Title:
Persistent Naming Based on Graph Transformation Rules to Reevaluate Parametric Specification

Authors:
Anaïs Cardot, anais.cardot@univ-poitiers.fr, University of Poitiers
David Marcheix, david.marcheix@ensma.fr, ENSMA
Xavier Skapin, xavier.skapin@univ-poitiers.fr, University of Poitiers

Keywords:
Persistent Naming, Parametric Specification, Rule-based Language, Topological Model

DOI: 10.14733/cadconfP.2018.387-391

Introduction:
CAD softwares are widely based on parametric specifications, which record operations used during the modeling process. Any specification edition results in new models after reevaluation. We address the issue known as *persistent naming problem*: the ability to find the right parameters and to reavaluate correctly each operation, despite the modeling object modifications. As an example, let a specification made of three operations be described in Fig. 1a: a cube is created, then a slot is applied (slicing face f into f_1 and f_2), followed by a slot inside f_2. Then, we shorten the first slot width then triggers the reevaluation (Fig. 1b): after the cube creation, its front face has not been sliced anymore, so face f_2 does not exist. The issue boils down to assign a name to each entity used in the initial evaluation in order to match it with another entity during reevaluation (f_x in this example). Most earlier works [2][3][4] separate

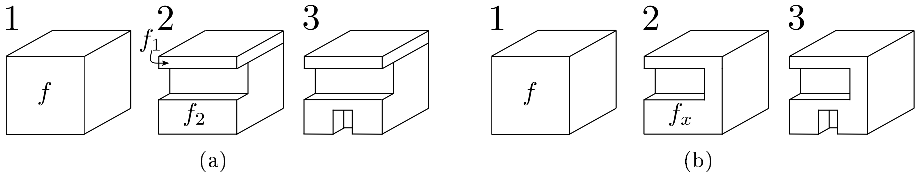


Fig. 1: Persistent Naming Problem; (a) Initial specification; (b) Reevaluation

2D from 3D processes and persistent naming solutions are entity dimension-specific. Some approach also record the history of every model entity, instead of keeping track of the ones used as operations parameter only. Finally, no solution has been presented to deal with every kind of specification edition.

Main idea:
We propose a persistent naming system independent of the model dimension, the dimension of entities (vertices, edges and so on) and withstanding the edition of the parametric specification (i.e. adding, deleting or moving operations). Our work relies on the modeling library *Jerboa* [2], based on the *G-Map* topological model [5]. This software describes any modeling operation as a transformation rule applied

© 2018 CAD Solutions, LLC, http://www.cad-conference.net

on a graph and prevents from any topological inconsistency. We detail how we have the mechanisms of parametric systems inside *Jerboa*.

G-Maps:

The d-dimensional *G-Map model* represents an object as a graph: nodes are called *darts* and each entity is defined as a specific set of darts linked by relationships denoted as α_i ($0 \leq i \leq d$). Constraints are defined on α_i to ensure that objects represented as G-Maps are quasi-manifolds. An example is shown in Fig. 2: (a) Geometric representation of a 2D object made of two faces; (b) Faces are separated from each other and are connected by an α_2 arc (blue line); (c) Face edges are separated and α_1 arcs (red lines) link their common vertices; (d) Edges are split and their ends are connected by α_0; (e) those ends form the graph nodes (darts), labeled $a, \ldots n$. This graph representation extends seamlessly in any dimension. An entity is represented by a single dart and a specific combination of α_i: for example, the face (resp. edge, vertex) related to dart e is made of all darts starting with a and reachable using α_0 and α_1 (resp. α_0 and α_2, α_1 and α_2) to produce the set of darts $\{a, \ldots, f\}$ (resp. $\{e, d, h, g\}$, $\{e, f, g, n\}$). In the following, the list of $\alpha_i, \ldots \alpha_j$ ($0 \leq i, j \leq n$) used to form an entity is called *orbit type* and is represented as $\langle i \ldots j \rangle$. For instance, the orbit type defining a 2D face is $\langle 01 \rangle$. More generally, for any d-dim, topological model, there are $(d+1)^2$ different orbit types, including the one designing a single dart and denoted as $\langle\rangle$; in 2D, those orbit types are $\langle\rangle$, $\langle 0 \rangle$, $\langle 1 \rangle$, $\langle 2 \rangle$, $\langle 01 \rangle$, $\langle 02 \rangle$, $\langle 12 \rangle$, $\langle 012 \rangle$.

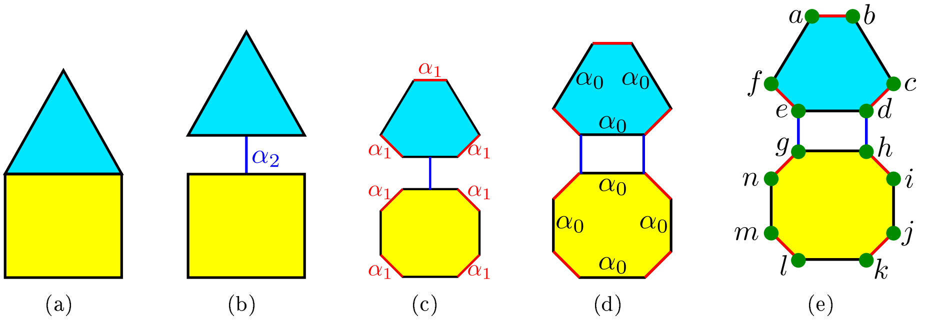


Fig. 2: G-Map representation of 2D objects; Only α_i relationships connecting pairs of different darts are represented

Graph Transformation Rules:

A graph transformation rule takes a G-Map as input and produces another G-map as output. Rules defined in *Jerboa* are graphically described as two patterns made of *nodes* separated by a left-to-right arrow (see Fig. 3a-b). The pattern on the left is to be filtered in the input G-Map, to identify every part which matches this pattern. The pattern on the right describes the transformation to apply on each of those matching parts, in order to produce the output G-Map. Fig. 3a shows the rule creates a triangular face from scratch: nodes $\mathbf{m_0}$ to $\mathbf{m_5}$ are generated from scratch and are linked using α_0 and α_1 to create a face as the one shown in Fig. 3c. In Fig. 3b, the left pattern is made of node $\mathbf{n_0}$ associated with the orbit type $\langle 01 \rangle$. Filtering any G-Map using this pattern returns all sets of darts linked by α_0 or α_1. The right pattern is made of three pairs (node, orbit type): $\mathbf{n_0}$ is copied in triplicate to produces nodes $\mathbf{m_0}$, $\mathbf{m_1}$ and $\mathbf{m_2}$ (in Fig. 3d, each dart $a_0, \ldots, f_0$ has been triplicated in $(a_0, a_1, a_2) \ldots$, (f_0, f_1, f_2)), forming the darts of the output G-Map. Regarding $\mathbf{m_0}$, its orbit type $\langle 0\_ \rangle$ means that $\mathbf{m_0}$ keeps the same relationship α_0 as $\mathbf{n_0}$, but α_1 is deleted. Regarding $\mathbf{m_1}$, $\langle \_2 \rangle$ means that the initial α_0 is deleted and the initial α_1 is replaced with α_2. We proceed the same regarding $\mathbf{m_2}$, Fig. 3e shows the state of the output G-Map at

Proceedings of CAD'18, Paris, France, July 9-11, 2018, 387-391
© 2018 CAD Solutions, LLC, http://www.cad-conference.net

this stage. Next, relationships α_0 and α_1 between $\mathbf{m_0}$, $\mathbf{m_2}$ and $\mathbf{m_2}$ are mirrored in the output G-Map (Fig. 3f): the initial face has been subdivided in three sub-faces in a consistent way. Note that this rule can be applied on any 2D polygonal face.

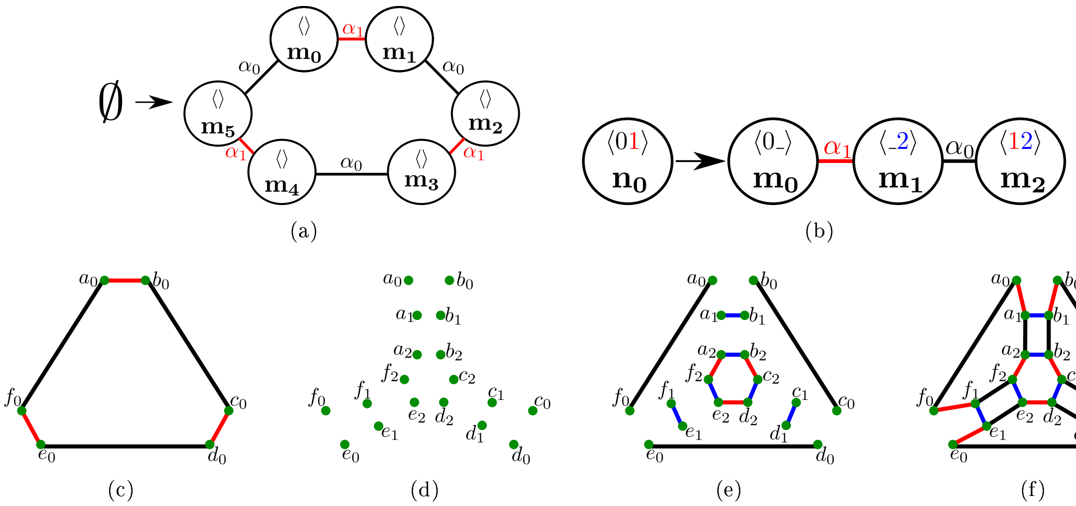


Fig. 3: Graph transformation rule; (a) *TriangleCreation* rule; (b) *FaceTriangulation* rule; (c) Input G-Map; (d) Darts of the output face; (e) First phase of α_i setup; (f) Second phase and output G-Map

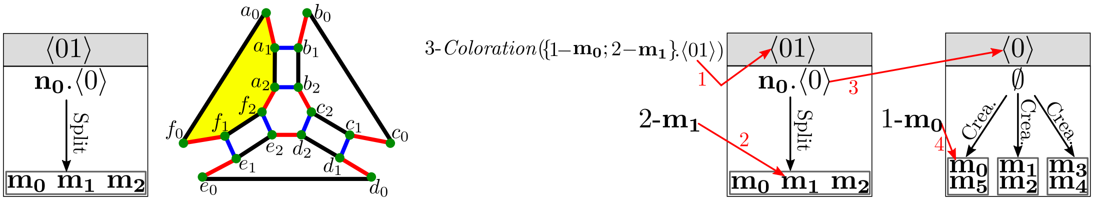
Persistent Naming System:

Persistent naming focuses on assigning an identifier to entities used as parameters of modeling operations, during the initial evaluation. This identifier should allow the matching between those entities and the right ones available in the reevaluation. Our naming system is based on darts: each dart of the entity upon which a rule is applied is assigned a *Persistent Id* (or *PId*). Each *PId* is composed of one or several pairs *RuleNum-NodeNum*, where *RuleNum* is the number of the rule in the initial evaluation and *NodeNum* is the node instance in the right part of the rule. As an example, a specification composed of two rules is shown in Fig. 3a-b: {1-*TriangleCreation*; 2-*FaceTriangulation*}. First rule generates nodes $\mathbf{m_0}$ to $\mathbf{m_5}$, so after applying it, the output G-Map contains darts a_0 to f_0 (Fig. 3c); their resp. *PId* are *1*-$\mathbf{m_0}$,...,*1*-$\mathbf{m_5}$. Second rule is then applied on the face, resulting in the G-Map shown in Fig. 3f. Each dart of this G-Map is associated with one node among $\mathbf{m_0}$, $\mathbf{m_1}$ or $\mathbf{m_2}$ and its *PId* is extended accordingly. As an example, darts a_0, a_1 and a_2 in Fig. 3f are derived from dart a_0 in Fig.3 b: they share the same *PId* prefix *1*-$\mathbf{m_0}$. Then, their *PId* is extended with specific information related to $\mathbf{m_0}$, $\mathbf{m_1}$ or $\mathbf{m_2}$: the complete *PId* of a_0 (resp. a_1, a_2) is therefore *{1*-$\mathbf{m_0}$*; 2*-$\mathbf{m_0}$*}* (resp. *{1*-$\mathbf{m_0}$*; 2*-$\mathbf{m_1}$*}*, *{1*-$\mathbf{m_0}$*; 2*-$\mathbf{m_2}$*}*). We proceed the same way for each dart of the output G-Map (as an example, *PId* of darts f_0, f_1 and f_2 are resp. *1*-$\mathbf{m_5}$*; 2*-$\mathbf{m_0}$, *1*-$\mathbf{m_5}$*; 2*-$\mathbf{m_1}$ and *1*-$\mathbf{m_5}$*; 2*-$\mathbf{m_2}$). Next, we define the *Persistent Name* (or *PN*) of entities which are used as parameters of operations, as the concatenation of the *PId* of one of its darts and the orbit type which represents this entity. For instance, the *PN* of the face related to dart a_0 in Fig. 3e is defined as *{1*-$\mathbf{m_0}$*; 2*-$\mathbf{m_0}$*}*.$\langle 01 \rangle$. Assume dart a_1 has been designated as the representative dart related to the face upon which 1-*TriangleCreation* has been applied. Then the specification's contents are {1-*TriangleCreation()*; 2-*FaceTriangulation*($\{1-\mathbf{m_0}; 2-\mathbf{m_1}\}.\langle 01 \rangle$)}.

Bulletin Boards and History Records:

© 2018 CAD Solutions, LLC, http://www.cad-conference.net

It is necessary to trace the evolution of orbit types between successive operations. Different kinds of evolutions are: *Creation*, *Deletion*, *Merge*, *Modification*, *No Change* and *Split*. To trace the evolution after a single operation, each rule is associated with a table called *Bulletin Board* (*BB*), and indexed by orbit types. Each table entry contains at least one tree graph with one or several leaves. The tree roots contain nodes of the rule's left part, combined with a specific orbit type. The tree leaves contain nodes of the rule's right part, which share the same leaf if they belong to the same orbit type as the entry index. The tree arcs are labeled with the kind of evolution undergone by orbit types. Fig. 4a describes the *Bulletin Board* entry indexed by $\langle 01 \rangle$ (2D face orbit) related to the Triangulation rule (Fig. 3b). The tree leaf regroups $\mathbf{m_0}$, $\mathbf{m_1}$ and $\mathbf{m_2}$, meaning that darts related to these nodes belong to the same 2D face in the output G-Map. Let (a_0, a_1, a_2) be those related darts: as shown in Fig. 3f, the face also contains f_0, f_1, f_2. It follows that all those darts have been generated using darts a_0 and f_0 of the input G-Map (Fig. 3c). a_0 and f_0 are related to node $\mathbf{n_0}$; moreover, since they are linked by α_0, the tree root is defined as $\mathbf{n_0}.\langle 0 \rangle$. The tree arc is labeled *Split*, meaning that the initial face has been split into several three sub-faces resp. generated by initial darts (a_0, f_0), (a_1, a_2) and (f_1, f_2). Every entry of the *BB* is filled in the same way.



(a) (b) (c)

Fig. 4: (a) Entry $\langle 01 \rangle$ of the *FaceTriangulation* rule's *Bulletin Board*; (b) Result of applying *Coloration* rule on the face (orbit type $\langle 01 \rangle$) related to a_1; (c) *History Record* (*HR*) of this face; Left: $\langle 01 \rangle$-indexed entry of *FaceTriangulation* rule; Right: $\langle 0 \rangle$-indexed entry of *TriangleCreation* rule

To trace evolutions along several operations, we used additional structures called *History Records*, defined for every *PN*. Let us add a third operation to the specification: 3-*Coloration*, which is applied on orbit type $\langle 01 \rangle$ related to dart a_1 (see Fig. 4b). As a reminder, a_1's *PId* is $\{1 - \mathbf{m_0}; 2 - \mathbf{m_1}\}$. Therefore, the *PN* used as parameter of 3-*Coloration* is $\{1 - \mathbf{m_0}; 2 - \mathbf{m_1}\}.\langle 01 \rangle$. The creation of the *HR* is achieved **backwards**, as shown in Fig. 4c, following red arrows: (1) Coloration is applied on $\langle 01 \rangle$ (related to a_1), which is used as an index in the *BB* related to 2-*FaceTriangulation*: the *BB*'s entry is shown in Fig. 4a. (2) Using the last part of a_1's *PId* ($\{2 - \mathbf{m_1}\}$), we select in this entry the tree which contains $\mathbf{m_1}$ and (3) we use the orbit type $\langle 0 \rangle$ defined in the tree root as an index for the *BB* related to 1-*TriangleCreation*. The matching entry is shown on the right side of Fig. 4c: nodes $\mathbf{m_0}$ to $\mathbf{m_5}$ shown in Fig. 3a are partitioned in three pairs of nodes linked by $\alpha 0$. The common root of these pair is $\emptyset$, meaning that have been created from scratch. (4) The first part of a_1's *PId* ($\{1 - \mathbf{m_0}\}$) is used to select the tree which contains $\mathbf{m_0}$. The orbit type related to the tree root is $\emptyset$, meaning that there is no previous operation.

Reevaluation:

To handle any edition of the initial specification, we create a *matching graph* for every parameter of interest, and we rely on its *HR* to determine how it is affected by the edition. As an example, we insert inside the specification the operation 1.1-*VertexInsertion*, between 1-*TriangleCreation* and 2-*FaceTriangulation*. Fig. 5a-d show the successive steps of the reevaluation. The *PN* build from a_1's *PId* and used as pa-

© 2018 CAD Solutions, LLC, http://www.cad-conference.net

rameter for 3-*Coloration* is $\{1-\mathbf{m_0}; 2-\mathbf{m_1}\}.\langle 01\rangle$; we use its *HR* and complete the *matching graph* at each step of the reevaluation to take into account 1.1-*VertexInsertion*. The *matching graph* is shown in Fig. 5e; note that although the *HR* has been created in reverse order, we follow it now in the correct order. (1) 1-*TriangleCreation* is reevaluated and a triangle identical to the initial one is created (Fig. 5a). The beginning of the *HR* (the $\langle 0\rangle$-indexed entry of *TriangleCreation* rule in Fig. 4c) indicates $1-\mathbf{m_0}$, so we consider $p_0.\langle 0\rangle$ (Fig. 5e), the orbit based on the dart generated from node $\mathbf{m_0}$ of the TriangleCreation rule. (2) 1.1-*VertexInsertion* is applied (Fig. 5b). The *BB* related to (not shown here) *VertexInsertion* rule indicates that there is a split for orbit type $\langle 0\rangle$: indeed, the edge composed of (p_0, u_0) has been split into distinct edges made of (p_0, w_0) and (u_0, v_0). Therefore, in the *matching graph*, two arcs labeled *Split* start from $p_0.\langle 0\rangle$ and point to one representative dart of each edge: we choose p_0 and u_0, still combined with $\langle 0\rangle$. (3) 2-*FaceTriangulation* is reevaluated (Fig. 5c). This time, similarly to first step, the entry related to $2-\mathbf{m_1}$ in the *BB*'s indexed $\langle 01\rangle$ (left part of Fig. 4c), leads to darts p_1 and u_1 generated by node m_1 of the FaceTriangulation rule: the *matching graph* is updated with an arc labeled *Split*, starting from p_0 (resp. u_0) and ending on p_1 (resp. u_1). (4) 3-*Coloration* is applied on each *matching graph*'s leaf, leading to the coloration of faces containing p_1 and u_1, as intended.

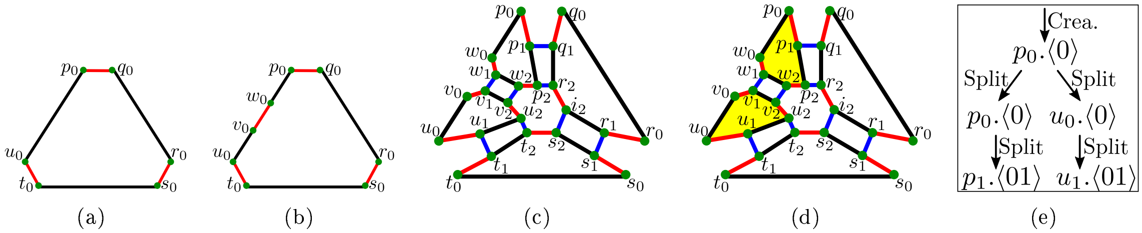


Fig. 5: Specification reevaluation; (a-d) Successive steps; (e) *matching graph*

Conclusion:

Our approach allows to reevaluate parametric specifications, regardless of the model dimension. The only kind of edition described is adding operation into specifications, but our method manages removing or changing operation order as well. Some development work on the *Jerboa* library is still required to test more complex specifications.

References:

[1] Baba-Ali, M.; Marcheix, D.; Skapin, X.: A Method To Improve Matching Process by Shape Characteristics in Parametric Systems. Computer-Aided Design and Applications, 3(16), 2009, 341-350. https://doi.org/10.3722/cadaps.2009.341-350

[2] Belhaouari, H.; Arnould, A.; Le Gall, P.; Bellet, T.: Jerboa, A Graph Tranformation Library for Topology-Based Geometric Modeling, Int. Conf. on Graph Transformation, 2014.

[3] Chen, X.; Hoffman, C-M.: Towards feature attachment, Computer-Aided Design, 27(9), 1995, 695-702. https://doi.org/10.1016/0010-4485(94)00027-B

[4] Kripac, J.: A mechanism for persistently naming topological entities in history-based parametric solid models, Computer-Aided Design, 29(2), 1997, 113-122. https://doi.org/10.1016/S0010-4485(96)00040-1

[5] Lienhardt, P.: N-dimensional generalized combinatorial maps and cellular quasimanifolds, Int. Journal of Computational Geometry and Applications, 4(3), 1994, 275?324. https://doi.org/10.1142/S0218195994000173

© 2018 CAD Solutions, LLC, http://www.cad-conference.net