Generic Computation of bulletin boards into Geometric Kernels

Mehdi Baba-Ali, David Marcheix, Xavier Skapin, Yves Bertrand

Generic Computation of bulletin boards into Geometric Kernels

Mehdi Baba-ali*
Signal Image Communications Lab
University of Poitiers.

David Marcheix†
LISI Lab/ ENSMA.

Xavier Skapin‡
Signal Image Communications Lab
University of Poitiers.

Yves Bertrand§
Signal Image Communications Lab
University of Poitiers.

Abstract

Nowadays, many commercial CAD systems are built on proprietary geometric kernels which provide an API containing a set of high level geometric operations (boolean operations, slot, chamfering, etc). Because of their complexity, these operations can generate important modifications on topological cells (vertices, edges, faces, volumes, etc.) of the objects. At the same time, many of these kernels need to know precisely what has occurred to each topological cell belonging to objects given or resulting from a previous high level geometric operation. At the end of each operation, the geometric kernel must provide a bulletin board describing cells' evolution through a list of events (split, merge, creation, deletion).
Most commercial geometric kernels use B-Rep structures and provide methods enabling the developer of a CAD system to retrieve a number of events that occurred on cells. These kernels have their own scheme for detecting events, based on their own taxonomy of situations, heuristics and evolution rules. Little is known of their details, which are proprietary information, let alone of the underlying theory, if any. Generally, for example, the detected events are not generic for all cells' dimensions. This lack of underlying theory limits the possibility to extend the use of these kernels to new domains of investigation.
In this paper, we propose a generic model that enables to create a bulletin board. This bulletin board will contain the complete list of events having occurred on cells of any dimension, and that belong to any topological model. The genericity of this model and the completeness in all dimensions of this list are based on the use of four elementary mechanisms (split_elem, merge_elem, crea_elem, del_elem). They are defined independently of the topological model, and allow the generation of the bulletin board, whatever the geometric operation. This model has been implemented using the geometric kernel of the modeler Moka, based on generalized maps.

Keywords: Bulletin board, Topological entity modification, Event follow-up mechanisms, Generalized maps

1 Introduction

Over the last fifty years or so, geometric modeling systems have evolved significantly. Initially limited to 2D, they now include complex 3D functionalities, ranging from the simulation of physical phenomena to the recording and complete and automatic replay of parameterized processes of conception.

In the field of CAD (Computer Aided Design), some parametric systems have imposed themselves on the market, and the number of hours of development dedicated to these systems is so huge that it is very difficult to redevelop a new complete rival system.

Therefore, most new modeling systems which appear on the market basically use a ready -to-use geometric kernel (*Parasolid*, *Acis*, *Cascade* [Brunier-Coulin et al. 2000], etc.). These kernels supply the modeling system with an API (Application Programming Interface) containing a set of high level geometric operations (boolean operations, slot, chamfering, etc.).

Structurally, a geometric modeler is composed of two different levels of abstraction (see figure 1). The first level, in direct interaction with the user, includes the application layer which gathers the high level geometric operations. The second level, forming the core of the modeler, includes a geometric kernel based on a specific model (B-Rep, CSG, etc.).

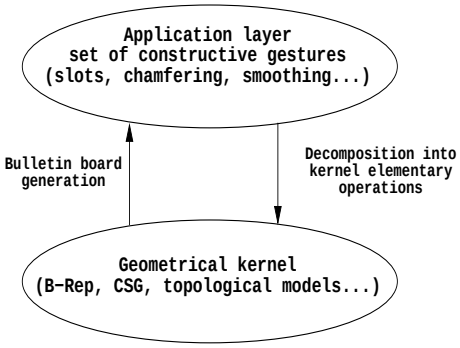


Figure 1: *Geometric modeler structure.*

Because of their complexity, the application layer's operations can generate important modifications on topological cells (vertices, edges, faces, volumes, etc.) of the objects. At the same time, many of these operations need to know precisely what has occurred to each topological cell belonging to objects given or resulting from a previous high level geometric operation. Next, at the end of each operation, the geometric kernel must provide a bulletin board describing cells' evolution through a list of events (split, merge, creation, deletion). For example, in the field of CAD, most parametric systems have developed home-grown solutions to resolve the persistent naming problems [Marcheix and Pierra 2002] (in our work, the persistent naming is just a case study). These solutions frequently use a graph in order to save face history during the conception process ([Kripac 1995], [Marcheix and Pierra 2002]). The construction of this graph needs to know how the faces of the geometric model evolve during an operation. In figure 2, block face $f_{1.1}$ is split into two faces ($f_{2.1}$ and $f_{2.2}$) after the application of

*e-mail: babaali@sic.sp2mi.univ-poitiers.fr
†e-mail:marcheix@ensma.fr
‡e-mail:xavier.skapin@univ-poitiers.fr
§e-mail:yves.bertrand@univ-poitiers.fr

the *difference* boolean operation. The graph in figure 2b stores this event that must be returned by the geometric kernel.

Currently, most geometric systems are developing new functionalities about standard construction procedures such as boolean operations for CSG or Euler operators for B-Rep modeling. These systems offer some solutions to the new needs expressed in many fields, such as CAD, architecture or geology. However, these solutions aren't satisfactory. Indeed, all commercial geometric kernels provide methods enabling the developer of a CAD system to retrieve a number of events occurred on cells. These kernels have their own scheme for detecting events, based on their own taxonomy of situations, heuristics and evolution rules. Little is known of their details, which are proprietary information, let alone of the underlying theory, if any. Generally, for example, the detected events are not generic for all cells' dimensions. This lack of underlying theory limits the possibility to extend the use of these kernels to new domains of investigation which need to detect events appearing on all i-cells (a cell of dimension i where i=0..n) and different aggregates of connected i-cells. An interesting formalization work has been proposed in 2000 in the DJINN project [Armstrong et al. 2000]. This report proposes a complete specification of the different functions that must be provide by the interface of a geometric kernel. In this report, we can find the specification of several functions dedicated to the management of events occured on cells. This necessary formalization work emphases the necessity to provide these functions but it gives no answer on the way of generating in a correct an generic way these events in the geometric kernel. An underlying theory is necessary in order to guarantee the reliability and the completeness of the generated events.

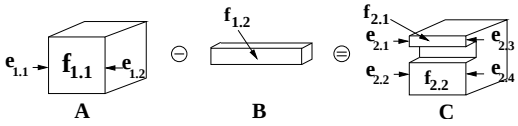
In this paper, we are interested in the different modifications that a high level geometric operation generates on a modeler, particularly in the case of a topology based kernel. In order to solve this problem, we propose a generic model that enables to insert the complete list of events (we must omit any event) having occurred on cells of any dimension, and that belong to any topological model, into a bulletin board. The genericity and the complete independence with the used geometric model are based on four elementary mechanisms (split_elem, merge_elem, crea_elem, del_elem) allowing the generation of the bulletin board, whatever the geometric operation.

This paper is structured as follows. In section 2, we present our solution for the generic generation of bulletin boards into the topological kernel. Section 3 describes an implementation of this solution using the geometric kernel of the modeler Moka (web site: http://www.sic.sp2mi.univ-poitiers.fr/moka) that is based on the topological model of generalized maps [Lienhardt 1994]. This process permits us to integrate and validate the event follow-up mechanisms. We conclude in Section 4.

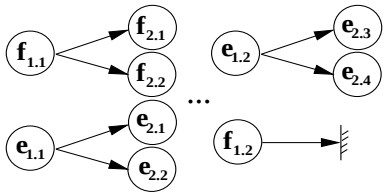
2 List of events generation

In this paper, our objective is to generalize the event follow-up mechanisms in order to describe a formalism robust enough to be implemented on any type of topological model.

The events which have occurred on the cells can be represented in the bulletin board as a list (every current geometric modeling system - Parasolid, Cascade, and so forth - has such a structure). This list must be complete and generic in any cell dimension. A bulletin board allows the tracking of topological cells' evolution inside a geometric model, after the application of a high level geometric operation (see figure 2a). To do so, the bulletin board links two sets of cells (respectively named *starting cells* and *ending cells*) with an event. Each event represents an interpretation of the topological evolution which has occurred on the starting cells (see figure 2c).



(a) Boolean operation of difference.



(b) Cell graph corresponding to a constructive gesture applied on the example of figure 2a.

Starting cells	*Ending cells*	Event
$\{f_{1.1}\}$	$\{f_{2.1}, f_{2.2}\}$	**face split**
$\{e_{1.1}\}$	$\{e_{2.1}, e_{2.2}\}$	**edge split**
…	…	…
$\{e_{1.2}\}$	$\{e_{2.3}, e_{2.4}\}$	**edge split**
$\{f_{1.2}\}$	$\{\ \}$	**face deletion**

(c) Bulletin board corresponding to a constructive gesture applied on the example of figure 2a.

Figure 2: *Events occured on cells after the application of a high level geometric operation.*

In figure 2a, the boolean operation between volumes A and B (high level geometric operation) generates several events and the corresponding bulletin board links several *starting cells* and *ending cells* sets (one pair set per line). Indeed, in the bulletin board shown in figure 2c, the face split event gathers the starting set $\{f_{1.1}\}$ and the ending set $\{f_{2.1}, f_{2.2}\}$ (see line 1). Moreover, the face deletion event gathers the starting set $\{f_{1.2}\}$ and the empty ending set $\{\}$ (see line 5).

2.1 Prerequisites

In a geometric modeler, we distinguish two levels of operations called "high level operation" and "elementary operation" (see figure 3). A high level operation (such as boolean operation) is a geometric procedure describing a complex geometric process. The topology resulting from this type of operation is highly dependent on geometry. Thus, the event list linking all *starting cells* and all *ending cells* can not be determined at this level (this is called "unpredictable"). For example, in figure 2a, the split of face $f_{1.1}$ depends on the position of volume B. Thus, to find the events, after the application of the boolean operation, between all *starting cells* $\{f_{1.1}, e_{1.1}, \ldots, e_{1.2}, f_{1.2}\}$ and all *ending cells* $\{f_{2.1}, f_{2.2}, \ldots, e_{2.3}, f_{2.4}\}$ (see figure 2c) is unfeasable. Available in application layer (see figure 1), a high level operation must be processed at the geometric kernel of a modeler with a set of elementary operations.

Unlike a high level operation, an elementary operation generates a predictable and finite list of events (figure 4 shows an elementary operation of *face splitting*, applied on F). This list includes the events of creation, deletion, merge and split of cells. In figure 4, we can only find two edge split (A_1, A_2), one face split (F) and two edge creation (A_3, A_4) events. In order to transcribe the events on the bulletin board, we define four mechanisms for following up events. Each of them is associated with the elementary operation (see figure 3) which generates the corresponding event. Indeed, the events' split, merge, creation and deletion of cells respectively match the mechanisms split_elem, merge_elem,

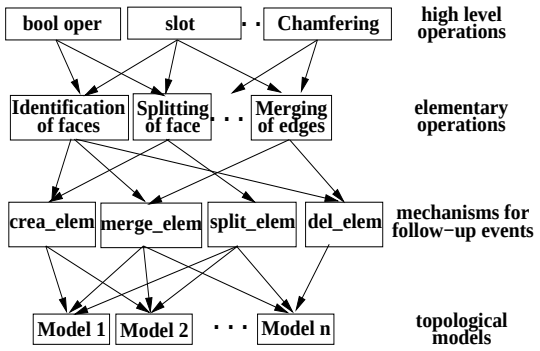


Figure 3: *Architecture of our model to generate the list of events which have occurred on cells.*

crea_elem and del_elem.

2.2 Mechanisms for event follow-up

To define these mechanisms, we need to introduce the definitions below:

- ID_Cell: an infinite set of identifiers which allow the characterization of every cell.
- ID_used: the set of the identifiers used during a high level operation ($ID\_used \subset ID\_Cell$).
- Dim_Max: the maximal dimension of cells in the topological model. In $2D$ space (resp. in $3D$ space), $Dim\_Max = 2$ (resp. $Dim\_Max = 3$).

After the application of any high level operation O, a set of events have occurred on the cells of the topological model. The identification of these elements consists in finding the links between the starting cells and the ending cells of O. We proceed as follows:

1. characterize each cell c by an identifier id ($id \in ID\_Cell$);
2. associate the pair (set_id, dim) with id. set_id represents a set of identifiers refering to all the cells from which c has been taken. set_id can be an empty set if c has just been created and dim represents the dimension of c. In figure 2a, pair $(\{f_{1.1}\}, 2)$ must be associated with an identifier $f_{2.2}$ because face $f_{2.2}$ originated from face $f_{1.1}$.

The faces F, F_1 and F_2 in figure 4a are respectively characterized by the identifiers id_1, id_4 and id_5 (see figure 4b). Identifier id_1 is associated with the pair $(\{id_1\}, 2)$ because we apply the initialization process of ancestors described below. Faces F_1 and F_2 derive from F, so both pairs $(\{id_1\}, 2)$ and $(\{id_1\}, 2)$ are respectively associated with identifiers id_4 and id_5.

The relations called $ancestor$ and $dimension$ are respectively defined by:

$$ancestor : \left\{ \begin{array}{l} ID\_Cell \rightarrow \mathfrak{P}(ID\_Cell) \\ id \mapsto ancestor(id) = set\_id \end{array} \right. \text{ and}$$

$$dimension : \left\{ \begin{array}{l} ID\_Cell \rightarrow [0, Dim\_Max] \\ id \mapsto dimension(id) = dim \end{array} \right.$$

($\mathfrak{P}(ID\_Cell)$ symbolizes the power set of ID_Cell).

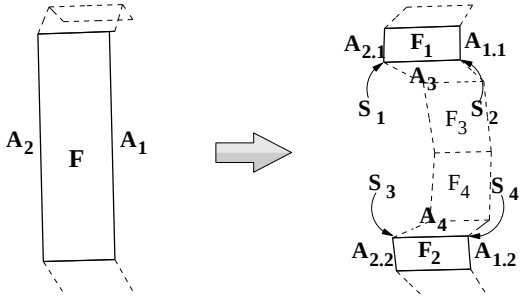
The ancestor of c must be initialized at the beginning of a high level operation. This process consists in giving the value of singleton $\{id\}$ to each set set_id. We formalize this initialization by:
$\forall id \in ID\_Cell ; id \in D_{ancestor} \Rightarrow ancestor(id) = \{id\}$
($D_{ancestor}$ is the range of function $ancestor$).

Let us consider the example shown in figure 4 and let us assume that we apply the initialization process on faces F_1 and F_2 at the beginning of the following high level operation. Identifier id_4 (resp. id_5) is associated with the set $\{id_4\}$ (resp. $\{id_5\}$).

All these definitions allow us to define mechanisms of an event follow-up formally. We only describe the mechanism *"split_elem"* because the principle is the same for every other mechanism.

2.2.1 Mechanism "split_elem"

Splitting cell c of dimension dim ($dim \neq 0$) results in two new cells c_1 and c_2 with the same dimension. Identifier id (resp. identifiers id_1 and id_2) which characterizes the cell c (resp. cells c_1 and c_2) is associated with the set of identifiers set_id (resp. two sets of identifiers set_id_1 and set_id_2). The mechanism *split_elem* allows to define set_id_1 and set_id_2 such as $set\_id_1 \subseteq set\_id$, $set\_id_2 \subseteq set\_id$ and $set\_id_1 \cup set\_id_2 = set\_id$. For example, in figure 4, we assume that we initialize the $ancestor$ of cell F before starting the *face splitting* elementary operation.



(a)

Cell	Identifier id_i	Set of identifiers (set_id_i)	Dimension
F	id_1	$\{id_1\}$	2
A_1	id_2	$\{id_2\}$	1
A_2	id_3	$\{id_3\}$	1
F_1	id_4	$\{id_1\}$	2
F_2	id_5	$\{id_1\}$	2
A_3	id_6	$\{\}$	1
A_4	id_7	$\{\}$	1
$A_{1.1}$	id_8	$\{id_2\}$	1
$A_{1.2}$	id_9	$\{id_2\}$	1
$A_{2.1}$	id_{10}	$\{id_3\}$	1
$A_{2.2}$	id_{11}	$\{id_3\}$	1
S_1	id_{12}	$\{\}$	0
S_2	id_{13}	$\{\}$	0
S_3	id_{14}	$\{\}$	0
S_4	id_{15}	$\{\}$	0

(b)

Figure 4: *Elementary operation "face splitting". a) Boundary representation of split face F. b) Information associated with some cells of the model shown in (a).*

set_id_1 is initialized with the singleton $\{id_1\}$ (id_1 characterizing F). After splitting F, identifiers id_4 and id_5, which respectively characterize faces F_1 and F_2, are associated with the same set set_id_1 ($set\_id_4 = set\_id_5 = set\_id_1$). Both F_1 and F_2 have F as ancestor and the sets of identifiers associated with id_4 and id_5 follow the preconditions ($\{id_1\} \subseteq \{id_1\}$ and $\{id_1\} \cup \{id_1\} = \{id_1\}$).

The mechanism "split_elem", described by algorithm 1, adds new relations linking the identifiers which characterize cells c_1 and c_2

Algo. 1 mechanism `split_elem`

Data: ▪ id, id_1, id_2: three identifiers;
▪ set_id_1, set_id_2: two sets of identifiers which characterize the origin of cells c_1 and c_2.

Result: adding the relations "$ancestor$" and "$dimension$" related to cells c_1 and c_2.

begin
  if $\exists\, id, id_1, id_2, set\_id_1, set\_id_2$
  such as
1 $dimension(id) \neq 0$ ***and***
2 $id \in D_{ancestor}$ ***and***
3 $id_1 \in ID\_Cell - ID\_used$ ***and***
   $id_2 \in ID\_Cell - ID\_used$ ***and***
   $id_1 \neq id_2$ ***and***
4 $set\_id_1 \subseteq ancestor(id)$ ***and***
   $set\_id_2 \subseteq ancestor(id)$ ***and***
5 $ancestor(id) = set\_id_1 \cup set\_id_2$
  then
6 $ID\_used \leftarrow ID\_used \cup \{id_1, id_2\}$;
7 $ancestor \leftarrow ancestor \cup \{(id_1, set\_id_1), (id_2, set\_id_2)\} - \{(id, ancestor(id))\}$;
   $dimension \leftarrow dimension \cup \{(id_1, dimension(id)), (id_2, dimension(id))\} - \{(id, dimension(id))\}$;
end

with their origin and their dimension (in order to update the functions $ancestor$ and $dimension$ defined in section 2.2). This mechanism takes the identifier id (resp. id_1 and id_2) characterizing cell c (resp. c_1 and c_2) and the sets set_id_1 and set_id_2, respectively characterizing the origin of cells c_1 and c_2, as parameters. Points 1 to 5 present the preconditions of algorithm 1; the algorithm itself is described in points 6 and 7.

1. We do not consider the mechanism *split_elem* on 0-dimensional cells. Indeed, the split of a vertex has, from a semantic point of view, no meaning.
2. id belongs to the range of function $ancestor$.
3. id_1 and id_2 are two new identifiers not yet used.
4. The set set_id_1 (resp. set_id_2) characterizes the ancestor of c_1 (resp. c_2) and is included in the set which characterizes the ancestor of c.
5. The union of set_id_1 and set_id_2 corresponds to the set $ancestor(id)$ which gathers the ancestors of c. Therefore, all the ancestors of c are added to the ancestors of both c_1 and c_2. Only the implementation of the mechanism, in accordance with a specific topological model, makes it possible to determine the exact contents of sets set_id_1 and set_id_2.
6. id_1 and id_2 are added to set ID_used.
7. The values of ancestor and dimension are updated by adding (resp. subtracting) the ancestor and the dimension of both c_1 and c_2 (resp. c).

2.3 Bulletin board generation

After relating each identifier id (and thus each cell c) of the geometric model to a set of identifiers set_id (resp. a dimension dim) characterizing the ancestor (resp. dimension) of c, we check every cell of this model at the end of high level operations (*i.e. a posteriori*), in order to retrieve every relation binding all identifiers id to their ancestor set of identifiers set_id and to the dimension dim of the cell they reference. These relations must be inserted in the bulletin board, in order to distinguish the events occurring both on the starting cells and the ending cells during any high level operation.

Of course, when deleting a cell, the recovery of the relations described just before cannot be only carried out *a posteriori*. Indeed, deleting a cell implies the loss of information which were associated with this cell. To solve this issue, first we propose to check the cells at the beginning of high level operations. The sets characterizing the ancestor of each identifier are gathered into a set D_1. Therefore, at the end of the high level operation, we gather the identifiers characterizing the ancestors of every existing cell in a second set D_2 and thus, we can determine which cells have been deleted by defining the set D ($D = D_1 - D_2$).

With the formalism defined in the previous section, checking cells can be considered as a checking set E defined by $E = \{(id, ancestor(id)) | id \in D_{ancestor}\}$. Some elements of E are gathered in n subsets (n may be equal to 0). Each subset, called S_{E_k} ($k \in [0, n]$), contains elements x of E ($x = (id, set\_id)$) which have the same set_id. Therefore, all cells with the same ancestor are gathered. Formally, S_{E_k} are defined as:
$S_{E_k} = \{\forall x_i, x_j | (x_i, x_j) \in S_{E_k}^2 \Rightarrow (x_i, x_j) \in E^2$ *and* $x_i = (id_i, set\_id_i)$ *and* $x_j = (id_j, set\_id_j)$ *and* $set\_id_i = set\_id_j$ *and* $set\_id_i \neq \emptyset\}$.

Figure 4b shows the information associated with the cells used by the elementary operation of *face splitting*. Set E, built at the end of the high level operation, contains pairs $(id_4, \{id_1\})$, $(id_5, \{id_1\})$, $(id_6, \{\})$, $(id_7, \{\})$, $(id_8, \{id_2\})$, $(id_9, \{id_2\})$, $(id_{10}, \{id_3\})$, $(id_{11}, \{id_3\})$, ..., $(id_{15}, \{\})$. With these pairs, we can define subsets $S_{E_0} = \{(id_4, \{id_1\}), (id_5, \{id_1\})\}$, $S_{E_1} = \{(id_8, \{id_2\}), (id_9, \{id_2\})\}$ and $S_{E_2} = \{(id_{10}, \{id_3\}), (id_{11}, \{id_3\})\}$. Every element of each subset has the same set_id.

We build a new set called S from subsets S_{E_k}. Each pair (A_k, B_k) of S gathers elements contained in the previously defined subset S_{E_k} (A_k and B_k are two sets of identifiers defined from S_{E_k}). Therefore, S is defined: $S = \{x_k = (A_k, B_k) | k \in [0, n]$ *and* $\forall x = (id, set\_id) | x \in S_{E_k} \Rightarrow id \in B_k$ *and* $A_k = set\_id\}$. This set gathers all events of cell split. In figure 4, S (built from the subsets S_{E_0}, S_{E_1} and S_{E_2}) contains elements $(A_0, B_0) = (\{id_1\}, \{id_4, id_5\})$, $(A_1, B_1) = (\{id_2\}, \{id_8, id_9\})$ and $(A_2, B_2) = (\{id_3\}, \{id_{10}, id_{11}\})$. From set D of deleted cells, we also build that set Z defined by: $Z = \{\forall x | x \in D \Rightarrow (\{x\}, \{\}) \in Z\}$.

For every other case (create and merge events), we define set A by: $A = \{\forall x | x \in E - \bigcup_{k=0}^{n} S_{E_k} \Rightarrow x = (id, set\_id)$ *and* $(set\_id, \{id\}) \in A\}$. In figure 4, set $E - \bigcup_{k=0}^{n} S_{E_k}$ contains elements $(id_6, \{\})$, $(id_7, \{\})$, $(id_{12}, \{\})$, $(id_{13}, \{\})$, $(id_{14}, \{\})$ and $(id_{15}, \{\})$ (see figure 4b). Therefore, A contains $(\{\}, \{id_6\})$, $(\{\}, \{id_7\})$, $(\{\}, \{id_{12}\})$, $(\{\}, \{id_{13}\})$, $(\{\}, \{id_{14}\})$ and $(\{\}, \{id_{15}\})$.

Eventually, every line of the bulletin board matches an element of set BB defined by: $BB = S \cup Z \cup A$. Indeed, for each pair $x = (e_1, e_2)$ of BB, we put e_1 among the starting cells and e_2 among the ending cells of the bulletin board. In figure 4, the elements of S are located on the first line of table 1, the elements of set A are located on the last line and Z does not contain any element.

Starting cells	Ending cells	Remark
$\{id_1\}$	$\{id_4, id_5\}$	$(\{id_1\}, \{id_4, id_5\}) \in S$
...	...	...
$\{\}$	$\{id_{12}\}$	$(\{\}, \{id_{12}\}) \in A$

Table 1: *Bulletin board generated after applying a high level operation which contains the elementary operation shown in figure 4.*

2.3.1 Interpretation of events

After inserting the elements of set BB into the bulletin board, the next step is to identify which events have occured (creation, deletion, split, merge, change) by looking at the cells located on the same line in the bulletin board. Our criterion used for the identification of events is the cardinality of both the set of starting cells ($\texttt{card}(SC)$) and the set of ending cells ($\texttt{card}(EC)$) inside the bulletin board. The identification process for each event is described like this:

- if SC is an empty set then the corresponding event is creation (see line 3 in Tab. 1).
- if EC is an empty set then the corresponding event is deletion.
- if $\texttt{card}(SC) = 1$ and $\texttt{card}(EC) > 1$ then the corresponding event is split (see line 1 in Tab. 1).
- if $\texttt{card}(EC) = 1$ and $\texttt{card}(SC) > 1$ then the corresponding event is merge.
- if $\texttt{card}(EC) = 1$ and $\texttt{card}(SC) = 1$ then the corresponding event is change.

Therefore, we deduce that: (1) set S contains only pairs describing the event "cell split" (see Tab. 1); (2) set Z exclusively contains pairs describing the event "cell deletion"; (3) set A contains pairs describing events of "cells creation" and "cell merge".

In the following section, we describe an implementation of the model presented above. We use the kernel of the modeler Moka (based on generalized maps) in order to show a case study.

3 Case study: generalized maps

The choice of generalized maps (G-maps) model as a study case is not fortuitous. Indeed, this model (see figure 5) enables to subdivide an nD space (in the example below, we limit us to the 3D case) into n-dimensional quasi-manifolds, orientable or not, with or without boundary [Lienhardt 1994]. Therefore, we have chosen this model for its generality, although we could have used any other B-rep model. In order to implement the previously described solution on the G-maps model, we have defined a structure called *tag*. In this section, we describe this structure as well as the various follow-up event mechanisms.

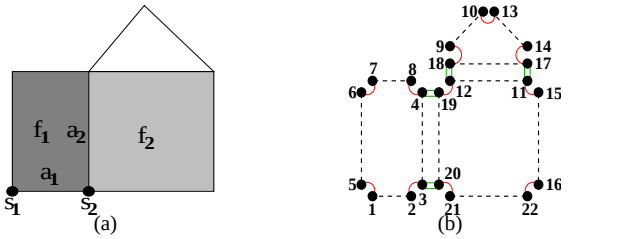
3.1 Generalized maps model

An n-dimensional generalized map is a set of abstract elements, called darts, and applications defined on these darts:

Definition 1 *(Generalized map) Let $n \geq 0$. A n-dimensional generalized map (or n-G-map) is $G = (B, \alpha_0, \ldots, \alpha_n)$ where:*

1. *B is a finite set of darts;*
2. *$\forall i, 0 \leq i \leq n, \alpha_i$ is an involution[1] on B;*
3. *$\forall i, j, 0 \leq i < i+2 \leq j \leq n, \alpha_i\alpha_j$ is an involution.*

[1] An involution f on S is a one mapping from S onto S such that $f = f^{-1}$.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
α_0	2	1	4	3	6	5	8	7	10	9	12	11	14	13	16	15	18	17	20	19	22	21
α_1	5	3	2	8	1	7	6	4	18	13	15	19	10	17	11	22	14	9	12	21	20	16
α_2	1	2	20	19	5	6	7	8	9	10	17	18	13	14	15	16	11	12	4	3	21	22

Figure 5: *(a) A 2D subdivision. (b) The corresponding 2-G-map (involutions are given explicitly in the array). Darts are represented by dots. Two darts in relation by α_0 share a dashed line (ex. darts 1 and 2). Two darts in relation by α_1 share a red arc (ex. darts 2 and 3). Two distinct darts in relation by α_2 share a pair of green lines (ex. darts 3 and 20); otherwise, the dart is its own image by α_2 (ex. dart 2). Dart 1 corresponds to (s_1, a_1, f_1), dart $2 = 1\alpha_0$ corresponds to (s_2, a_1, f_1), $3 = 2\alpha_1$ corresponds to (s_2, a_2, f_1), and $20 = 3\alpha_2$ corresponds to (s_2, a_2, f_2). The vertex incident to dart 2 is $\langle\alpha_1, \alpha_2\rangle(2) = \{2, 3, 20, 21\}$, the edge incident to dart 3 is $\langle\alpha_0, \alpha_2\rangle(3) = \{3, 4, 19, 20\}$, and the face incident to dart 9 is $\langle\alpha_0, \alpha_1\rangle(9) = \{9, 10, 13, 14, 17, 18\}$.*

Let G be an n-G-map, and S be the corresponding subdivison. Intuitively, a dart of G corresponds to an (n+1)-tuple of cells $(c_0, ..., c_n)$, where c_i is an i-dimensional cell that belongs to the boundary of c_{i+1} [Brisson 1993] (Fig. 5). α_1 associates darts corresponding with $(c_0, \ldots, c_n)$ and $(c'_0, \ldots, c'_n)$, where $c_j = c'_j$ for $j \neq i$, and $c_i \neq c'_i$ (α_i swaps the two i-cells that are incident to the same $(i-1)$ and $(i+1)$-cells). When two darts b_1 and b_2 are such that $b_1\alpha_i = b_2$ $(0 \leq i \leq n)$, b_1 is said i-sewn with b_2. G-maps represent cells in an implicit way:

Definition 2 *(i-cell) Let G be an n-G-map, b a dart and $i \in N = \{0, ., n\}$. The i-cell incident to b is the orbit*[2]

$$\langle\rangle_{N-\{i\}}(b) = \langle\alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n\rangle(b)$$

Intuitively, an i-cell is the set of all darts which can be reached starting from b, by using any combination of all involutions except α_i. The set of i-cells is a partition of the darts of the G-map, for each i between 0 and n. Two cells are disjoined if their intersection is empty, i.e. when no dart is shared by the cells. More precisions about G-maps are provided in Ref. [Lienhardt 1994].

3.2 Solution brought by the "tag" structure

Tracking the modifications of topological i-cells [Lienhardt 1994] $(0 \leq i \leq 3)$ of a G-map G can be done by means of tracking the modifications on the darts constituting these i-cells. It consists in calculating which cells contain this dart before and after a high level operation. To do so, a data structure called "tag" is defined. It is a pair of quadruplets of integers related to each dart $b \in G$. The

[2] Let $\{\Pi_0, ..., \Pi_n\}$ be a set of permutations on B. The orbit of an element b relatively to this set of permutations is $\langle\Pi_0, ..., \Pi_n\rangle(b) = \{\Phi(b), \Phi \in \langle\Pi_0, ..., \Pi_n\rangle\}$, where $\langle\Pi_0, \ldots, \Pi_n\rangle$ denotes the group of permutations generated by $\{\Pi_0, \ldots, \Pi_n\}$.

two components of this pair are respectively called "current_tag" and "ancestor_tag", and are defined as:

1. *current_tag*: it corresponds to quadruplet (cur_id_0, cur_id_1, cur_id_2, cur_id_3) and respectively characterizes each i-cell c_i ($0 \leq i \leq 3$) containing dart b by an identifier cur_id_i;

2. *ancestor_tag*: it corresponds to quadruplet (anc_id_0, anc_id_1, anc_id_2, anc_id_3) and respectively characterizes each i-cell, ancestor of c_i, by an identifier anc_id_i (i matches a dimension of c_i).

Figure 6 shows the tags related to the darts constituting an edge before and after an elementary operation of *edge splitting*. For example, quadruplet $(4, 1, 2, 3)$ corresponding to the *current_tag* of dart b_1 in figure 6a, means that b_1 belongs to the 0-cell number 4, to the 1-cell number 1, to the 2-cell number 2 and to the 3-cell number 3. This principle is similar for every other dart.

All the current_tags related to darts belonging to an i-cell c_i have the same identifier cur_id_i. In figure 6a (resp. figure 6c), darts b_1 and b_2 (resp. darts b_3 and b_4) forming 1-cell A (resp. 0-cell S) have the same identifier cur_id_1 1 (resp. the same identifier cur_id_0 8). However, an i-cell gathers a set of identifiers anc_id_i. Indeed, c_i contains ancestor_tags that do not necessarily have the same identifier anc_id_i. Moreover, this set can be empty when all identifiers anc_id_i are set to value ID_NULL (ID_NULL is a particular value which means that an identifier has no value). In figure 6c, darts b_3 and b_4 have ancestor_tags that contain an anc_id_0 set to ID_NULL (represented by the character "-"). The current and ancestor tag structures allow one to associate an identifier cur_id_i (which characterizes each i-cell c_i) with a set of identifiers set_id characterizing the ancestor of c_i.

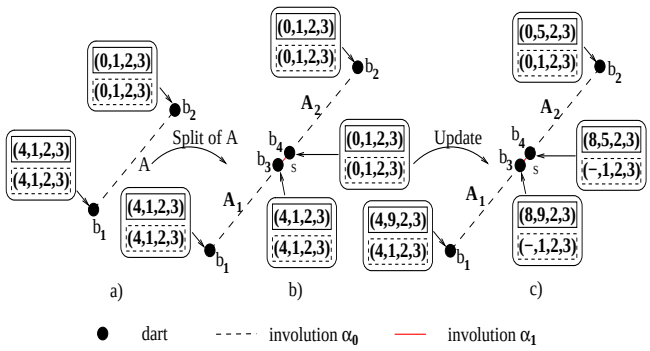


Figure 6: *Edge splitting. For every dart, the current_tags are surrounded with full border, while the ancestor_tags are surrounded with dashed border. a) Darts b_1 and b_2 form edge A before the split. b) A is split in $A_1 = (b_1, b_3 = \alpha_0(b_1))$ and $A_2 = (b_2, b_4 = \alpha_0(b_2))$. Vertex S connecting A_1 and A_2 is made of both b_3 and $b_4 = \alpha_1(b_3)$. b_1 and b_2 save and propagate their tags to the new darts of A_1 and A_2: b_3 and b_4. c) Update of current_tag of b_1 and b_2, and update of both tags associated with b_3 and b_4.*

For each dimension, the identifiers of cells are defined modulo 4 because we work in a 3D space. Thus, we can directly deduce the dimension of the cell from its identifier. More precisely, each i-cell has got an identifier cur_id_i such that $cur\_id_i \bmod 4 = i$. In that way, we have implemented the function $dimension$ defined in section 2.2. Then, we can update the current_tag structure on the darts of figure 6c: edge A_1, created during the split, is characterized by the number 9 (9 *mod* $4 = 1$), and this value is set to identifier cur_id_1 of b_1 and b_3. This principle is the same for edge A_2: it receives the number 5, and this value is set to identifier cur_id_1 of b_2 and b_4. Eventually, vertex S receives the number 8 and this value is set to identifier cur_id_0 of b_3 and b_4.

The next section is dedicated to the implementation of the elementary mechanisms for following up events to Moka during the elementary operations forming the co-refinement high level operation.

3.3 Co-refinement high level operation

Until now, in order to realise a machining operations on an geometric objects, we use classic boolean operations such as the union, the difference or the intersection. The co-refinement is a common denominator for the calculation of an intersection and allows to obtain, after an extraction (this step allows to save only one volume resulting from a 3D subdivision of the co-refinement operation) phase, the same results as the three preceding operations.

The co-refinement operation in dimension 3 consists in generating the spatial subdivision resulting from the intersection between two original 3D subdivisions. Generally, applying co-refinement consists in computing the intersections of faces of the original subdivisions and updating the topology to obtain a valid final subdivision.

Most existing works in this field ([Perrin 2005], [Gardan et al. 2003], [Kitajima and Yamaguchi 1992], [Mantyla and Tamminen 1983], [Ma and Tang 1988]) are essentialy based on applying the boolean operation on two volumes. They let us classify the different parts of an object against the interior or the exterior of another object. In most cases, these methods only allow to process surfacic objects (*ie.* 2D topological objects having 3D embedding) and to build a single volume from the initial objects.

As in the case of boolean operations, many problems can occur when computing intersections. Indeed, the 3D co-refinement algorithm we use is based on a method frequently used in the algorithms associated with boolean operations on surfacic objects, but Guiard [Guiard 2006] has extended this method for 3D grids, with a new algorithm based on the intersections of pairs of faces, and has implemented it with the G-maps model. The use of boolean operations is fundamental in many modelers. Moreover, the important number and the unpredictable behaviour of events which occur during the co-refinement operation makes them difficult, and thus particularly interesting, to track.

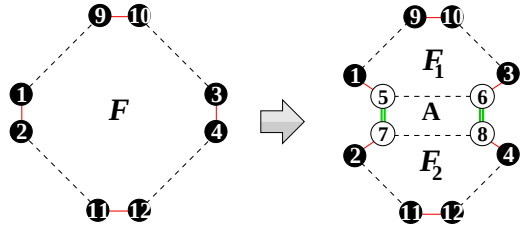
The operation of co-refinement is decomposed in the following way: insertion of a dangling edge into a face, *edge splitting*, *face splitting* and face identification. This set represents only a part of the elementary operations that can be defined in a geometric kernel and particularly in the case of a G-map kernel. However, in order to integrate the mechanisms of events' follow-up, the approach described in this paper is still the same and can easily be extended to the whole set of elementary operations. We present here the necessary functions to describe every elementary operation used in the co-refinement algorithm:

- `Copying_Tag(`b : *Dart*, O : *Orbit*`)`: this function propagates *current_tag* and *ancestor_tag* related to dart b to all darts constituting orbit O.
- `Rep_Anc_ID(`O : *Orbit*, d : *[0..Dim_max]*, id : *ID_Cell*`)`: this function updates the identifier anc_id_d of each *ancestor_tag* related to the darts constituting orbit O with value id (id must pass the condition: id *mod* $4 = d$).
- `Rep_Cur_ID(`O : *Orbit*, d : *[0..Dim_max]*, id : *ID_Cell*`)`: This function has the same role as function `Rep_Anc_ID`. However, it uses the *current_tag* structure.

We only describe the elementary operation "*face splitting*" because the principle is similar for every other elementary operation.

3.3.1 Face splitting

Splitting a face F results in the insertion of an edge A into F. The result of this split is the generation of two new faces incident to F and called F_1 and F_2 (see figure 7). Mechanisms *split_elem* and *crea_elem* relate to this function.



(a) (b)

Before

dart	ancestor_tag				current_tag			
1	4	5	2	3	4	5	2	3
2	4	9	2	3	4	9	2	3
3	8	13	2	3	8	13	2	3
4	8	17	2	3	8	17	2	3
9	12	5	2	3	12	5	2	3
10	12	13	2	3	12	13	2	3
11	16	9	2	3	16	9	2	3
12	16	17	2	3	16	17	2	3

After

dart	ancestor_tag				current_tag			
1	4	5	2	3	4	5	6	3
2	4	9	2	3	4	9	10	3
3	8	13	2	3	8	13	6	3
4	8	17	2	3	8	17	10	3
5	4	-	2	3	4	21	6	3
6	8	-	2	3	8	21	6	3
7	4	-	2	3	4	21	10	3
8	8	-	2	3	8	21	10	3
9	12	5	2	3	12	5	6	3
10	12	13	2	3	12	13	6	3
12	16	17	2	3	16	17	10	3

(c)

Figure 7: *Face splitting. (a) Darts* $\{1, 2, 3, 4, 9, 10, 11, 12\}$ *belong to* F *before the split. (b)* F *is split into* F_1 *(composed of darts* $\{1, 3, 5, 6, 9, 10\}$*) and* F_2 *(composed of darts* $\{2, 4, 7, 8, 11, 12\}$*). Edge* A*, incident to* F_1 *and* F_2*, is made of darts* $\{5, 6, 7, 8\}$*. (c) The content of each tag related to one of these darts at before and after face splitting.*

This elementary operation is performed by:

`SplitFace`(b_1, b_2: *Dart*) $\rightarrow$ Dart: this function splits the face incident to darts b_1 and b_2. It inserts a topological edge between b_1 and b_2 in order to obtain $\alpha_2(\alpha_1(b_1)) = \alpha_0(\alpha_1(b_2))$.

The function `Name_Splitted_Face`, described in algorithm 2, associates the tag structures with the darts constituting F_1 and F_2. `Name_Splitted_Face` is used after `SplitFace` and it takes two darts, b_1 and b_2, incident to face F, as input parameters.

We show the behaviour of algorithm 2 in figure 7. Suppose that darts b_1 and b_2 are numbered 1 and 4 respectively.

The following steps are performed (figure 7c summarizes the whole processs):

1. Every dart of the inserted edge A is processed (let b' be one of these darts). Each dart b' receives a copy of the tags related to dart $\alpha_1(b')$. In figure 7, darts $1, 3, 2$ and 4 respectively propagate their tags to darts numbered $5, 6, 7$ and 8.
2. Because A is a new edge, the edge identifier of each ancestor_tag, anc_id_1, related to the darts belonging to A is initialized with the value ID_NULL. In figure 7, darts $\{5, 6, 7, 8\}$ have their identifier anc_id_1 set to ID_NULL (represented by the character "-").
3. For the same reason, a new value is given to identifier cur_id_1 for each current_tag related to the darts forming A. In figure 7, identifier cur_id_1 of the current_tags of darts $\{5, 6, 7, 8\}$ is set to 21.
4. We assign a new value to identifier cur_id_2 for each dart belonging to F_1. In figure 7, the value 6 is assigned to identifier cur_id_2 for each current_tag of the darts numbered $\{1, 3, 5, 6, 9, 10\}$.
5. The same process is done on face F_2. In figure 7, identifier cur_id_2 of each current_tag related to the darts numbered $\{2, 4, 7, 8, 11, 12\}$ receives the value 10.

Steps 2 and 3 correspond to the mechanism *crea_elem* applied on A. Indeed, these steps relate the identifier 21 characterizing A to an empty set of identifiers. Steps $1, 4$ and 5 implement the mechanism *split_elem* (see section 2.2) applied on F. The propagation (resp. the update) of tags performed during step 1 (resp. during steps 4 and 5) enables cells F_1 and F_2 to inherit the ancestor of cell F: the identifier anc_id_2 2 (resp. to be characterized by identifiers 6 and 10).

Algo. 2 Function `Name_splitted_face`

Data: b_1 and b_2: two darts representing the parameters of function `SplitFace`.

Result: Allocation of valid tags to the darts forming the faces F_1 and F_2.

begin

1 **foreach** $b' \in \ < \alpha_0, \alpha_2, \alpha_3 > (\alpha_1(b_1))$ **do**
  `Copying_Tag`($\alpha_1(b')$,<> ($\alpha_0(b')$));
2 `Rep_Anc_ID`($< \alpha_0, \alpha_2, \alpha_3 > (\alpha_1(b_1)), 1,$ *ID_NULL*);
3 `Rep_Cur_ID`($< \alpha_0, \alpha_2, \alpha_3 > (\alpha_1(b_1)), 1,$ *Get_New_ID*(1));
4 `Rep_Cur_ID`($< \alpha_0, \alpha_1, \alpha_3 > (b_1), 2,$ *Get_New_ID*(2));
5 `Rep_Cur_ID`($< \alpha_0, \alpha_1, \alpha_3 > (b_2), 2,$ *Get_New_ID*(2));

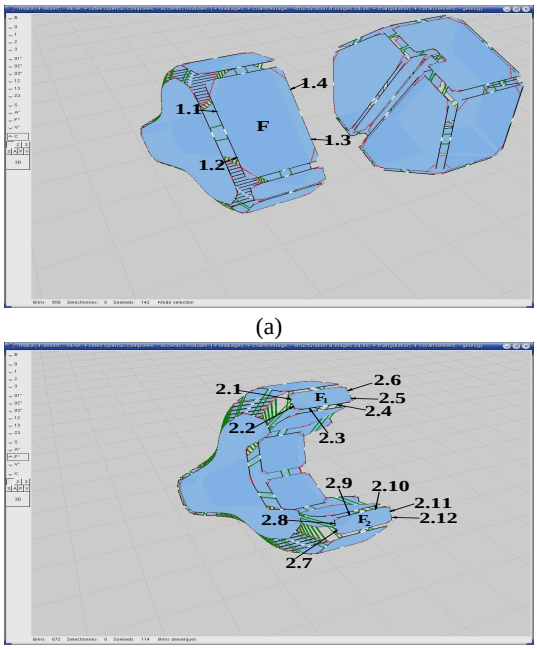
end

3.4 Results

In this section, we present the results obtained after the application of follow-up event mechanisms on the G-map model; we have successfully experimented our method on numerous models built using different operations, including classical boolean operations. The following example (see figure 8) has been created with boolean operations. Considering the high number of darts (872) involved in this figure, we can not show the tag structure related to each dart. Therefore, we only show some specific darts. The other darts can easily be deduced by definitions and examples introduced in sections 2.2 and 3.3.

In particular, the contents of tags (see Tab. 2a and 2b) obtained after application of the difference boolean operation (see figure 8) show that:

- Edges A_3 and A_4, respectively represented by darts $\{2.3, 2.4\}$ and $\{2.9, 2.10\}$, have just been created. Indeed,



(a)

(b)

Figure 8: *G-map model of spanner resulting from high level operation (boolean difference). Both figures a) and b) are Moka screenshots which show spanner in the top view. The darts are symbolized by half-edges, and involutions α_0 (resp. α_1, α_2) are symbolized by white (resp. red, green) lines.*

(a) Before high level operation

Dart	Ancestor_tag	Current_tag
1.1	$(16, 17, 6, 3)$	$(16, 17, 6, 3)$
1.2	$(4, 17, 6, 3)$	$(4, 17, 6, 3)$
1.3	$(20, 25, 6, 3)$	$(20, 25, 6, 3)$
1.4	$(24, 25, 6, 3)$	$(24, 25, 6, 3)$

(b) After high level operation

Dart	Ancestor_tag	Current_tag
2.1	$(16, 17, 6, 3)$	$(16, 197, 98, 35)$
2.2	$(-, 17, 6, 3)$	$(160, 197, 98, 35)$
2.3	$(-, -, 6, 3)$	$(160, 285, 98, 35)$
2.4	$(-, -, 6, 3)$	$(148, 285, 98, 35)$
2.5	$(-, 25, 6, 3)$	$(148, 201, 98, 35)$
2.6	$(24, 25, 6, 3)$	$(24, 201, 98, 35)$
2.7	$(4, 17, 6, 3)$	$(4, 133, 74, 35)$
2.8	$(-, 17, 6, 3)$	$(176, 133, 74, 35)$
2.9	$(-, -, 6, 3)$	$(176, 297, 74, 35)$
2.10	$(-, -, 6, 3)$	$(172, 297, 74, 35)$
2.11	$(-, 25, 6, 3)$	$(172, 145, 74, 35)$
2.12	$(20, 25, 6, 3)$	$(20, 145, 74, 35)$

Table 2: *current_tags and ancestor_tags matching the high level operation shown on figure 8.*

their identifier anc_id_1 is equal to ID_NULL.

- Vertices S_1, S_2, S_3 and S_4, respectively represented by darts $\{2.2, 2.3\}$, $\{2.4, 2.5\}$, $\{2.8, 2.9\}$ and $\{2.10,\ 2.11\}$ (among others), have also just been created: their identifier anc_id_0 is equal to ID_NULL.
- Face F, constituted by darts $\{1.1, 1.2, \ldots, 1.3, 1.4\}$ and characterized by the identifier cur_id_2 6 (see Tab. 2a), is split. Faces F_1 ($cur\_id_2 = 98$) and F_2 ($cur\_id_2 = 74$) result from this split. Therefore, every dart being part of F_1 and F_2 (resp. $\{2.1, 2.2, 2.3, \ldots,\ 2.4, 2.5, 2.6\}$ and $\{2.7, 2.8, 2.9, \ldots, 2.10, 2.11, 2.12\}$) has its identifier anc_id_2 equal to 6.
- Edge A_1 (resp. edge A_2), constituted by darts $\{1.1, 1.2\}$ (resp. $\{1.3, 1.4\}$) and characterized by identifier cur_id_1 17 (resp. 25), is split in edges $A_{1.1}$ ($cur\_id_1 = 197$) and $A_{1.2}$ ($cur\_id_1 = 133$) (resp. $A_{2.1}$ and $A_{2.2}$: $cur\_id_1 = 201$ and $cur\_id_1 = 145$). Therefore, darts $\{2.1, 2.2, 2.7, 2.8\}$ forming both $A_{1.1}$ and $A_{1.2}$ have their identifier anc_id_1 equal to 17, and darts $\{2.5, 2.6,\ 2.11, 2.12\}$, forming both $A_{2.1}$ and $A_{2.2}$, have their identifier anc_id_1 equal to 25.

The bulletin board (see Tab. 3), sums ups all those events. Moreover, we can indicate:

- the split of edges A_1 and A_2 (resp. of face F) characterized by identifiers anc_id_1 17 and 25 (resp. by identifier anc_id_2 6) ;
- the creation of vertices (resp. edges) S_1, S_2, S_3 and S_4 (resp. A_3 and A_4) characterized by identifiers cur_id_0 160, 148, 176 and 172 (resp. by identifiers cur_id_1 285 and 297);
- other events have occured during this high level operation (table 3 only shows some of them). We can quote the deletion of vertex (resp. of edge and of face) characterized by identifiers anc_id_0 76 (resp. by identifiers anc_id_1 109 and anc_id_2 54) or the creation of cells (vertex and edge) characterized by identifiers cur_id_i 281 and 160.

Our work is currently focused on the completeness of the generated events; our method has been implemented regardless of complexity. As explained in Sections 2.2 and 2.3, there are two global traversals of the model (in order to associate and to retrieve tag structures from darts at the beginning and at the end of the high-level operation). These traversals can be optimized to process only the tag structures which have been modified by the operation.

4 Conclusion

In this paper, we have proposed a generic model allowing a bulletin board to be generated during a constructive operation (i.e. supplied by the API of a geometric modeling system). This bulletin board contains the complete list of events which reflects the evolution of cells of any dimension in a geometric system based on a topological model. The method uses some mechanisms for event (creation, deletion, split, merge) follow-up. More precisely, any high level operation is decomposed into a finite set of elementary operations. Each of them is associated with one or many event follow-up mechanisms. The model is generic because, on the one hand it is exclusively based on mechanisms defined independently of any geometric model, making different types of implementation possible, and, on the other hand, because these mechanisms are defined in a generic way for any dimension.

Our implementation is based on the model of generalized maps (G-maps). By using a "tag" structure associated with each dart of a G-map, this system characterizes, through "current_tags", each cell of the current geometric model; and links through "ancestor_tags" these cells to the other cells which existed at the beginning of the high level operation. Next, the system assigns the right tag structure to each dart.

We have successfully experimented this principle on numerous models built with different operations; the co-refinement high level

Starting cells	Ending cells	Event type
{61}	{61}	no modification
{}	{281}	edge creation
{}	{160}	vertex creation
...	...	...
{}	{**148**}	**vertex creation**
{}	{**160**}	**vertex creation**
{}	{**176**}	**vertex creation**
{}	{**172**}	**vertex creation**
{**25**}	{**145,201**}	**edge split**
{**17**}	{**133,197**}	**edge split**
{**6**}	{**74,98**}	**face split**
{}	{**285**}	**edge creation**
{}	{**297**}	**edge creation**
...	...	...
{76}	{}	vertex deletion
{109}	{}	edge deletion
{54}	{}	face deletion

Table 3: *Bulletin board matching the high level operation shown on figure 8.*

operation is a good case study because it is complex and often used in the geometric modeling domain. The last step of the bulletin board generation process is based on the traversal of darts and the recognition of events. Our implementation enables us to realize generic mechanisms for event follow-up. It is a method useable for many and various practical application domains. It also permits the topological operations to be separated from the bulletin board generation; therefore, while topological operations are always the same for a given geometric operation, there can be several ways to generate the bulletin board, with respect to the events this bulletin board should collect.

The next step of this research is to study the possibility to test mechanisms which have been formally and independently defined in order to do a high-level experimentation and to compare the results obtained with the events contained in the bulletin board. Finally, the genericity of tag structure should allow our approach to be extended to space-time ($4D$) modeling for animation.

References

ARMSTRONG, C., BOWYER, A., CAMERON, S., CORNEY, J., JARED, G., MARTIN, R. R., MIDDLEDITCH, A., SABIN, M., AND SALMON, J. 2000. *A Geometric Interface for Solid Modelling Specification and Report*. Information Geometers Ltd.

BRISSON, E. 1993. Representing geometric structures in d dimensions: topology and order. *Discrete and Computational Geometry 9*, 387-426.

BRUNIER-COULIN, D., PASCAL, D., AND MCCARTHY, D. 2000. OCAF: Open Cascade Application Framework. *Matra Datavision*.

GARDAN, Y., MINICH, C., AND PERRIN, E. 2003. Boolean operations on feature-based models. *Journal of WSCG Winter School of Computer Graphics 1-3*, 11.

GUIARD, N. 2006. Construction de modles gologiques 3D par co-raffinement de surfaces. *PhD thesis, Ecole des mines*.

KITAJIMA, K., AND YAMAGUCHI, M. 1992. A shell oriented boolean set operations algorithm suited for the b-reps based on boundary edge loops. *Systems and computers 23*, 6.

KRIPAC, J. 1995. A mechanism for persistently naming topological entities in history-based parametric solid models (topological id system). *Proceedings of Solid Modeling95*, 21-30.

LIENHARDT, P. 1994. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications 4(3)*, 275-324.

MA, D., AND TANG, R. 1988. Realizing the boolean operations in solid modeling technique via directed loops. *Computer and Graphics 12*, 3/4.

MANTYLA, M., AND TAMMINEN, M. 1983. Localized set operations for solid modeling. *Computer and Graphics 17*, 3.

MARCHEIX, D., AND PIERRA, G. 2002. A survey of the persistent naming problem. *7th ACM Symposium on Solid Modeling and Applications*.

PERRIN, E. 2005. Opérations booléennes: trente années d'un algorithme toujours au cœur des systèmes de CAO. *Revue internationale d'ingnierie numérique 1*, 265-289.