

Proceedings of the XXVII Iberian Latin American Congress on Computational Methods in Engineering September 3 to 6, 2006 - Belém, Pará - BRAZIL.

BOOLEAN OPERATIONS ON NON-MANIFOLD AND B-REP SOLIDS FOR MESH GENERATION

Marcos Chataignier de Arruda Luiz Fernando C. R. Martha Antônio Carlos de O. Miranda

Antonio Carlos de O. Miranda
arruda@tecgraf.puc-rio.br
lfm@tecgraf.puc-rio.br
amiranda@tecgraf.puc-rio.br
Grupo de Tecnologia em Computação Gráfica (Tecgraf) & Departamento de Engenharia
Civil, Pontificia Universidade Católica do Rio de Janeiro (PUC-Rio)
Av. Marquês de São Vicente, n.225 – Gávea, 22453-900, Rio de Janeiro – RJ – Brazil
William Wagner M. Lira
william@ctec.ufal.br
Laboratório de Computação Científica e Visualização, Centro de Tecnologia, Universidade
Federal de Alagoas, Maceió – AL – Brazil

Abstract. In a solid modeler, one of the most powerful tools to create three-dimensional objects with any level of geometric complexity is the application of Boolean set operations. They constitute intuitive and popular ways of combining solids, based on operations applied to sets. The main types of Boolean operations commonly applied to solids are: union, intersection and difference. In order to assure that the resulting objects have the same dimension as the original objects, without loose or dangling parts, a regularization process is usually applied after a Boolean operation. To regularize means to restrict the result in a way that only filling volumes are allowed. In practice, regularization is performed by classifying the topological elements and removing internal or lower dimensional structures. However, in many engineering applications, the adopted geometric model may contain idealized internal parts, as in the case of multi-region models, or lower dimensional parts, as in the case of solids that contain dangling slabs, which are represented as zero-thickness surfaces in the model. The objective of this work is the development of a generic algorithm that allows the application of Boolean set operations in a geometric modeling environment applied to finite element mesh generation. This environment adopts a non-manifold B-Rep representation that considers an undefined number of topological entities (group concept), works with objects of different dimensions, and works with objects not necessarily plane or polyhedral (parametric curved surfaces). The present paper discusses the implementation of this algorithm in a preexisting geometric modeler named MG, using the concept of object oriented programming and keeping the user interface simple and efficient.

Keywords: Geometric Modeling, Non-manifold Solids, Boolean Operations, Boundary Representation, Mesh Generation

1. INTRODUCTION

In geometric modeling for finite element analysis, the mesh is generated based on the geometric description of the model's domain. Therefore, the creation of a geometric model that is able to reproduce real engineering objects that need to be studied is a crucial step within the set of procedures that have to be done in order to perform a complete analysis using the finite element method (FEM).

There are several approaches used by different modelers that deal with geometric modeling to make sure that a large number of models based on real objects can be generated. CSG-based systems use rigid motion transformations and Boolean operations directly to combine basic primitives creating more complex solids. These systems do not save explicit information related to the solids' boundaries, but rather work with the concept of *set of points* in Euclidian space. Systems based on the B-Rep type of solid representation save explicit information about the boundaries of solids and the adjacency relationships between the topological entities of various dimensions that constitute these boundaries (e.g. vertices, edges and faces) (Hoffmann, 1989).

Using Boolean set operations (union, intersection and difference) to combine solids represented by boundaries (B-Rep) is not a trivial task, especially when dealing with curved primitives. The most common problems are related to accuracy, robustness and efficiency.

Numerical inaccuracies are a major problem in any system that has to manage simultaneously geometric and topological information. Since these systems must use floating-point numbers to save coordinates of points, parametric or implicit equations of curves and surfaces, and to perform translations, rotations and other kinds of transformations applied to primitives, they are prone to errors due to the lack of precision in some operations. These errors, unfortunately, might lead to serious problems like crashes and impossible output information. Many solutions have already been proposed to deal with this problem, based on diverse concepts such as the use of model-based or machine tolerances (Shewchuk, 1997), interval arithmetic and the exact computation paradigm (Keyser *et al*, 2000). Until today, however, no system was completely able to solve all kinds of problems related to numerical inaccuracies in environments where curved primitives are allowed.

Robustness is a more relative concept, since it depends on the domain established for the problem in focus. The more restrictions are imposed on the operations used to create objects one wants to model, the more robust the modeler turns out to be. Creating a robust modeler without imposing restrictions on input and output objects and on the operations the user can perform with these objects is a very difficult task, and still remains a challenge to the developers of geometric modeling systems.

Efficiency might not seem to be a major problem when simple models are being constructed. But for complex models it can be a determinant factor for choosing between one or another geometric modeler and between one or another kind of solid representation. Modelers that are meant to deal with complex models containing non-linear primitives and a large number of objects must be implemented using optimized algorithms, which may take a reasonable amount of time to perform the necessary operations to generate the complete model.

There is still another problem when Boolean operations are applied to solids in a 2manifold environment: they do not form a closed set of operations. This means that even when the input data is composed by 2-manifold solids, it is not guaranteed that the result of the application of a Boolean operation over theses solids will be a valid solid representable in the same domain as the original solids. And even when it is, some intermediate stages may not be so. Depending on the way the modeler deals with these intermediate data, it might cause the system to abort the whole operation. Non-manifold representations are one way to deal with this kind of problem. A modeler with a *non-manifold* environment allows the Boolean operations to form a closed set of operations (Weiler, 1986). In this kind of representation, solids with multiple regions (internal interfaces) and degenerate portions (shells, wires) are representable in a consistent way. Nevertheless, the implementation of a system that provides a *non-manifold* environment is a very complex task.

The focus of the work here described is the development of an algorithm to perform Boolean operations in *non-manifold* environments containing curved primitives of any type and dimension. The main idea was to adapt the basic concepts of other previously implemented and tested algorithms in order to make them as general as possible. By making them general we mean making them applicable to any kinds of objects with any shape or dimension, including dangling and loose parts.

Doing this was not a trivial task. We would like to stress that the algorithm proposed in the present paper has no intention to be the only possible way to deal with the combination of *non-manifold* solids through the application of Boolean operations. It is one possible way to manage the problem, which we believe to be very satisfactory for many engineering purposes (Arruda, 2005). Gardan & Perrin (1995) proposed an interesting method for reducing 3D Boolean operations to a 2D problem in *manifold* environments.

Common algorithms for Boolean operations in 3D would only consider volumes as possible input and output data. Our algorithm works normally with entities of different dimensions, since it treats separately each kind of topological entity. Cases where this kind of situation occur may be of interest when one wants to deal with parts of a model where one of the dimensions can be considered irrelevant in relation to the rest of the model (e.g. flat objects in which the thickness is too small compared to their length or width). They may also be of interest when we do not want to discard internal parts of solids, dangling or loose. These parts might represent restrictions to the domain (e.g. when a finite element mesh will be generated for the solid) or layers of a different kind of material, with physical or chemical properties different from the rest of the solid. Another example in the geological field is a set of faults that cross one or more horizons of soil layers. These faults are usually represented by planes, which penetrate the solids representing the soil layers. Some faults might not cross an entire horizon, becoming a dangling face in the interior of the solid representing the solid layer.

During the implementation of this algorithm in an existing geometric modeler, MG, we tried to keep in mind the three main problem sources described above, which can easily appear while performing any task that involves dealing with arbitrary geometry of solids and topological information concerning the boundaries of these solids.

Although the algorithm itself does not contain any specific procedure to treat inaccuracy problems, the use of appropriate model-based tolerances was crucial to make sure that in every stage of the algorithm there was consistency between the geometrical and the topological data.

Robustness was perhaps the most pursued objective. Since we wanted the algorithm to be as general as possible, without imposing restrictions on input or output data, we had to consider every particular case that could happen due to unusual or unlikely data (e.g. solids touching in a single vertex, faces belonging to the boundaries of three or more solids, face overlapping, internal structures such as dangling or loose faces and multiple internal shells).

Efficiency was also a main concern during this implementation. We wondered how it would be possible to create and implement such a general algorithm without the need for routines that could increase the total amount of time spent to solve the problem. As it will be possible to verify when the algorithm is presented, the main routine is extremely simple, consisting of three main sub-routines that divide the resolution of the problem into basic steps. However, the algorithm can only be applied after some initial conditions are satisfied.

We emphasize that these conditions are not restrictions to the input data. They are only meant to make sure that the topological information that will be passed to the algorithm is consistent with the kind of treatment that will be given to it. Unfortunately, in some cases, the application of these initial conditions can be the most time-demanding task of the entire procedure. That is because these initial conditions include the intersections between all surface patches present in the model (which can have a very complex geometric description) and multiple-region detection.

In Section 2, we describe the main concepts used in this work and in previous work related to geometric modeling in *non-manifold* environments, and the application of Boolean set operations on B-Rep solids. In Section 3, we present the algorithm for applying Boolean set operations on B-Rep *non-manifold* solids, describing the initial conditions of applicability of the algorithm and the basic procedures to be done according to the desired operation. In Section 4, we show some examples of the application of the algorithm over objects modeled in MG and we discuss some limitations of the modeler. Finally, in Section 5 we present the conclusions with a summary of the main difficulties found during the implementation of the algorithm.

2. MAIN CONCEPTS AND PREVIOUS WORK

Geometric modeling, as described by Weiler (1986), involves the creation, manipulation, maintenance and analysis of the representations of geometric shapes of two- and threedimensional objects. It has application in many different areas, such as film production, design of mechanical parts for industry, scientific visualization, etc. The present work focuses on the reproduction of objects for engineering purposes.

Although during the evolution of modeling systems geometric modeling was the most recent kind of approach used as a base for their construction, there are today several works related to this approach, such as those by Weiler (1988), Rossignac & O'Connor (1990), Rossignac & Requicha (1991), Cavalcanti, Carvalho & Martha (1997), and Lira (2002).

B-Rep is a kind of representation totally based on the concept of topology. It focuses on the existence and inter-relationship among topological elements, such as vertices, edges, faces, loops, shells and solids. Detailed description of B-Rep-based systems can be found in Mäntylä (1988).

Non-manifold is a term used in geometric modeling referring to situations that are not 2manifold (Weiler, 1986). In a non-manifold environment, a cone touching a surface in only one point, more than two faces meeting at a common edge and a wire emanating from a single point of a surface are possible topological situations. Cavalcanti (1992) defines a nonmanifold solid as the immersion of several manifold solids in \Re^3 , allowing them to intersect one another but restricting the intersections to the dimensions 0 or 1. The objects are topological varieties, but their immersion in \Re^3 allows the geometric coincidence of distinct topological structures.

Systems that provide *non-manifold* environments are difficult and complex to implement, and there is not a large number of them available. The potential of *non-manifold* modeling has not yet been totally explored, and the amount of systems being developed based on this kind of representation has been increasing.

Boolean set operations are common set operations used to combine solids in order to create more complex objects. The main Boolean operations and their common definitions are:

- Union: the resulting solid occupies the space previously occupied by all the original solids.
- Intersection: the resulting solid occupies the space previously occupied simultaneously by all the original solids.

• Difference: the resulting solid occupies the space previously occupied by one of the original solids that the other solids did not occupy.

Examples of these operations applied to two solids can be seen in Fig. 1.



Figure 1- Examples of Boolean operations applied to a box and a cylinder: a) original solids; b) union; c) intersection; d) difference between box and cylinder; e) difference between cylinder and box.

Obviously, these definitions must be extended in order to provide the necessary generality we desire to our algorithm. The extended definitions will be presented in a later section.

There are many works about Boolean set operations, in CSG-based systems as well as in B-Rep-based systems. Interesting reading concerning this matter can be found in works by Gandan & Perrin (1995), Sun & Hu (2001) and Requicha & Voelcker (1984). Keiser *et al* (2004) present the combination of curved primitives through the application of Boolean operations using the concept of boundary evaluation and exact arithmetic, and show some very expressive results.

3. THE ALGORITHM FOR BOOLEAN OPERATIONS

This section presents the proposed algorithm for Boolean operations on *non-manifold* and B-Rep solids (in fact, the algorithm works for primitives of any dimension, not only for solids). As previously commented, this algorithm consists in three main stages. The first stage is the classification of the topological entities; the second stage is the application of the desired Boolean operation over these entities, based on the classification previously made for each entity; and the third stage consists in regularizing the result, if requested by the user.

The advantage of using a *non-manifold* domain to apply the Boolean operations lies in turning them into a closed set of operations. This means that the application of these operations over entities representable in a pre-established domain generates results also representable in that same domain. Therefore it is not necessary to impose restrictions or to use approximations to guarantee the validity of the result.

Even the regularization process, which consists in eliminating from the result of a Boolean operation all the entities of lower dimensions that remained loose or dangling, may not be sufficient to assure the representation of the result in a *manifold* modeling environment.

There are two important issues to be considered before the presentation of the algorithm itself. The first one is regarding the concept of boundary that will be used in this work. We consider the *boundary* of a topological entity all topological entities hierarchically inferior to it (entities of lower dimensions) that define its contour and that can be obtained by adjacency relationships through its data structure. The boundary of an entity can contain several disconnect portions, but each portion must form a connected and closed set of entities of lower dimensions. For example, the edges and vertices of the external loop of a face and those of an internal loop of the same face constituted of a closed and alternated cycle of edges and vertices are part of the face boundary. However, a wire edge loose in the interior of a face constituting an internal loop formed only by this one edge is not considered as part of the face boundary. The same reasoning can be used for the faces belonging to an internal shell of a solid. If they form a closed set of faces, constituting a smaller region inside the main region, they are considered as part of its boundary. But an internal shell constituted by one or more faces without forming a closed region is not considered as part of the solid's boundary. This particular definition of the boundary concept does not imply that these internal loose or dangling structures cannot be used as input data for the algorithm of Boolean operations. It only means that when we refer to the boundaries of entities in some parts of the algorithm, we are not including these structures.

The second issue to be considered is the definition of the concept of *group* of entities. In our context, a *group* is defined as a set of topological entities forming a sub-set of the studied model. Groups may contain vertices, edges, surface patches (faces) and regions. A group can contain several entities with different dimensions, including dangling entities or entities completely loose in space.

For each topological entity, there must be a way of accessing its adjacent entities with levels of topological hierarchy immediately above and below its own. For instance, it must be possible to access the edges and regions adjacent to a given face. Obviously, this constitutes a sufficient set of adjacency relationships, since any other desired adjacency relationship can be obtained from them (Weiler, 1986).

The Boolean operations will always be applied to two groups of entities at a time. The two groups must be under the same topological framework. Each group can possess any number of topological entities. These entities must be pre-processed to meet the conditions of applicability of the algorithm, which will be presented in the next subsection.

3.1 Conditions of applicability of the algorithm

The conditions of applicability of the algorithm are established to guarantee an essentially topological treatment of the problem. Generally speaking, the pre-processing of the input data is made to assure that vertices, edges and faces do not intersect but only touch each other, yet not necessarily along the boundaries of these elements (e.g. the end of an edge can touch the interior of a face). This means that any kinds of intersections between points, curves and surface patches must be calculated before applying the algorithm, by dividing the respective entities into two or more entities of the same type and creating the topological elements (vertices, edges and faces) that will satisfy the *non-intersection* properties presented below:

• Regions belonging to the same group cannot intersect each other, unless along their boundaries. However, regions belonging to different groups may have common volumes.

- A face can only intersect another face along the boundary of at least one of the two faces.
- Edges cannot intersect each other unless at their end points.
- Edges cannot intersect faces unless at the edges' end points.
- Vertices must be spatially distinct.

Some pertinent observations must be made concerning the non-intersection properties. It is possible to notice that spatially coincident entities, that is, entities with exactly same geometric description and boundary, are not allowed. If two or more entities like edges and faces coincide, they must be reduced to only one entity (we call that *fixing* the model). Moreover, in order to consider face overlapping without disobeying the properties described above, the following procedure has to be adopted: first the intersection between the faces is calculated. The intersection curves are either the existing edges of the faces' boundaries or segments originated from these edges. Then, these faces are destroyed and new faces are created in such a way that these new faces are simply adjacent to each other. This is possible because the original edges are divided into new edges (new vertices are also created wherever it is necessary).

Region detection is also necessary for closed volumes to be treated as regions. This is necessary because the user may be interested in treating a closed and connected set of faces just as a hollow shell, instead of as a region.

There are no restrictions concerning the geometry of the analyzed model. If all the conditions commented in this subsection are satisfied, the algorithm will work using exclusively the topological information available.

3.2 Description of the algorithm

The algorithm that will be presented is based on the one proposed by Mäntylä (1988) for Boolean operations in B-Rep representations. However, just the central idea of Mäntylä's algorithm was used, since it is only applicable to polyhedral *manifold* solids. Besides, Mäntylä's algorithm works only with two solids at a time, while our algorithm works with the concept of *group* defined above.

The proposed algorithm for Boolean operations in non-manifold B-Rep representations is summarized in Fig. 2, for two target groups named A and B.



Figure 2– Algorithm for Boolean operations in B-Rep representations.

The first step corresponds to the pre-processing stage. All intersections between surfaces and curves must be computed and all regions must be detected. The second step consists in storing in some appropriate data structures (such as lists or vectors) the topological entities belonging to each group, along with all the topologically inferior entities that form their boundaries or that remain dangling or loose in their interiors (these cases apply to faces or regions). Every topological entity must be stored only once. There must be separate lists to store entities in different groups and with different dimensions (e.g. a list containing the vertices of group A and another list containing the vertices of group B).

The third step consists in determining the common entities present in both groups. These entities must be stored in another list (which will be referred to as *common_ent*).

The fourth step consists in classifying the entities stored. The entities from one group are classified before the entities from the other group. We will use the notation *AinB*, *AoutB* and *INTERS* to indicate, respectively, an entity from group A that is located inside any region of group B, an entity from group A that is located outside all regions of group B, and an entity belonging to the *common_ent* list.

It is indispensable that methods for point location (*point in region* and *point on face*) are available, or else the classification stage will be impracticable. The descriptions of some

algorithms for point location can be found in works by De Berg *et al* (1997) and Preparata & Shamos (1990).

Three new structures can be created. Each structure should store a reference to a topological entity with one specific dimension (0, 1 or 2) and an enumerable type, for example, which will store the entity's classification. Then, three new lists can be created, one for each type of structure created previously. These lists will store, respectively, all the vertices from both groups with their respective classification, all the edges from both groups with their respective classification.

Classifying a vertex means testing it against all the regions from the other group (the group that does not contain this vertex). If the vertex belongs to the common_ent list, it is not necessary to test it; its classification will be *INTERS*.

Classifying an edge means checking the classification already given to its vertices. If the edge belongs to the *common_ent* list, it is not necessary to test it; its classification will be *INTERS*. If at least one of the vertices is inside any region of the other group or outside all regions of the other group, the edge will be classified in the same way as the vertex. If both vertices are classified as *INTERS*, then any point belonging to the edge, except for its ends, must be tested against the regions of the other group. The classification of the edge will be the same given to the point.

Classifying a face (surface patch) means checking the classification already given to the vertices belonging to its *outer* loop. If the face belongs to the *common_ent* list, it is not necessary to test it; its classification will be *INTERS*. If at least one of the vertices is inside any region of the other group or outside all the regions of the other group, the face will be classified in the same way as the vertex. If all the vertices are classified as *INTERS*, then any point belonging to the interior of the face must be tested against the regions of the other group. The classification of the face will be the same given to the point.

It is important to emphasize that the *point in region* and *point on surface* geometric algorithms used in the classification stage must be called only if necessary. Tests with the bounding boxes surrounding the point and the region/surface should be done first, for efficiency (this implicates the use of model-based tolerances).

In the fifth step of the algorithm, the classified topological entities are stored in separate lists.

In essence, applying Boolean operations over the groups' entities means removing vertices, edges and faces that are not part of the result, according to their classification. Since we are working in a *non-manifold* environment, every single entity has to be considered as being a part of the result or not, because we are not only interested in results that have the same dimension of the input entities. Therefore, the sixth step of the algorithm consists in removing undesirable faces, edges, and vertices. The order in which this occurs is important: first the undesirable faces are removed, then undesirable edges, and lastly, undesirable vertices.

Table 1 summarizes the sixth step of the algorithm for Boolean operations, which may be considered the main contribution of this work (Arruda, 2005). Entities with different dimensions are separated, and the table shows which of these entities should be removed, according to the Boolean operation that is being applied. Some of the terms used need more detailed explanation:

- *boundA* means the number of regions of group A bounded by the face.
- *theta* is the angle (in degrees) between the normal vectors of one face in relation to the two solids that the face bounds.
- *DangOrLoose* means an edge that is located in the interior of a face, dangling or loose.

- *Loose* means a vertex that is loose in the interior of a face.
- *facebound* means the number of faces bounded by the edge.
- *edgebound* means the number of edges incident to the vertex.
- *insideA* means that the face is inside any region of group A.
- *outsideA* means that the face is outside all regions of group A.
- *resbound* means the number of regions formed after the conclusion of the Boolean operation, which are bounded by the face.

Table 1. Boolean operations algorithm: entities that should be removed.

	Classification of the entity							
	AinB	BinA	AoutB	BoutA	INTERS			
	UNION							
FACES	boundA > 0	boundA > 1	None	None	boundA > 1			
					boundB > 1			
					boundA = 1			
					boundB = 1			
					theta $= 180$			
EDGES	boundA > 0	boundB > 0	None	None				
	DangOrLoose	DangOrLoose			Nona			
	belonging to	belonging to			None			
	removed face	removed face						
VERTICES	boundA > 0	boundB > 0	None	None	None			
	Loose belonging	Loose belonging						
	to removed face	to removed face						

	INTERSECTION						
FACES	boundA > 1	boundB > 1	All	All	boundA > 1 boundB > 1		
EDGES	None	None	All	All	None		
VERTICES	None	None	All	All	None		

	DIFFERENCE (A - B)					
FACES	All	boundB = 0	None	All	boundB > 1 boundA = 1 boundB = 0 insideB	
		boundB > 1			boundB = 1 boundA = 0 outsideA	
		resbound = 0			boundA = 0 boundB = 0 boundA = 1	
					boundB = 1 theta = 0	
EDGES	All	facebound = 0 facebound > 0 and all adjacent faces have been	None	All	facebound = 0 facebound > 0 and all adjacent faces have been	
VERTICES	All	edgebound = 0 edgebound > 0 and all incident edges have been removed	None	All	edgebound = 0 edgebound > 0 and all incident edges have been removed	

Regularization of the result. Regularizing the result of a Boolean operation means to eliminate from the result all the entities of lower dimensions, in such a way that only filling volumes are allowed. This means that the only entities that should remain after the process of

regularization are the faces belonging to the boundary of a given region, the edges of the boundaries of these faces, and the vertices at the ends of these edges. Therefore, loose or dangling edges and faces and loose vertices that might eventually appear on the result of a Boolean operation must be removed if the user requires the regularization process.

Regularized Boolean operations are important in applications that intend to avoid a subset of the *non-manifold* results that can be obtained by applying Boolean operations on *manifold* objects. They are also important in processes where only filling volumes are relevant or meaningful, as in industrial processes for constructing mechanical parts. In these processes, entities of lower dimensions, such as faces or edges, are not representable because they have null width.

The regularization procedure can be described as follows:

- Using the resulting entities of the application of a Boolean operation, the region detection algorithm is called. This means having access to lists of faces belonging to the boundary of each formed region.
- The faces that are not part of any of those lists must be removed, because they are loose or dangling faces.
- The edges that do not belong to the boundary of any remaining face must be removed.
- The vertices that do not belong to the boundary of any remaining face must be removed.

4. **RESULTS**

In this section we present some results we have obtained by implementing the algorithm described in the previous section in the MG modeler (Coelho, 2000 and Lira *et al*, 2002). It is important to emphasize that tools for surface intersection and for point location (*point in region* and *point on surface*) must be available in the modeler where the algorithm is being implemented. First, some *manifold* outputs are shown to demonstrate the validity of the algorithm in a *manifold* environment. Then, we present some *non-manifold* results generated by applying Boolean operations over *manifold* solids and over topological entities with different dimensions (e.g. faces and solids).

4.1 *Manifold* outputs

Applying Boolean operations over simple primitives like cylinders, boxes and spheres may lead us to more complex solids of innumerous shapes. It is possible to appreciate some of these solids in Fig. 3.



Figure 3- Examples of *manifold* solids obtained by applying Boolean operations over simpler primitives.

Figure 4 shows all the stages of the creation of a pile and the generation of a mesh of tetrahedrons for the solid that represents it. First, a union operation between two boxes and a cylinder is performed. Then, a new cylinder is inserted in the model and subtracted from the solid generated by the union operation. The last step is the generation of a finite element mesh of tetrahedrons for the solid generated after the difference operation. We must emphasize that each solid that will serve as input data for a Boolean operation must be recognized as a closed region and that all the intersections between surface patches must be calculated before the application of any Boolean operation.



Figure 4- Creation of a pile: a) primitive solids; b) primitive solids after intersection between surface patches; c) union operation between boxes and cylinder; d) insertion of a new cylinder; e) Difference between the solid generated in c) and the new cylinder; f) Generation of the mesh of tetrahedrons for the resulting solid (pile).

Figure 5 shows an example of Boolean operations applied to two solids bounded by generic NURBS surfaces. Figure 5a shows the original solids (a "twisted cube" and a "bottle") with all the intersections between surfaces already computed. Figure 5b shows the intersection between these two solids. Figure 5c shows the difference operation between the "twisted cube" and the "bottle".



Figure 5- Intersection and difference between two solids bounded by NURBS surfaces.

4.2 Non-manifold outputs

Sometimes, even when the input data are representable in a *manifold* environment, the output data may not be so. That is the case of the results shown in Fig. 6. In the first example, some dangling faces are part of the result (faces that are not part of the boundary of any solid). In the second example, there are two solids touching each other through a common edge. This edge is adjacent to four different faces.



Figure 6- Non-manifold results obtained by applying Boolean operations over manifold solids.

Figure 7 shows an example of the application of Boolean operations over entities of different dimensions. This means we have selected two groups of entities, one containing only faces and the other containing a solid.



Figure 7- Boolean operations applied over entities of different dimensions.

4.3 The modeler's limitations

Although our algorithm intends to cover all possible cases, including particular or pathological ones, such as face overlapping or faces and solids touching each other at a single point, it is not fair to say that a correct implementation of the algorithm in a modeler that provides a *non-manifold* environment is enough to guarantee that every single case will be able to be represented. Since the algorithm depends on the existence and correct functioning of two specific tools, intersection of curved primitives and region detection, it will not generate the expected results if the pre-processing stage experiences some kind of problem. These two tools, especially the intersection of curved primitives, are very difficult to implement or to be found already implemented in a robust way.

Besides these two tools, the algorithm also depends on the existence and correct functioning of two geometric algorithms, *point in region* and *point on face*. Even though these geometric algorithms are not very difficult to implement or even to be found in some geometrical libraries available, they might not be efficient enough to guarantee the generation of the result in a reasonable amount of time.

Our experience shows that there are always some cases where the intersection algorithm fails due to some particularity not considered before. Since it is based on the convergence of the Newton-Raphson method, there are always cases where the algorithm does not converge, or where it takes too long to converge (exceeding the maximum allowable number of iterations), or where the tolerance used is not appropriate to the specific case being processed. Therefore, some particular cases such as a face touching the interior of another face along an edge or face overlapping may cause the algorithm to fail.

Figure 8 shows an example where the intersection algorithm has failed. The model consists of two cylinders with the same base radius and the same height, arranged in space in such a way that they interpenetrate. As it can be seen, this interpenetration is made in a particular way, so that several faces only touch each other at single points.



Figure 8- Particular case that causes the intersection algorithm to fail.

5.4 Efficiency and robustness

It is difficult to talk about the efficiency and the robustness of the algorithm without considering the efficiency and robustness of the intersection and region detection algorithms. We also have to consider the geometric algorithms used for point location, which also might fail or be more time-demanding than expected.

We believe the algorithm itself is very robust, since it is able to deal with any possible situation, including peculiar ones, in which the topological entities have unusual adjacency

relationships, at least considering real-world models. The algorithm also proved to be quite efficient for most analyzed models. Even for big and complex models, depending on the number of times the algorithms for point location had to be called, the results were generated in a very satisfactory amount of time.

Nevertheless, considering all the necessary steps for the complete functioning of the algorithm, from the pre-processing of input data to the generation of results, there are some determining factors that might deteriorate the efficiency and robustness of the algorithm:

- Geometric complexity of the primitives present in the model
- Number of intersections between surfaces
- Number of closed regions present in the model
- Number of times the point location (particularly the point in region) algorithms have to be called

Thus, it is not simple to make absolute statements about these matters, since they depend on the auxiliary algorithms that have to be linked to the main algorithm when implementing it on a pre-existing geometric modeler.

6. CONCLUSIONS

We have presented an algorithm for Boolean operations in *non-manifold* environments and for curved primitives with any level of complexity and any dimension. The algorithm was implemented in a pre-existing geometric modeler for finite element solid mesh generation, named MG, which allows the creation of non-linear curves and surfaces using the NURBS representation (Piegl, 1991) and (Piegl & Tiller, 1999). MG works with a hybrid data representation in which every model, besides being represented by the modeler's internal data structure, can have another representation given by the CGC library (Cavalcanti, 1997), whenever the model needs to be topologically tied up.

The main idea behind the development of this algorithm was making it as general as possible in order to deal with any situation, even when the input data adjacency relationships were very peculiar, such as two surfaces touching each other at one single point, two solids touching each other along a common edge, or surface overlapping. The *group* concept was presented, meaning a set of topological entities of any dimension gathered together to be combined with entities belonging to another group through a Boolean operation.

To achieve the level of generality desired, some initial conditions had to be imposed in order to guarantee an essentially topological treatment of the problem. These initial conditions implicate pre-processing the input data to assure that all topological entities just touch each other (even if not along their boundaries) and that all regions have been detected before calling the algorithm.

The algorithm is very simple and intuitive, and its implementation in any B-Rep-based system should not require much time or a very large number of code lines. However, the system must incorporate non-linear primitive intersection, region detection and point location algorithms in order to assure the correct functioning of the Boolean operations algorithm.

The results obtained with the MG modeler were very satisfactory. Several tests were performed, using *manifold* and *non-manifold* input data and leading to *manifold* and *non-manifold* output data, yet not respectively. The algorithm proved to be quite efficient when the complexity of the model was not too high. By high model complexity we mean: presence of a significant number of high-degree curved primitives; a considerable number of surface intersections; a considerable number of closed regions; too many topological vertices of one group located inside the regions of the other group; and presence of surface overlapping. However, when these situations occurred, pre-processing always seemed to be the slowest step of the whole procedure. This means that if the Boolean operations algorithm proposed in

this work were implemented on another system with more efficient algorithms for surface intersection and region detection, the same model would be processed much faster.

Some work can be done to improve the efficiency of the point location and the surface intersection algorithms, in order to make our algorithm for Boolean operations faster. The elimination of numerical errors is also a line of research that must be carefully studied to avoid inconsistencies between geometry and topology. The algorithm can also be implemented as a class library, to be applied in different geometric modelers.

Acknowledgements

We would like to thank the Computer Graphics Technology Group (Tecgraf) and Pontifical Catholic University of Rio de Janeiro (PUC-Rio) for the financial support and for providing the necessary space and resources used during the development of this work. We are also grateful to Carolina Alfaro for the manuscript's proofreading.

REFERENCES

- Arruda, M. C., 2005. *Operações Booleanas com sólidos compostos representados por fronteira*. Msc dissertation, PUC-Rio, Departamento de Engenharia Civil.
- Cavalcanti, P. R., 1992. *Criação e Manutenção de Subdivisões do Espaço*. PhD thesis, Pontificia Universidade Católica do Rio de Janeiro, Departamento de Informática.
- Cavalcanti, P.R., Carvalho, P.C.P., & Martha, L.F., 1997. Non-manifold Modeling: An Approach Based on Spatial Subdivision. *Computer-Aided Design*, vol. 29, n. 3, pp. 209-220.
- Coelho, L.C.G., Gattass, M., & Figueiredo, L.H., 2000. Intersecting and Trimming Parametric Meshes on Finite-Element Shells. *International Journal for Numerical Methods in Engineering*, vol. 47, n. 4, pp. 777-800.
- De Berg, M., Van Kreveld, M., Overmars, M., & Schwarzkopf, O., 1997. Computational Geometry: Algorithms and Applications. Springer-Verlag, Berlin.
- Gardan, Y. & Perrin, E., 1995. An algorithm for reducing 3D Boolean operations to a 2D problem: concepts and results. *Computer-Aided Design*, vol. 28, n. 4, pp. 277-287.
- George, P.L., 1991. Automatic mesh generation. John Wiley & Sons Ltd.
- Hoffmann, C.M., 1989. *Geometric & Solid Modeling: An Introduction*. Morgan Kaufmann publishers, Inc.
- Keyser, JC., 2000. *Exact Boundary Evaluation for Curved Solids*. PhD thesis, Department of Computer Science, University of North Carolina, Chapel Hill.
- Keyser, JC., Culver, T., Foskey, M., Krishnan, S., & Manocha, D., 2004. ESOLID a system for exact boundary evaluation. *Computer-Aided Design*, vol. 36, pp. 175-193.
- Lira, W.W.M., Coelho, L.C.G., Cavalcanti, P.R., & Martha, L.F., 2002. A Modeling Methodology for Finite Element Mesh Generation of Multi-Region Models with

Parametric Surfaces, *Computer & Graphics*, ISSN: 0097-8493, vol. 26, no. 6, pp. 907-918.

- Mäntylä, M., 1988. An Introduction to Solid Modeling Computer. Science Press, Rockville, Maryland.
- Piegl, L., 1991. On NURBS: A Survey. *IEEE Comput. Graph. and Appl.*, vol. 10, n. 1, pp. 55-71.
- Piegl, L. & Tiller, W., 1999. The Nurbs Book. 2nd edition. Springer-Verlag.
- Preparata, F.P. & Shamos, M.I., 1990. Computational Geometry: An Introduction. Springer Verlag, New York.
- Requicha, A.G. & Voelcker, H.B., 1984. *Boolean operations in solid modeling: boundary evaluation and merging algorithms*. Technical Memorandum, n. 26, Production Automation Project, University of Rochester, Rochester, New York.
- Rossignac, J.R. & O'Connor, M.A., 1990. A Dimensional-independent Model for Pointsets with Internal Structures and Incomplete Boundaries. *Geometric Modeling for Product Engineering*, pp. 145-180, North Holland.
- Rossignac, J.R. & Requicha, A.G., 1991. Constructive Non-regularized Geometry. *Computer Aided Design*, vol. 23, n. 1, pp. 21-32.
- Shewchuk, J.R., 1997. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry*, vol. 18, pp. 305-363.
- Sun, W., & Hu, X., 2002. Reasoning Boolean operation based modeling for heterogeneous objects. *Computer-Aided Design*, vol 34, pp. 481-488.
- Weiler, K., 1986. *Topological Structures for Geometric Modeling*. Ph.D. thesis, Rensselear Polytechnic Institute, Troy, N.Y.
- Weiler, K., 1988. The Radial-Edge Structure: A Topological Representation for Non-Manifold Geometric Boundary Representation. *Geometric Modeling for CAD Applications*, North Holland, pp. 3-36.