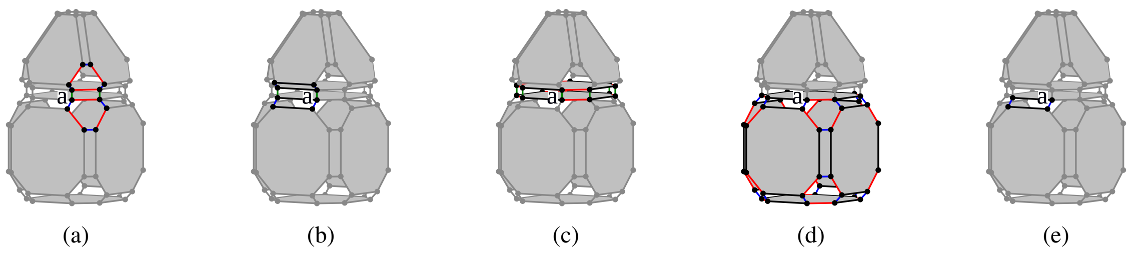


Figure 3: Orbit decomposition of a geometric 3D object

2 MAIN CONCEPTS

In this section, we present the generalized maps, graph transformation rules and their subsequent concepts which are necessary to the understanding of our contribution.

2.1 Generalized maps

Generalized maps (or G-maps) [Lie91; DL14] allow the representation of manifold geometric objects (with or without boundaries), based on a cellular n-dimensional topological structure. The representation of an object as a G-map comes intuitively from its decomposition into topological cells (vertices, edges, faces, volumes, and so on). For example, the 3D topological object (Fig. 2a) can be decomposed into two volumes (Fig. 2b): a cube and a pyramid. These volumes are *linked* along their common faces with a 3-link, drawn in green. The index "3" means that the link connects two 3-dimensional (possibly a single one) volumes. In the same way, volumes are split into faces connected with blue 2-links (Fig. 2c). Then, faces are split into edges connected with red 1-links (Fig. 2d). Lastly, edges themselves are split into vertices with black 0-links (Fig. 2e) to produce the 3-G-map describing the objects shown in Fig. 2a. A G-map is therefore a graph, the nodes (named *darts*) are vertices of edges of faces of volumes and the arcs are i-links. By convention, border darts have 3 loops which are not represented to make the figures easier to read.

G-maps have conditions guaranteeing objects consistency, for example, two faces are always linked along an edge.

Topological cells are not explicitly represented in G-maps but only implicitly defined as subgraphs named *orbits*. They can be computed using graph traversals defined by an originating dart and by a given set of link labels. For example, the 0-cell (or the object's vertex) incident to some dart a (Fig. 3a) is the subgraph which contains a and all darts reachable from a, using links labelled by 1, 2 or 3 and the links themselves. This subgraph is denoted by $G\langle 1,2,3\rangle(a)$ where $\langle 1,2,3\rangle$ is the *type* of the orbit and models a vertex. The 1-cell (or edge) incident to a (Fig. 3b) is the subgraph $G\langle 0,2,3\rangle(a)$ which contains a and all the reachable darts using links labelled by 0, 2 or 3 and the corresponding links. The 2-cell (or face) incident to a (Fig. 3c) is the orbit $G\langle 0,1,3\rangle(a)$. The 3-cell (or volume) incident to a (Fig. 3d) is the orbit $G\langle 0,1,2\rangle(a)$. Note that orbits are more general than cells. For example, the volume edge $G\langle 0,2\rangle(a)$ (Fig. 3e) is the $\langle 0,2\rangle$-orbit incident to a.

2.2 Graph transformation rules

Jerboas's [Bel+14; Arn+22] graph transformation rules allow the formalization of operations over G-maps. In a few words, a *rule* $r : L \longrightarrow R$ and a *match* $m : L \rightarrow G$ to a G-map G, describe the transformation $G \longrightarrow^{r,m} H$ from G to H. The match m allows the replacement of a subgraph of G described by the left-hand side of the rule L with another one described by the right-hand side R, in order to produce H.

Informally, in the extrusion rule (Fig. 4), the left-hand side is made of only one node n_1 (orange) labelled with the $\langle 0,1\rangle$ face type: this way, it can match any face. For the match $m : n1 \mapsto 6$ from L to G (Fig. 5a), the node n_1 matches the whole face $G\langle 0,1\rangle(6)$. On the right side, the node n_1 label remains $\langle 0,1\rangle$. This means that, after applying the rule, the matched face $\langle 0,1\rangle(6)$ has been preserved, in other words $G\langle 0,1\rangle(6) = H\langle 0,1\rangle(6)$ (Fig. 5). In R, the new node n_2 (blue) creates, a copy of the matched face in H. However, n_2's label is $\langle 0,\_\rangle$ meaning that 0-links are preserved and 1-links are deleted. Therefore, n_2 creates face edges $\langle 0\rangle$ from

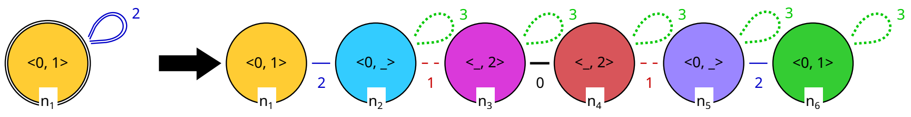
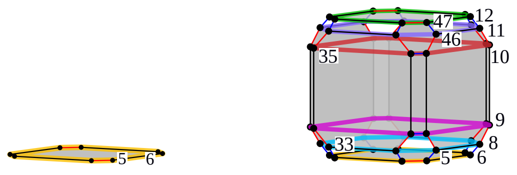


Figure 4: Rule extruding a face into a volume



(a) Graph G: matched by n_1 (b) Graph H : a cube

Figure 5: Extrusion of a face into a cube (rule Fig. 4)

the edges of the matched face. In a similar way, n_3 (pink) creates another copy of the matched face. Because it is labelled $\langle\_,2\rangle$, 0-links are deleted, 1-links relabelled with 2, n_3 creates edge vertices $\langle 2\rangle$ from the matched face's vertices. Finally, the nodes n_4, n_5, n_6 create the same orbits than nodes n_3, n_2 and n_1, respectively. The nodes' labels, called *implicit arcs*, match the highlighted links (Fig. 5b).

The arc between n_1 and n_2, called *explicit arc*, is 2-labelled in the extrusion rule (Fig.4) and 2-links one-to-one the preserved orange darts and the created blue darts (Fig. 5). Similarly, the explicit arc between n_2 and n_3, 1-links one-to-one the blue and pink darts.

The node n_1 (Fig. 4) is a *preserved node* because it belongs to both the left and right-hand sides of the rule. Nodes n_2 to n_6 are *created nodes* because they belong only to the right-hand side. *Deleted nodes* belong only to the left-hand side. Note that the extrusion rule does not have any deleted node.

The *orbit* notion is extended to patterns of rules.

Jerboa's rules provide syntactic properties which guarantee the preservation of the consistency of G-maps [Arn+22].

2.3 Orbit tracking

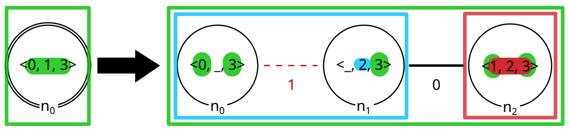
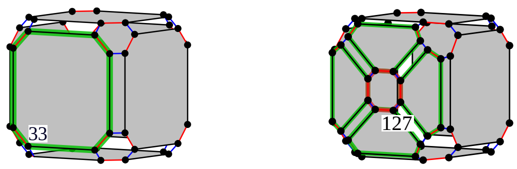


Figure 6: Rule triangulating a face



(a) Graph H: front face matched by n_0 (b) Graph I: triangulated front face

Figure 7: Triangulation of a face (rule Fig. 6)

Rules contain the necessary information characterizing the topological changes affecting an orbit throughout an application [Gai+23a]. Thus, the tracking of an orbit is automatically made without any addition other than the rules' syntactic analysis.

Consider the example of the triangulation (Fig. 6) and its application on H's dart 33 (Fig. 7a). The left-hand side of the rule matches the whole green square face $H\langle 0,1,3\rangle(33)$ while its right-hand side splits it into four green triangles in graph I (Fig. 7b).

Similarly to G-maps' orbits, in the left-hand side of the rule, the $\langle 0,1,3\rangle$-orbit incident to n_0 is the orbit containing the node reachable through arcs labelled in $\langle 0,1,3\rangle$ and those same arcs. This left-hand side orbit is written $L\langle 0,1,3\rangle(n_0)$. Therefore, $L\langle 0,1,3\rangle(n_0)$ is the orbit matching the green face.

In green (Fig. 6), an 1-arc connects the nodes n_0 and n_1 and a 0-arc connects n_1 and n_2, thus forming the $\langle 0,1,3\rangle$-orbit (face) incident to n_0. This orbit matches the four faces resulting from the application of the rule (Fig. 7b). The syntactic analysis of the rule allows us to deduce that the face orbit is *split* along its implicit 1-arcs because the second implicit 1-arc of n_0 in the left-hand side is relabelled outside of the face $\langle 0,1,3\rangle$-orbit type in the right-hand side. Consequently, the matched face of graph H is split along its vertices' 1-links and into four faces in graph I. Similarly, a rule *merges* two or more $\langle o\rangle$-orbits when a k-th implicit arc is relabelled from i to j, where $i \notin \langle o\rangle$ and $j \in \langle o\rangle$, while there was no such k-th implicit arc in any node of the left-hand side.

In red, the vertex orbit $R\langle 1,2,3\rangle(n_2)$ incident to n_2, matches the vertex $I\langle 1,2,3\rangle(127)$. Since $R\langle 1,2,3\rangle(n_2)$ only contains n_2 which has been created, thus the orbit itself is *created* and the application of the rule creates the vertex $I\langle 1,2,3\rangle(127)$.

In blue, the face vertex $\langle 1,2\rangle$-orbit incident to the preserved node n_0 matches the face vertices of the green face such as the vertex orbit $H\langle 1,2\rangle(33)$. The node n_1 is added to the orbit through a 1-arc, $R\langle 1,2\rangle(n_0)$, thus modifying it. Consequently, the matched face's vertices of graph H are modified in graph I.

Finally, the face edge $\langle 0\rangle$-orbit incident to the preserved node n_0 matches the green face's edges in H. Since no node is added nor deleted from the orbit nor any arc is relabelled, thus the orbit is *not modified*.

Through this analysis, topological changes can be logged within bulletin boards which are automatically computed without requiring any other intervention.

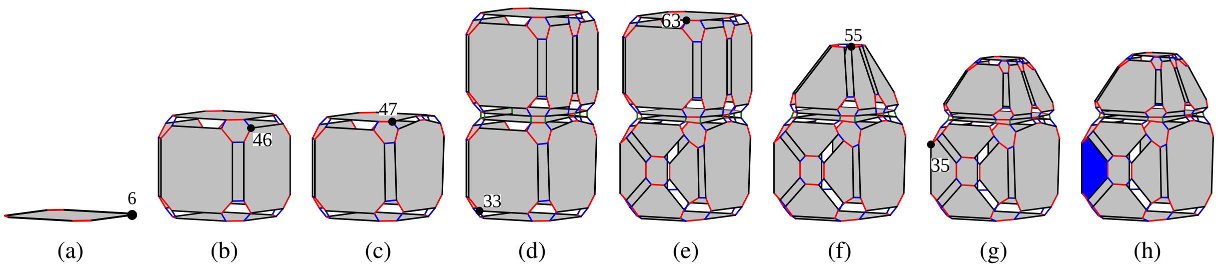


Figure 8: Evaluation (a) 1-square(pos); (b) 2-extrude(PN_1,vec); (c) 3-insert(PN_2); (d) 4-extrude(PN_3,vec); (e) 5-triangulate(PN_4); (f) 6-collapse(PN_5); (g) 7-chamfer(PN_6); (h) 8-colour(PN_7)

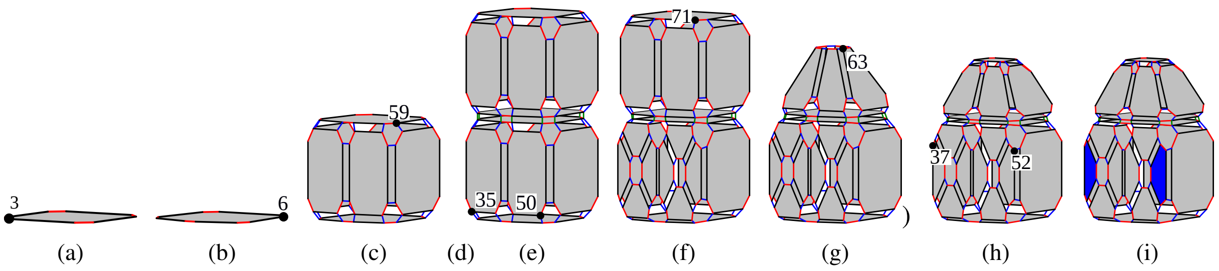


Figure 9: Reevaluation (a) 1-square(pos); (b) ADD1-insert(3); (c) 2-extrude(PN_1,vec); (d) DELETE 3-insert(PN_2); (e) 4-extrude(PN_3,vec); (f) 5-triangulate(PN_4); (g) 6-collapse(PN_5); (h) 7-chamfer(PN_6); (i) 8-colour(PN_7)

3 PARAMETRIC SPECIFICATION

During an object's construction, a *parametric specification* records both the rules representing the applied operations and their parameters (both topological and geometric) in order to describe the modeling process. Editing a parametric specification means that rules may be added, deleted, moved, and their parameters can be changed. With such changes, the topological parameters may have different concrete names, be deleted and so on. As a result, using the parameters' concrete names eventually lead to unexpected results (at best) or failure at runtime. To this end, persistent names are required to robustly identify topological parameters across an object reevaluation.

This section will follow the evaluation (Fig. 8) of a modeling process as an example in order to illustrate the creation of a parametric specification and its persistent names. Its reevaluation (Fig. 9) illustrates a variant modeling process where a rule is added between applications 1 and 2, and where the application 3 is deleted.

In this paper, all the figures were generated using Jerboa and the software overlay developed to implement the concepts presented in this section and the following ones. An overview of the results can be found here: `http://xlim-sic.labo.univ-poitiers.fr/jerboa/doc/model-reevaluation-based-on-graph-transformation-rules/`

3.1 Persistent name

Since rules use darts as topological parameters, it follows that each persistent name must represent a unique dart. As it happens, rules make it possible to determine unambiguously, which node filters or creates any dart.

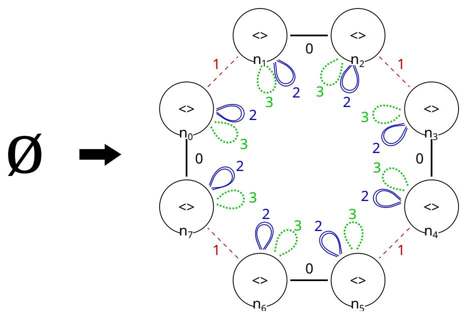


Figure 10: Rule creating a square face

For example, the square rule (Fig. 10) creates 8 darts *ex nihilo* (Fig. 8a), one per node. Thus, dart 3 is created during the first application, by the square's node n_3. The history of dart 3 is noted $[1n_3]$. Similarly, since dart 5 (resp.) 6 is created by node n_5 (resp. n_6), its history is $[1n_5]$ (resp. $[1n_6]$).

The second application (Fig. 8b and for more details Fig. 5) of the face extrusion rule (Fig. 4), creates darts from the matched darts. This rule is applied on the square face created during the first application. Since this face contains 8 darts, each of them is matched by node $n1$ of the extrusion rule (Fig 5). As a consequence, nodes n_2 to n_6 create darts copied from the 8 matched darts. Therefore, dart 6's history is now $[1n_6;2n_1]$, its copy 8's history is $[1n_6;2n_2]$, its copy 9's history is $[1n_6;2n_3]$, its copy 10's history is $[1n_6;2n_4]$ and so on. Similarly, histories of darts 46 and 47 are $[1n_5;2n_5]$ and $[1n_5;2n_6]$, respectively.

For the same reason, histories of darts 33 and 35 are $[1n_3;2n_2]$ and $[1n_3;2n_4]$, respectively. It follows that the cube's 48 darts all have a different history.

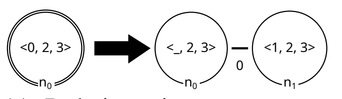


Figure 11: Rule inserting a vertex on an edge

The third application (Fig. 8c) uses the rule inserting a vertex on an edge. Both darts 46 and 47 are matched by node n_0 of the insertion rule. Their respective histories are $[1n_5;2n_5;3n_0]$ and $[1n_5;2n_6;3n_0]$ now. Conversely, darts 33 and 35 have not been not matched during the vertex insertion and their respective histories remain $[1n_3;2n_2]$ and $[1n_3;2n_4]$. The object's 52 darts, again, have each a different history.

In short, the history of any dart of the topological model is entirely defined by the rules applied during the evaluation process. This process guarantees to associate each dart with a unique history; hence, we use this history as the persistent name of the dart. Therefore, $PN_1 = [1n_6]$, $PN_2 = [1n_5;2n_5]$, $PN_3 = [1n_5;2n_6;3n_0]$ and so on.

3.2 Parametric Specification syntax

A number of fields are used to describe an application within a parametric specification, namely an application number, a rule application with the topological parameters and the geometric parameters.

Listing 1: Initial parametric specification

```
1-square(pos)
2-extrude(PN1=[1n6], vec)
3-insert(PN2=[1n5;2n5])
4-extrude(PN3=[1n5;2n6;3n0], vec)
5-triangulate(PN4=[1n3;2n2])
6-collapse(PN5=[1n4;2n6;4n6])
7-chamfer(PN6=[1n5;2n6;3n0;4n4;6n2])
8-colour(PN7=[1n3;2n4;5n0])
```

The parametric specification above represents the modeling process of an initial evaluation (Fig. 8) where each persistent name uses the history of the dart's number displayed in the previous construction step (dart 6 for PN_1, 46 for PN_2, 47 for PN_3, 33 for PN_4 and so on.).

Furthermore, a set of tags are used to describe an application whenever it is either added (ADD), deleted (DELETE) or moved (MOVE) at reevaluation. Thus, the initial parametric specification, once edited as shown in List 2 produces the reevaluation process shown in Fig. 9.

Listing 2: Edited parametric specification

```
1-square(pos)
ADD1-insert(3)
2-extrude(PN1=[1n6], vec)
DELETE 3-insert(PN2=[1n5;2n4])
...
8-colour(PN7=[1n3;2n4;5n0])
```

4 EVALUATION

Although a persistent name represents the history of a dart, an orbit is subject to topological changes and, thus, requires the construction of its own history in order to be accurately matched at reevaluation. Once the initial evaluation (Fig. 8) is done and its parametric specification (1) has been built, an evaluation's Directed Acyclic Graph (or DAG) must be issued for each persistent name before any parametric specification can be reevaluated. An evaluation DAG traces the history of each topological parameter back to the first created orbits it originates from, thus allowing the matching of the corresponding topological parameter at reevaluation time.

4.1 Evaluation DAG

An evaluation DAG is built parsing the applications and nodes of a persistent name from end to start. It is sorted by levels representing the different applications inside a history. Each level is made of an orbit level and an event level. The orbit level contains a node's name and some orbits. The event level contains a rule's application number and some events.

For example, let us consider $PN_3 = [1n_5;2n_6;3n_0]$ (Fig. 8d). PN_3 represents the topological parameter upon which the face extrusion rule is applied. This rule is filtered by hook n_1, matching the face $\langle 0,1\rangle(47)$ (Fig. 8c). The matched orbit's history is built from its dart's history (*i.e.* its persistent name). The DAG is built bottom-up by a backward traversal through the persistent name. Since PN_3 is made of 3 parts, its evaluation DAG contains 3 levels. The last part of PN_3 is $3n_0$, meaning that the dart of interest is filtered by the node n_0 of the third operation in the initial parametric specification. Therefore, the Orbit level 3 contains both n_0 and the matched orbit $\langle 0,1\rangle$. As shown in (List. 1), the third operation is the vertex insertion on an edge (Fig. 11). The right side of this rule has modified the orbit $\langle 0,1\rangle(n_0)$. We infer that the Event level 3 contains the third operation (`3-insert`) and the event's name `MODIFICATION`.

Continuing through the persistent name's backward traversal, the previous element `2n6` allows determining the DAG's second level in a similar way. Finally, `1n5` allows computing the first level. At last, the produced evaluation DAG (Fig 12) represents the volume face $\langle 0,1,3\rangle$-orbit's history resulting from applying `4-extrusion`. This DAG can be read top-down:

Level 1 The application `1-square` creates the volume face $\langle 0,1\rangle(n_5)$.

Level 2 From this volume face, `2-extrusion` creates the volume face $\langle 0,1\rangle(n_6)$.

Level 3 Finally, this latter volume face is modified by `3-insert`, inserting a vertex on its edge.

The application of the extrusion rule matches face $\langle 0,1\rangle(n_5)$. Then, the extrusion creates the face

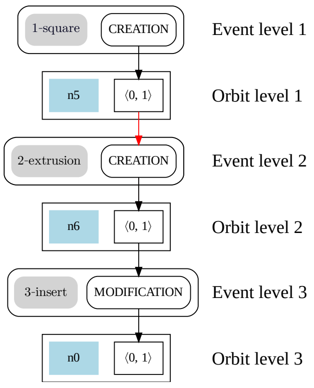


Figure 12: PN_3's evaluation DAG

$\langle 0,1\rangle(n_6)$ (Fig. 8b). Finally, applying the vertex insertion rule modifies the face $\langle 0,1\rangle(n_0)$ (Fig. 8c).

The syntactic analysis of the rules enables events to be computed only once. These events can be stored in cache to automatically build other evaluation DAGs.

4.2 Traces and origins

In order to accurately represent the history of an orbit, two types of arrows are used in an evaluation DAG (and later in the reevaluation DAG): black trace arrows and red origin arrows.

A *black trace arrow* allows orbit evolution tracing. For example, the triangulation rule splits an initial face into multiple subfaces. If one of these subfaces is referenced in the DAG, it is connected by a trace arrow to the initial face. Therefore, a trace arrow connects two orbits of the same dimension. A *red origin arrow* allows linking an orbit with the orbit that generated it, thereby connecting two orbits of potentially different dimensions. For example, continuing with the triangulation rule (Fig. 6), we observe that upon applying this rule, each edge of the initial face generates a different subface. If any of these subfaces is referenced in the DAG, it is then connected by an origin arrow to the edge that generated it. This is what can be observed considering again the example in Fig. 8 and more precisely the colour rule's PN_7 parameter. PN_7 represents dart 35's history and PN_7's evaluation DAG (Fig. 13) represents the history of the volume face that needs to be colored (the volume face adjacent to dart 35). This volume face $\langle 0,1\rangle(n_0)$ is the result of operation `5-triangulate` which splits the volume face $\langle 0,1\rangle(n_4)$ and has the face edge $\langle 0\rangle(n_4)$ as its origin (respectively represented by a black and red arrow between orbit level 2 and event level 3).

As explained in section 4.1, this DAG is built using a traversal of PN_7 and a bottom-up construction. This process is done in a similar way for both traces and origins, allowing for an efficient persistent naming mechanism that also takes into account the impact of origin modifications during reevaluation. To illustrate this, let us consider the previous face edge $\langle 0\rangle(n_4)$, which is the origin of the volume face $\langle 0,1\rangle(n_0)$ that needs to be colored. Suppose that, due to the addition of an operation in the edited specification, this origin may be split into two face edges. Upon applying `5-triangulate`, these two face edges will generate two volume faces, which can then be matched during reevaluation to the face to be colored, leveraging on the origin orbit recorded in the DAG.

In a formal way, an origin orbit can be automatically deduced through the syntactic analysis of a rule. More precisely, if n is a hook and n' is not a preserved node different from n, the origin of an orbit $R\langle o\rangle(n')$ is the suborbit $L\langle o'\rangle(n)$ consisting of the set of n's implicit arcs which are:

- rewritten on $R\langle o\rangle(n')$;
- not rewritten on $R\langle o\rangle(n')$ and belonging to $\langle o\rangle$.

For example, let us calculate the origin of a volume face ($\langle 0,1\rangle$-orbit) resulting from a split in the triangulation rule (Fig. 6). Only the implicit arc 0 in the $L\langle 0,1\rangle$-orbit incident to the hook n_0 is rewritten on $R\langle 0,1\rangle(n_0)$). Hence, the origin orbit of a volume face split by the triangulation rule is a face edge ($\langle 0\rangle$-orbit). When the

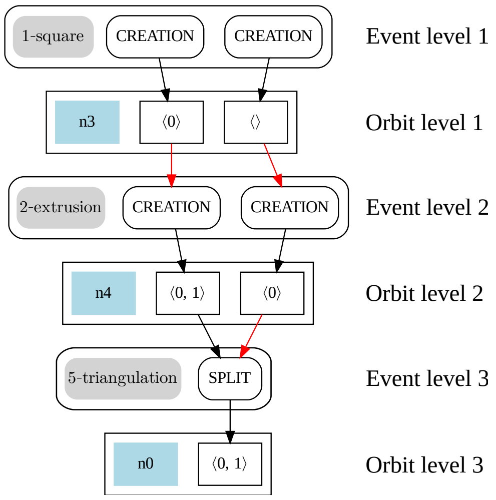


Figure 13: PN_7's evaluation DAG

orbit is either *split* or *merged*, the syntactic analysis allows the deduction of an *origin* in addition to a traced orbit. When the orbit is *created*, if the left-hand side of the rule is empty (meaning the orbit is created from scratch), there is neither trace nor origin (event level 1 in Fig. 13 where the 1-square rule creates the edge face $\langle 0\rangle(n_3)$ from scratch). Otherwise, there is no trace but an origin (event level 2 in Fig. 13 where the 2-extrusion rule creates the front face $\langle 0,1\rangle(n_4)$ of the cube from the origin previous face edge $\langle 0\rangle(n_3)$). When the orbit is just *modified* or *not modified*, there is only a traced orbit (event level 3 in Fig. 12 where the 3-insert rule modifies the top face $\langle 0,1\rangle(n_6)$ of the cube inserting a vertex on its boundary).

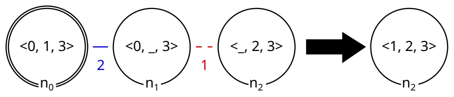


Figure 14: Rule collapsing a face into a vertex

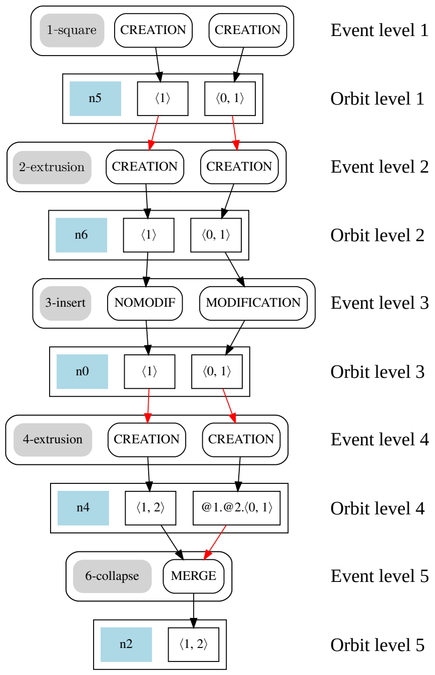


Figure 15: PN_6's evaluation DAG

4.3 Paths to origins

We have defined the origin of an orbit $R\langle o\rangle(n')$ when n' is not a preserved node different from the hook. In the opposite case, it is necessary to add to the origin the *path* that allows reaching the implicit arcs of the hook from n', because the implicit arcs of an origin are those of the hook, not those of n'.

For example, in the face collapse rule (Fig. 14), node n_2 is a preserved node different from the hook. Assume we want to define the origin of the volume vertex $\langle 1,2\rangle(n_2)$. A path represents the traversal in the right-hand side of the rule from the node n_2 to the hook node n_0. The traversed explicit arcs from n_2 to n_0 are, in the following order, 1 and 2 (written @1.@2 in the evaluation DAG). This can be seen in the PN_6's evaluation DAG (Fig. 15), where applying `6-collapse` generates a merge of vertices and the origin of the volume vertex $\langle 1,2\rangle(n_2)$ is the volume face @1.@2.$\langle 0,1\rangle(n_4)$. Actually, node n_4 of the extrusion rule (Fig. 4) used at the previous level of the DAG (level 4) matches dart 55 on the lateral faces of the cube (Fig. 5) because it is the dart matching the history stored in PN_6 DAG (`1-square` creates the initial bottom volume face $\langle 0,1\rangle(n_5)$. Then, `2-extrusion` applied on this bottom face creates the volume face $\langle 0,1\rangle(n_6)$ which is then modified by `3-insert` in $\langle 0,1\rangle(n_0)$. Finally, `4-extrusion` applied on this modified volume face creates node n_4, which matched dart 55). The volume face @1.@2.$\langle 0,1\rangle(n_4)$ reached starting from dart 55 and following links 1 and 2 is indeed the top face expected to be collapsed.

5 REEVALUATION

Each evaluation DAG represents an orbit's history which is valid with regards to the initial evaluation. When reevaluating, editing the parametric specification makes the topological parameters subject to changes. Thus, it is necessary to build reevaluation DAGs from the evaluation DAGs in order to update topological parameters. Once built, a reevaluation DAG can designate one, several, or no orbit depending on the editing of the parametric specification.

In this section, we keep using the previous example (Fig. 8 and 9) and its edited parametric specification (Lists 1 and 2), containing an added vertex insertion on the square's edge right after its creation and the deletion of the vertex insertion on the cube's edge.

5.1 Reevaluation DAG

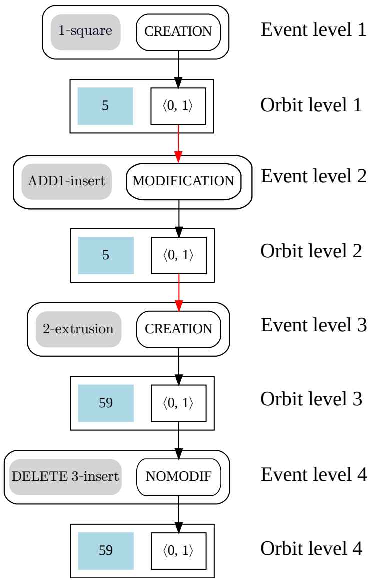


Figure 16: PN_3's reevaluation DAG

Contrary to the related evaluation DAG, a reevaluation DAG is built top-down throughout the reevaluation process. While an evaluation DAG represents the orbit's history of a topological parameter, the reevaluation DAG represents the history of this very same orbit after editing the parametric specification.

For example, let us consider PN_3 being the topological parameter of `4-extrusion`, which extrudes the cube's top face to produce a second cube. The reevaluation process builds PN_3's reevaluation DAG (Fig. 16 step-by-step from its evaluation DAG (Fig. 12):

Level 1 The application `1-square` has no topological parameter and, thus, is identically reevaluated. Once again, its node $n5$ creates a single dart 5 and the volume face $\langle 0,1\rangle(5)$ is identically reevaluated. This is why the orbit level 1 of the reevaluation DAG references dart 5.

Level 2 The second application is an added one. `ADD1-insert` does match an edge of the tracked volume face $\langle 0,1\rangle(5)$ and modifies it, as deduced from the rule. Therefore, the event level 2 contains `MODIFICATION` and the orbit level 2 contains the same dart 5 and orbit $\langle 0,1\rangle$.

Level 3 The third application is the extrusion of the square face into a cube. During its application, dart 5 is matched by the hook n_1 (Fig. 4). From the evaluation DAG, the tracked dart is the copy of dart 5 created by node n_6. Applying the rule allows finding out this dart, numbered 59.

Level 4 Finally, the last application of PN_3 is deleted. Consequently, the modification that occurred during the initial evaluation does not occur at reevaluation. Therefore, the event level 4 contains `NOMODIF` and the orbit level 4 contains the same dart 59.

The reevaluation DAG identifies the concrete name using its persistent name. PN_3's concrete name is 59 (cf. Fig. 9c and 9e). We now study a more complex example with the reevaluation DAG of PN_6 (Fig. 17),which represents the pyramid's top vertex (Fig. 8f):

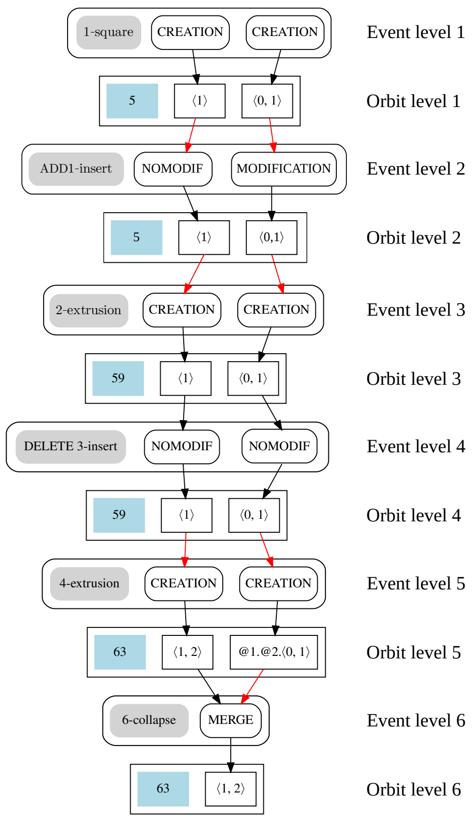


Figure 17: PN_6 reevaluation DAG

Level 1 to 4 These three levels are similar to PN_3's reevaluation DAG (Fig. 16), with the tracking of the volume face vertex $\langle 1\rangle$-orbit in addition to the $\langle 0,1\rangle$-orbit one. The $\langle 1\rangle(5)$-orbit is created by the `1-square` application. `ADD1-insert` matches and preserves one dart of this orbit with the node n_0 of the insertion rule (Fig. 11). Thus, the event level 2 contains `NOMODIF`. Then, this orbit is copied to create a new volume face vertex by `2-extrusion`. Again, `DELETE 3-insert` does not modify the orbits.

Level 5 The fifth application `4-extrusion` of the extrusion rule (Fig. 4) matches the volume face $\langle 0,1\rangle$ with its hook $n1$ and creates a copy of dart 59 dart 63. As in the initial evaluation, it creates a volume vertex and a face. Then, the event level contains two `CREATION`. The orbit level references dart 63 and contains both orbits $\langle 1,2\rangle$ and $@1.@2.\langle 0,1\rangle$.

Level 6 Finally, while the tracking of `5-collapse` shows that the application keeps merging the volume vertices incident to the matched face, it preserves dart 63 which is matched by node $n2$. Therefore, the event level contains one `MERGE` and the orbit level references dart 63 and contains the volume vertex $\langle 1,2\rangle$.

At last, PN_6's concrete name is dart 63 (cf Fig. 9g).

These two examples here are quite straightforward as there was only one possible candidate dart each time. However, in some complex specifications, there can be more than a single dart to choose between.

5.2 Parameter matching strategies

The editing of the parametric specification leads to having a different DAG at reevaluation (with event levels and/or branches being added, deleted or both). For example, an orbit split present in the evaluation DAG may disappear during the reevaluation, a merging can be added and so on. Several matching strategies can then be considered depending on the application's context. This can be illustrated with PN_7's example which designates the face that must be coloured (Fig. 8h). The addition of `ADD1-insert` application at reevaluation splits the origin of designated face, resulting in the addition of a branch in the reevaluation DAG. Let's work through PN_7's reevaluation DAG shown in Fig. 18:

Level 1 As seen previously, `1-square` creates the tracked orbits traced in the evaluation DAG.

Level 2 `ADD1-insert` matches dart 3. Since the vertex insertion rule (Fig. 11) splits the volume face edge $\langle 0\rangle(3)$, its history is also split and there are two concrete names to consider. It follows that the event level contains two `SPLIT`, one for each volume face edge suborbit, and two `NOMODIF`, one for each dart 3 and 4.

Level 3 `2-extrusion` extrudes the square face into a cube. The extrusion rule (Fig. 4) matches dart 3 with its hook n_1. It extrudes the face edge $\langle 0\rangle(3)$ into a volume face, the dart $\langle\rangle(3)$ into a face edge and its node n_2 creates the dart 37 as a copy of dart 3. The same goes

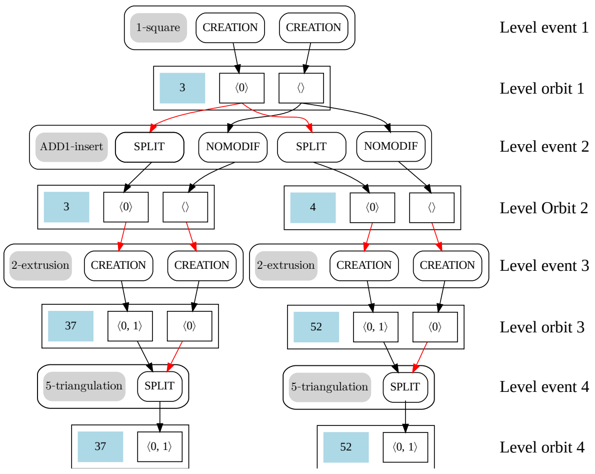


Figure 18: PN_7 reevaluation DAG

for the history on the right. The rule matches dart 4, it creates the same two orbits and a dart 52 as a copy of dart 4. both event levels contain two `CREATION`. Both orbit levels contain orbits $\langle 0,1\rangle$ and $\langle 0\rangle$.

Level 4 `5-triangulation` triangulates the faces designated by the level above. Both darts 37 and 52 are matched and preserved by the rule.

Upon reevaluation, PN_7's DAG matches to different darts. An option would be to colour only one face (either $\langle 0,1\rangle(37)$ or $\langle 0,1\rangle(52)$). Another option would be to apply `8-colour` two times, one for dart 37 and one for dart 52 (as shown in Fig. 9i which represent our default strategy). In case the reevaluation DAG has two or more leaves, it shows all the possible entities that can be matched for a specific persistent name and to set a strategy up. Either way, such a strategy allows users to have a choice and best fit their modeling intents, depending on the application's context.

6 CONCLUSION

In this paper, we widen the naming problem studies to the rule-based graph transformation modeling systems. We take advantage of the formalism of both generalized maps and graph transformation rules to tackle the reevaluation mechanism task. Generalized maps offer an homogeneous representation of an object in all dimensions while Jerboa's rules define geometric modeling operations on which it is actually possible to perform syntactical analysis. We implement: a persistent name scheme where each persistent name represents a unique dart's history through the successive applications of rules and their matching nodes. Then, for each persistent name, an evaluation DAG is built in order to trace an orbit's history from the bottom and up to the orbits it originates from. To our knowledge, unlike other methods, our solution tracks only the entities used in the parametric specification and the ones they originate from. Representing the complete history of an orbit in an evaluation DAG allows for an efficient persistent naming mechanism that takes into account the impact of both origins and traces modifications during reevaluation. Finally, reevaluation DAGs are built from a top-down traversal of evaluation DAGs and allow matching each topological parameter on one or more, sometimes none, values depending on the editing of the parametric specification. Thanks to our method, not only the naming problem is tackled within the usual framework of parameters edition, but we also take the specification edition into account (operation addition, deletion or move). Moreover, this approach provides all the possible updated values of the parameters and, thus, enables implementing different strategies.

More complex operations can make use of several rules brought together in a script. Later works will revolve around widening this reevaluation mechanism to scripts.

REFERENCES

[ALS15] K. Arroyo Ohori, H. Ledoux, and J. Stoter. "A dimension-independent extrusion algorithm using generalised maps". In: *International Journal of GIS* 29.7 (2015), pp. 1166–1186.

[Arn+22] A. Arnould et al. "Preserving consistency in geometric modeling with graph transformations". In: *Mathematical Structures in Computer Science* 32.3 (2022), pp. 300–347.

[Bab10] M. Baba-Ali. "Systeme de nomination hierarchique pour les systemes parametriques". PhD thesis. 2010. URL: `http://theses.univ-poitiers.fr/notice/view/5362`.

[Bei+10] J.N. Beirao et al. "Implementing a Generative Urban Design Model: Grammar-based design patterns for urban design". In: *eCAADe*. 2010, pp. 265–274.

[Bel+14] H. Belhaouari et al. "Jerboa: A graph transformation library for topology-based geometric modeling". In: *International Conference on Graph Transformation*. Springer. 2014, pp. 269–284.

[Ben+17] F. Ben Salah et al. "A general physical-topological framework using rule-based language for physical simulation". In: *12th International Conference on Computer Graphics Theory and Application (VISIGRAPP/GRAPP)*. 2017.

[BTG15] E. Bohl, O. Terraz, and D. Ghazanfarpour. "Modeling fruits and their internal structure using parametric 3Gmap L-systems". In: *The Visual Computer* 31 (2015), pp. 819–829.

[Car+19] A. Cardot et al. "Persistent naming based on graph transformation rules to reevaluate parametric specification". In: *CADA* 16.5 (2019), pp. 985–1002.

[CBS23] D. Cascaval, R. Bodik, and A. Schulz. "A Lineage-Based Referencing DSL for Computer-Aided Design". In: *Proceedings of the ACM on Programming Languages* 7 (2023), pp. 76–99.

[CH95] X. Chen and C. M. Hoffmann. "On editability of feature-based design". In: *CAD* 27.12 (1995), pp. 905–914.

[DL14] G. Damiand and P. Lienhardt. *Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing*. A K Peters/CRC Press, Sept. 2014.

[DZ24] F. Dai and W. Zhao. "A Persistent Naming Discrimination Method Based on the Sweeping Direction". In: *2024 4th International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. IEEE. 2024, pp. 75–83.

[ESR] ESRI. *ArcGIS CityEngine product page*. `https://www.esri.com/en-us/arcgis/products/arcgis-cityengine/overview`. Accessed 2023-05-09.

[FH18] S. H. Farjana and S. Han. "Mechanisms of persistent identification of topological entities in CAD systems: A review". In: *Alexandria engineering journal* 57.4 (2018), pp. 2837–2849.

[Gai+23a] M Gaide et al. "Automatic Detection of Topological Changes in Geometric Modeling Operations". In: *Computer Graphics and Visual Computing*. 2023, pp. 9–18.

[Gai+23b] M. Gaide et al. "Model Reevaluation Based on Graph Transformation Rules". In: *Computer Graphics and Visual Computing*. 2023, pp. 61–63.

[HH00] Y. Halbwachs and Ø. Hjelle. "Generalized maps in geological modeling: Object-oriented design of topological kernels". In: *Advances in Software Tools for Scientific Computing*. Springer. 2000, pp. 339–356.

[HMV09] S. Haegler, P. Müller, and L. Van Gool. "Procedural modeling for digital cultural heritage". In: *EURASIP Journal on Image and Video Processing* (2009).

[Hor+09] S. Horna et al. "Consistency constraints and 3D building reconstruction". In: *CAD* 41.1 (2009), pp. 13–27.

[Lie91] P. Lienhardt. "Topological models for boundary representation: a comparison with n-dimensional generalized maps". In: *CAD* 23.1 (1991), pp. 59–82.

[Lin74] A. Lindenmayer. "Adding continuous components to L-systems". In: *L systems* (1974), pp. 53–68.

[Mar06] D. Marcheix. "A persistent naming of shells". In: *International Journal of CAD/CAM* 6.1 (2006), pp. 125–137.

[MH05] D. Mun and S. Han. "Identification of topological entities and naming mapping for parametric cad model exchanges". In: *International Journal of CAD/CAM* 5.1 (2005), pp. 69–81.

[Mül+06] P. Müller et al. "Procedural modeling of buildings". In: *ACM SIGGRAPH Papers*. 2006, pp. 614–623.

[QB15] R. Quattrini and E. Baleani. "Theoretical background and historical analysis for 3D reconstruction model. Villa Thiene at Cicogna". In: *Journal of Cultural Heritage* 16.1 (Jan. 2015), pp. 119–125.

[Saf+20] M. Safdar et al. "Feature-based translation of CAD models with macro-parametric approach: issues of feature mapping, persistent naming, and constraint translation". In: *Journal of Computational Design and Engineering* 7.5 (2020), pp. 603–614.

[Ter+09] O. Terraz et al. "3Gmap L-systems: an application to the modelling of wood". In: *The Visual Computer* 25 (2009), pp. 165–180.

[Wu+01] J. Wu et al. "A face based mechanism for naming, recording and retrieving topological entities". In: *CAD* 33.10 (2001), pp. 687–698.

[Xue+16] G. Xue-Yao et al. "Name and Maintain Topological Faces in Rotating and Scanning Features". In: *International Journal of Grid and Distributed Computing* 9 (Mar. 2016), pp. 21–26.