34

Considerations on Topological Naming

Rémi Lequette
Matra-Datavision, Direction Scientifique
31, avenue de la Baltique 91954 Les Ulis Cedex France
Tel: (33-1)69 82 24 00 Fax: (33-1)64 46 02 19
Email: r-lequette@paris1.matra-dtv.fr

Abstract

Topological naming is a mechanism giving persistent identities to entities in a geometric model (faces, edges, vertices). This mechanism is used to attach attributes and feature definitions in a parametrized solid modeler. In this paper we first discuss the semantic of topological naming and we show that different interpretations can be found depending on application. Then we study the principles of solutions which are labeling of entities and pattern matching. We identify the requirements on the geometric modeling engine to implement topological naming. We present and compare three existing algorithms.

Keywords

Topological Naming, Parametric Solid Modeling, BRep, CADCAM

1 INTRODUCTION

Topological naming refers to the problem of identifying parts in a geometric model and tracking them when the model is modified. Traditionally this problem was related to attributes, like a face color in a solid model, and the propagation of the attributes, the solid is cut and the face is modified but keeps its color. The emergence of high-level editing tools in applications using geometric modeling which regenerate the model after an edit has obliterated this usual solution. A specific example is given by parametric modelers (Hoschek and Dankwort, 1994) which store a model as a sequence of modeling operations, those modelers use extensively topological naming because each operation is attached to the previous state of the model. For example a blend is attached to an edge and if a previous operation in the model is edited the edge must be correctly identified to be blended again.

The emergence of geometric modeling libraries (geometric engines) which are independent from applications raises the question of which service should be provided by those engines to implement topological naming.

Many commercial CADCAM system implement a solution for topological naming and two recent papers (Capoyleas, Chen, Hoffmann 1995,1996), (Kripac 1995) have been issued by the academic community.

M. J. Pratt et al. (eds.), *Product Modeling for Computer Integrated Design and Manufacture*

In this paper we investigate the semantic of topological naming and the principles of solutions. We discuss three implementations, the two papers mentioned above and the work done at Matra-Datavision.
In the first part we investigate more precisely the need for topological naming, namely attribute management and high-level editing. We discuss the semantic of topological naming, i.e. what is really expected, and we show that there is no clear definition because the semantic is driven by the applications and sometimes topological naming may be used in the wrong place.
In the second part we examine the principles underlying solutions, they are labeling of topological entities while they are created and modified and pattern matching to resolve ambiguities using adjacent entities and geometric clues. We consider how the geometric engine can be used to provide those solutions.
In the third part we discuss three existing algorithms. Capoyleas, Chen, Hoffmann is used in the ERep model which is a textual description of parametric solid models. Kripac uses a history graph of faces. Matra-Datavision implementation is close to Kripac's and uses original features of the CAS.CADE geometric engine API.

2 THE NEED FOR TOPOLOGICAL NAMING

To give the most general definition let us imagine a model used by a CADCAM application (or any other software application using geometric modeling). This model can be divided in two parts, the geometric model which describes a set of 3d points and the rest of the model which contains the geometry related data. Those application data may themselves be separated in two parts, data which have an influence on the geometry (like constraints, parametric graph, dimensions, ...) and data which are attached to the geometry (like colors, materials, ...), those data are usually called attributes.
Of course this division is theoretical and the data structure of the application may not show such a clear distinction. However if the application is well designed, or if the application uses an all-purpose geometric engine, this division will certainly be apparent in the architecture.
Parts of a geometric model are defined using topological features such as faces or edges, so to be able to attach information to them the geometric model must contain them explicitly. This implies that the geometric model must contain a Boundary Representation (BRep), even if it is not the primary representation.
The geometric model must contain a BRep to give access to faces, edges or vertices but the model does not have to be a solid model. The models can be made of solids, shells, wires, depending on the application needs. Topological naming is mainly used in parametric solid modeling but there is no doubt that it is also a very important issue for history-based surface modeling.
Attributes are technical data attached by the application to parts of the geometry. They can be attached to vertices, edges, faces or solids, as for example mesh density in an FEM preprocessor, or surface finish and material in an NC programming application. Attributes may also be attached to group of entities, for example a loop of edges, or a set of faces. This can even happen implicitly because the modeler has represented a logical face by several physical

faces, this is common with curved faces like cylinders. The problem of the application is to reattach and update features and attributes when the geometry of the model is modified.
BRep is a low-level representation of the geometry and usually applications do not provide users with tedious low-level operations to edit the geometry specially in solid modeling where they can corrupt the model. Applications provide high-level modeling operations using the algorithms in the geometric modeler. To facilitate editing of the model at a higher level the application usually records in the model the arguments of the algorithms to let the user modify them. For example if the application is used to create a box it may register the three dimensions, and for a blend it may register the blended edge and the radius of the blend. Many current Design applications provide parametric modelers which record all the operations used to build a model and replay them when some parameters are modified. Topological naming is extensively used in those application to record the geometric arguments of algorithms, for example the edge to be blended. If parameters of operations performed before the blend are modified the application needs to identify the new edge to be blended. We can say that applications uses topological naming to update the geometric arguments of high-level editing operations and to update technical attributes.

Semantic of Naming

To provide sound implementations it would be useful to have a precise definition of the problem to solve, i.e. a correct semantic of topological naming. Unfortunately this is not an easy thing to do in full generality. For the reasons we are going to present now topological naming is not a well-defined mathematical problem.
The main reason is that this semantic is part of the application, and ultimately lies in the mind of the user who selects the entities to be named and have his own idea on the corresponding entity after a modification. It is well known that capturing this kind of meaning in an algorithm is not an easy task, and in the case of topological naming the main source of trouble is that even the human user may be unable to find the corresponding entity after major modifications. Applications, specially Design applications, allow very flexible editing of the models which may change dramatically their appearance and make topological naming meaningless. For example one can change the profile of a swept solid so that the topology of the solid is completely different. This problem should not be exaggerated because it makes sense that topological naming has meaning only for "topologically close" modifications, but it is impossible to define precisely what is "topologically close".
Many problems are introduced in topological naming because of "cardinality change", i.e. when the number of entities changes. Either this number is reduced, edges or faces disappear, two edges or two faces are merged into one because they have the same geometric support, or the number is increased, two surfaces having one intersection curves now have two, a face or an edge is split into two or more parts. The semantic of the naming depends heavily of what the application does with the entity, and the cardinality change may be harmless or very annoying.
We claim that many problems are inherent to the use of topological naming because using a set of topological entities do not carry the correct semantic information. Let us explain what we mean with two examples.
The first example concerns dimensions which are used for locating features or associative dimensioning of drawings. Dimensions should not be attached to topological entities which

carry "too much" information. In Figure 1 (a) dimension D3 is attached to face F which is the result of merging two faces. Suppose now that dimension D2 is modified as in Figure 1 (b), there is an ambiguity to regenerate the slot because two faces may be used as reference for D3. The applications should avoid topological naming in this kind of situation and attach dimensions to geometry (axis, plane, cylinder) which is constrained by other dimensions. This kind of approach is not very common today but would be much more consistent with traditional engineering dimensioning.

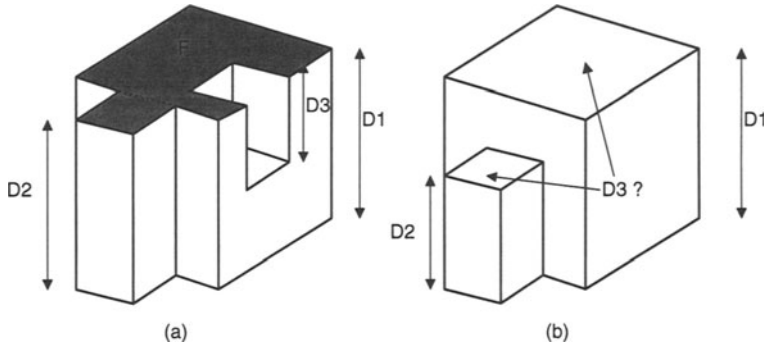

Figure 1 Ambiguous dimensioning through face merging

In the previous example topological entities carried to much information for dimensioning, in the following example, see Figure 2, we could say that they do not carry enough information. In this example we want to blend all the edges between the rib and the rest of the model, if the edges to be blended have been named, then when the rib is moved to an other location it may be impossible to find the new edges to blend because the adjacent faces may be different. Only the algorithm creating the rib is able to describe accurately those edges as the edges between the rib and the rest of the model.

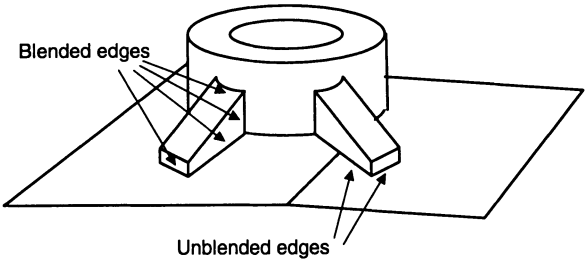

Figure 2 Incorrect naming for edges connecting a rib with the rest of the model

3 PRINCIPLES OF SOLUTIONS

In this section we examine the general principles for topological naming implementations. The first technique is to label the entities according to the way they were constructed. Labeling may not be sufficient because entities are split or merged and the one to one relation with the label is lost. Another technique must be provided to resolve ambiguities, this technique is topological and geometrical pattern matching. We will examine both techniques and put a special emphasis on the functionality which must be found in the geometric modeler to implement them.

Labeling

The first way to name topological entities is to label them when they are constructed. For example in a circular blind hole two faces are created which can be labeled : the "bottom planar face", and the "lateral cylindrical face", see Figure 3. Actually the cylindrical face may be represented by more than one physical face in some geometric modelers, but this subdivision should have no influence on naming and illustrate the point that named faces should be understood as group of faces (shells), and named edges as group of edges (wires).

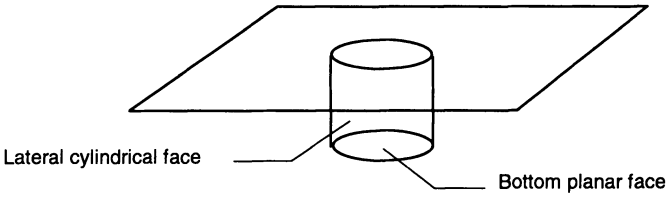

Figure 3 Simple labeling of the faces in a blind hole

This simple enumerated labeling scheme does not cover all the operations used to create geometry, in other situations one must add topological entities to the label, those topological entities are part of the arguments of the construction algorithm. For example when a model is created by sweeping a profile the lateral faces and edges can be labeled with the edges and vertices of the profile, the faces and edges at the beginning and end of the sweeping path (if this path is not close as in a 360° rotational sweep) must combine an enumerated label ("begin" or "end") and a topological entity. Figure 4 illustrate labeling in a linear sweep.

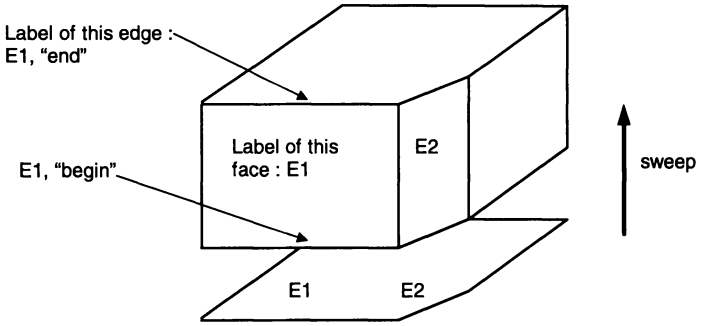

Figure 4 Labeling in a sweep

The same kind of scheme can be applied to all the feature operations which are based on sweeps. Blend and chamfers can be also labeled from the edge which was chamfered or blended. If the modeler provides shelling operations the faces in the shell can be labeled from the faces and edges of the shelled objects. When using advanced modeling operations, for example in surface modeling, it may be necessary to use more than one entity in the arguments to label an entity in the result, for example for complex sweeping operations involving a path and a profile, a face in the result must be labeled from an edge in the profile and an edge in the path.

When an entity is labeled in terms of other entities the application must also label the original entities in case the arguments of the operations are modified. Ultimately this lead to labelling the edges in the profiles and paths which are used in the operations.

Some geometrical engines, actually most of them, do not give access to the origin of each entity in the result of a construction. There are ways to work around this limitation but they may use undocumented features of the engine and be sensitive to modifications. In some cases, like the hole example, a search may be performed in the faces of the object to find the planar or cylindrical face. In sweeps many engines use the edges of the swept profile in the result so one can search those edges in the faces of the result. Blends and chamfers can be performed edge by edge to track the correspondence between edges and faces. For more complex operations like shelling or advanced sweeps it is difficult to find a good solution.

Once we have covered the labeling of topological entities at creation we must consider the operations, like the booleans, which modify entities, deleting, splitting or merging them. The geometric engine must provide a way to know which entities are merged or split in order to assign them their labels. When entities are merged an entity may end up with multiple labels, and when an entity is split multiple entities may end up with the same label. To trace the labels through entities modifications we need some support from the geometric engine as it will be difficult to find workarounds. This support can be provided through two forms, either the geometric engine provides entity attributes and propagates those attributes during the operations when entities are modified, or the API describes explicitly the modifications. If the second solution is available we may notice that there are no conceptual differences between construction and modification of entities, we can consider that entities created by merging or splitting are created from previous entities just like entities created by sweeping of blending.

Our last concern is the representation of the labels in the data structure of the application, a label consists of an identification of the construction operation which created the entity, an enumerated label which has a meaning for this operation ("begin", "end" for example), and labels of entities which have generated the entity. The labels can be stored with the topological entities as attributes supported by the geometric engine. An other solution is to store the labels in a graph structure which records the origin of the entities. Accessing the label (or labels) of an entity tells the application which modeling operations generated this entity, so it is a convenient way to determine which operation to edit after a graphical selection of the entities.

Pattern Matching

An implementation of topological naming can also use pattern matching or graph isomorphism in the topological graph, instead of saying "the lateral face" of the blind circular hole we could say "the face which is adjacent to two other faces, each adjacency through only one edge", describing the topological configuration around the face. A pure topological description cannot describe uniquely the entities because topological graphs are highly symmetrical, for example it is impossible to characterize a face of a cube using only adjacency information. Some other information must be used to characterize the topological entities, so the pattern matching techniques are best used as a complement to labeling to remove ambiguities. For example in Figure 5 the end face of a linear prism has been split in two faces F1 and F2, both faces have the same label, to make a distinction we can add an adjacency information saying that F1 is adjacent to the faces labeled from the edges E1 and E2.

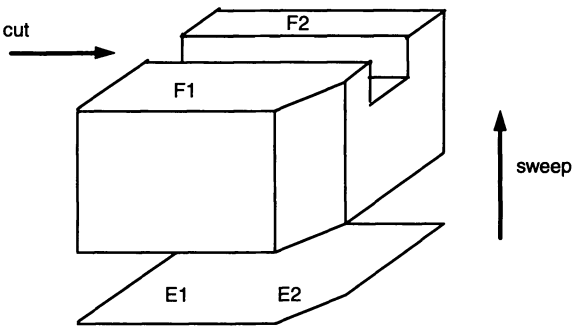

Figure 5 Using adjacency to solve ambiguities

Capoyleas, Chen and Hoffmann have made a complete study of this problem and they have correctly pointed out that sometime the first adjacency of an entity is not enough to characterize it uniquely and the name must include entities adjacent to the first adjacent entities. They have also noted that in some symmetrical situations there is no solution using only adjacency and geometric information must be added, they suggest using the orientation of the edges and the orientation of the swept surfaces.

4 IMPLEMENTATIONS

Topological naming is present in most commercial CAD software and Capoyleas, Chen and Hoffmann showed the flaws in some of them. We will now discuss two implementations : (Capoyleas, Chen and Hoffmann 1995) and (Kripac 1995) and we will present the one we have developed at Matra-Datavision.

Capoyleas, Chen and Hoffmann studied topological naming in the context of the ERep project (Hoffman and Juan 1993) which is a textual high-level representation for parametric solid models. An ERep file describes a solid in term of successive feature operations using variational profiles, the features used are protrusion, cuts, chamfers and blends. Topological naming is used to record the arguments of the features, edges to blend, faces where a cut starts and ends, etc.. and to record the entities where the variational profiles are attached to the model. The topological naming uses a labeling scheme completed with adjacency information, the labels are stored in the attributes of the geometric engine (ACIS). Labels are defined for the supported features which are implemented using sweeping and boolean operations. The authors do not explain how they identify which faces are generated from which edges in sweeps but we assume they use the method described in the previous chapter, they do not mention including other construction operations as shelling. A strong point of this work is the study of adjacencies and geometric clues to remove ambiguities. The ERep representation has a lot of appeal for reference model description because it is a language with a well defined semantic both for operations and naming so there is theoretically no ambiguity on the model. It is an unevaluated model and a BRep must be attached to it for practical use, but this BRep does not have to be stored and ERep is independent from the geometric engine.

Kripac presents a different approach based on a face history graph. First we must notice that he only label faces, edge and vertex naming is done via face adjacency. Labeling only faces can save time and space assuming that the number of edges which need to be named is small. Kripac does not explain how initial label are assigned to new faces, so his scheme is quite general regarding the construction operations but needs to be completed. The originality of Kripac's work is to store the faces in an history graph, each node of the graph is a face in the final model or in an intermediary state of the evaluation. Edges of the graph connect the faces to the faces which generated it through splitting or merging. So instead of propagating initial labels on faces this method labels each face with its direct ancestors. The great advantage of this method is that the labeling of a face does not need to go to the origin but can use an intermediary state of the face which is present in the model before and after the modification. Let us illustrate this point with an example, in Figure 6 three modeling operations are used, first a block is constructed, then a cut is made and face F1 is split in F2 and F3, third a hole is punched in face F3 which becomes face F4. Suppose now that we are interested in naming face F4 and that we move the location of the hole. If we use labels, the label of F4 is inherited from F1 which was labeled by the block primitive construction, so we must add adjacency information to the name because F2 and F4 have the same label in the final object.

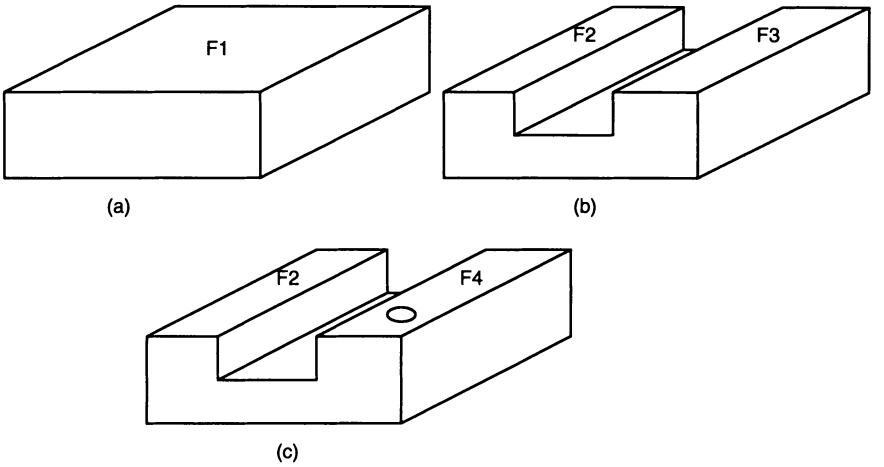

Figure 6 Three steps in a model construction: (a) block (b) cut (c) hole

Now let us consider on Figure 7 the two history graphs of the model before and after the modification (moving the hole), the face F4 is F4' in the new model. Kripac's algorithm will first try to use F3 the first ancestor as a label for F4, and as F3 exists also in the new graph it will immediately assign F4' as the new entity for F4, if F3 were not present in the graph (if the cut was modified) F1 would have been used as a label and adjacency information would have been added.

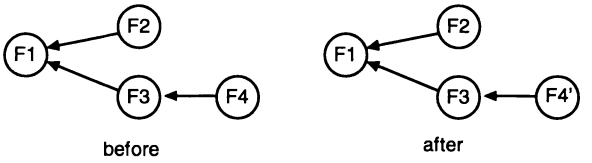

Figure 7 History graph before and after modification

This approach has two consequences, first all the intermediary states of the model are kept to access their faces, this can consume a lot of memory space unless the geometric engine can handle it, keeping intermediary state will speed up reevaluation of the model because it does not have to start from the beginning but only from the first edited operation. The second consequence is that the description of the model before the modification and the face history graph must be kept during the evaluation of the modification because the naming needs access to the graph before and after the modification. We may note that in this situation the pattern matching parts of the names are not kept permanently but they are generated on-the-fly during the reevaluation. If this kind of naming is used the BRep of the model and the history graph must be stored with the high-level description of the model.

At Matra-Datavision we have implemented a topological naming scheme for the Euclid CADCAM software, the geometric engine is provided by the CAS.CADE Software Factory (CAS.CADE 1996). Our approach is very similar to Kripac's as we use an history graph to store the evolution of entities and their original labels. We store in the graph the result of merge and splits and also the relation between entities through construction like an edge swept into a face. A strong point of the CAS.CADE geometric engine is the ability to keep all the intermediate states of the model with the minimum overhead because topological entities which are common between two geometric models are shared.
The most innovative feature used in this implementation is the CAS.CADE API which allows an easy access to information used to label entities and to build the history graph. The CAS.CADE API provides user-level access to this information, this access may be specific to an algorithm, for example the sweep, or generic between algorithms, for example the split and merge information. Having a generic way to access algorithms information helps to incorporate new operations in the naming scheme, they just have to provide the generic information.

5 CONCLUSION

In this paper we discussed topological naming, an important issue in application using parametrized geometric modeling. We saw that applications must carefully examine the semantic of their operations before implementing topological naming. Implementation of topological naming uses two techniques, labeling of topological entities and pattern-matching using adjacency and geometric clues. To implement labeling the geometric engine must provide information about the topological entities creation and modification in algorithms. We discussed the implementations of (Capoyleas, Chen and Hoffmann) and (Kripac) and the one developed at Matra-Datavision using the CAS.CADE geometric engine. Further works regarding topological naming should discuss the semantic issue by proposing other ways to label entities. A promising path could be to combine naming with feature recognition.

6 REFERENCES

Capoyleas, V. Chen, X. Hoffmann, C.M.(1996) Generic naming in generative, constraint-based design *Computer-Aided Design* Vol. 28 pp. 17-26

CAS.CADE (1996) Geometry and Topology User Manual Matra-Datavision

Chen, X. Hoffmann, C. M., (1995) On Editability of Feature Based Design *Computer-Aided Design* Vol. 27 pp. 905-914

Hoffmann, C. M. Juan, R. (1993) ERep, an editable, high-level representation for geometric design and analysis, in *Geometric Modeling for Product Realization* (ed. Wilson, P.R. Wozny, M.J. and Pratt, M.J.) North-Holland

Hoschek, J. and Dankwort, W. (1994) Parametric and Variational Design *Teubner Verlag*

J. Kripac (1995) A mechanism for persistently naming topological entities in history-based parametric solid models, in *Proceedings of the 3rd ACM Symposium on Solid Modeling* (ed. Hoffmann, C. M. and Rossignac, J.R.) Salt Lake City UT, ACM Press