Unified Geometric Modeling by Non-Manifold Shell Operation

Masatake Higashi*, Hideki Yatomi**, Yoshihiro Mizutani** and Shin-ichi Murabata***

*: Toyota Technological Institute, 2-12-1, Hisakata, Tempaku-ku, Nagoya 468, Japan

**: Nihon Unisys Ltd., 1-1-1, Toyosu, Koutou-ku, Tokyo 135, Japan

***: Toyota Motor Corp., 1, Toyota-cho, Toyota 471, Japan

Abstract

We propose a method for unified geometric modeling by non-manifold shell operations. The method enables a designer to generate a solid shape freely combining shells which are composed of surface (lamina) shapes and solid shapes.

Our research emphasis is on the process for modeling a non-manifold model as well as its representation. Considering that a new shape is constructed by combining two shells so as to interchange their surface connections and separate the combined non-manifold shell, we establish a unified algorithm to process a regularized set operation of solids and construction of a solid from a surface shell and a solid or from several surface shells. A designer can thus obtain a desired solid shape easily by combining shells whose shapes are determined locally according to design requirements.

The non-manifold shell which appears as an intermediate shape in the design process or in the calculation is represented by a cycle structure and entirely manipulated by Euler operators. The cycle structure is a data structure extended from a half-edge structure to represent a non-manifold shape and cycles around a vertex and an edge as well as a face. Non-manifold Euler operators include vertex connection and two types of edge connection in addition to the manifold Euler operators. We treat a surface shape as a non-manifold shape which is degenerated from a solid shape and is an open shell with two surfaces. This makes it possible to treat outer and inner surfaces equally, while only outer surfaces which make closed shells have meaning for a solid shape.

We show the effectiveness of the method by presenting some examples produced by a prototype system which realizes the shell operation by geometrical and topological routines.

2nd ACM Solid Modeling '93-5/93/Montreal, Canada
© 1993 ACM 0-89791-584-4/93/0005/0075...$1.50

Keywords: geometric modeling, non-manifold geometry, shell operation, Euler-Poincare formula, regularized set operation, half-edge structure, cycle structure

1. Introduction

Nowadays, CAD/CAM systems enjoy wide spread use in industry [6]. The key for integrating the systems from the beginning of the design to the manufacture is a geometric model in the computer. In paticular, a solid model which represents a three-dimensional solid shape in the computer is necessary for applications involving three-dimensional topology, such as FEM analysis of solid elements and NC machining of products. But for various reasons designers may not use a solid modeling system to design a desired shape:

- The human interface to generate a solid such as Boolean set operation of primitives and local operation of solids is not matched to designers' feeling and thinking process.
- The designers are not allowed to create imperfect or ambiguous shapes which appear in the designing process and are determined part by part according to the design requirements.
- Expressing and performing engineering changes or simple corrections of design error cannot be easily done by designers.
- CAD/CAM systems are unstable and calculation stops during operations sometimes because of numerical errors.

These problems arise from insufficient flexibility of the geometric modeling framework. To enhance modelling capabilities, many approaches, such as parametric or variational design, form feature manipulation, geometric reasoning, and non-manifold geometric modeling, have been studied and implemented in practical systems. One of the basic technologies here is coexistence of different dimensional entities in the data structure, namely a vertex, an edge, a face and a region.

A non-manifold geometric model is expected to deal with this. Weiler proposed a non-manifold data structure [11], called a "radial edge structure," and its manipulation operators [12]. Gursoz et al. [1] extended this to a vertex-based representation which presents new entities such

as zones and disks around a vertex. Other non-manifold modeling technologies are also proposed [5][13][14]. Such studies focus on representation and lower level operators of the non-manifold model. On the other hand, Rossignac and O'Connor [7] proposed Selective Geometric Complexes (SGC) which are composed of finite collections of mutually disjointed cells and provide a framework for representing objects of mixed dimensionality. And Rossignac and Requicha [8] proposed Constructive Non-Regularized Geometry (CNRG) for dealing with mixed-dimensional objects as extended Constructive Solid Geometry (CSG). A geometric and topological algorithm for the intersection between objects with different dimensionality was proposed by Gursoz et al. [2].

Compared to the above approaches, we focus on generation of a solid as our research objective, seeking to enable designers to generate an imperfect shape as an intermediate one, to freely combine shells composed of surfaces and solids, and to extract a solid from the non-manifold shell without manual trimming and deleting operations. We present a geometric modeling framework to manipulate solids and surfaces in a unified way and generate a solid through a simple algorithm. The framework is constructed according to the following strategies:

(1) We treat only shells (two-non-manifold) and generate all the shapes which designers require.

(2) We distiguish a surface from a solid. A solid has only outer faces, while a surface has two faces, outer and inner, which should be treated equally and be useful in shell separation by peeling off them.

(3) We present a unified simple algorithm for the shape generation which combines shells so as to interchange their surface connections and separate the connected non-manifold shell without changing the connectivity.

Our framework for shell operation is a natural extension of the conventional two-manifold solid modeling. So the algorithm of the shell operation is common for the set operation of solids including the regularized operation and the solid generation from the non-manifold shell which is a combination of surfaces and/or solids. It uses only two-dimensional shells and represents three-dimensional regions as subdivisions of a three-dimensional Euclidian space E^3 by the boundaries that are shells. There are two parts: geometrical calculation and topological manipulation, whose roles are clarified and can be implemented independently. These make the algorithm simple. Furthermore, since the intermediate process of the operation is stored in the data structure, the algorithm becomes rigorous by restarting the process when the system stops.

To represent and manipulate the non-manifold shell, we also propose a data structure, named a cycle structure, and its Euler operators. The cycle structure is an extension of the half edge structure by Mäntylä [4] and manages cycles around a vertex and an edge as well as a surface. The Euler operators corresponding to the extended Euler-Poincare formula include edge connection and vertex connection of the shells.

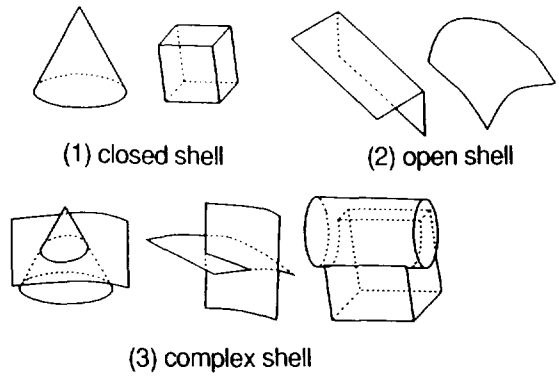


Fig. 1 Classification of shells

This paper covers shell operation in Section 2, data structure and Euler operation set in Section 3, generation of solids by set operation and closed shell extraction in Section 4, and a summary is given in Section 5.

2. Shell operation

In this section, we first define a shell and terminology used in this paper. And next we present the concept of the shell operation which generates solids. Then we describe a general algorithm of the shell operation.

2.1 Shell and shell operation

A *shell* is defined as a collection of faces which are oriented by their boundaries and connected at the edges. A shell must satisfy the following two criteria.

- The orientations of adjacent faces must be the same.
- A shell does not intersect at other than edges.

We classify the shell into three types: a *closed shell*, an *open shell* and a *complex shell*.

A *closed shell* has no boundaries and is homeomorphic to a sphere. It has only outer faces to divide E^3 into an outer region and an inner region which corresponds to a given volume. The outer region is represented by the directions of the normal vectors of the faces, and cycles going around boundaries, which are called *surface loops*, are directed for the surface normals. Typical examples of this type of shell are the primitive solids shown in Fig. 1 (1). An *open shell* has boundaries, which we call *boundary edges*, and is homeomorphic to a disc. We also call the open shell a *surface shell* or merely a *surface*. The open shell has faces, which can be termed outer faces, on both sides of the shell. Since it cannot divide E^3 into regions, there is one tree-dimensionsl open space surrounding it. So we consider that the surface shell is a non-manifold shape, whose shape is a lamina and whose two faces are degenerated [9]. An example is given in Fig. 1 (2). A *complex shell* is a combination formed by making the above types of shells intersect. This type of shell has more than two faces at the intersecting edges. Examples of this type are shown in

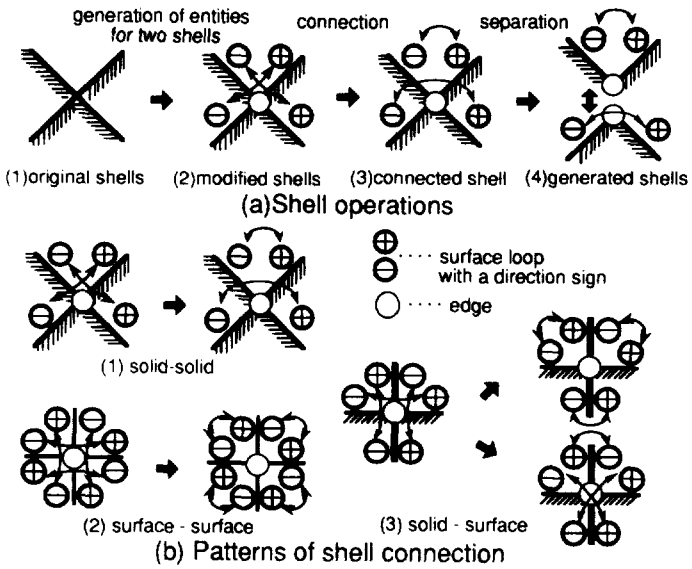


Fig. 2 Shell operation and patterns of shell connection

Fig. 1 (3).

The shell can be classified from another point of view into *manifold* and *non-manifold* types. A *manifold shell* is defined in [4]: a two-manifold shell is a logical space where every point has a neighborhood topologically equivalent to an open disc of E^2. By definition, only a closed shell is a manifold shell, because a surface shell has boundaries and a complex shell has branches at the intersection edges.

We manipulate these types of shells by shell operation so as to connect and separate shells. An intermediate shape in the calculation for generating solids is represented as a complex shell. On connecting shells, the shell operation interchanges the connecting surfaces of the shells and separates the connected shell without changing the connectivity (see Fig. 2 (a)). These shell operations generate different shells from the original ones. For example, in the set operation of solids, the *union* and *intersection* of the given solids are generated, and when a solid intersects with a surface and has a loop intersection curve, two solids and a surface with a hole are generated.

The connectivity of shells is represented by a link of faces of the shell along the common edges. Since faces are oriented by the surface loops, the directions of the two adjacent loops to be linked should be opposite. When two shells intersect with each other, the face linkage is changed to achieve this. This re-link assures compliance with the first criterion of the shell: the same orientation of the adjacent faces. Signs of two surface loops along the edge are noted "+" and "−" according to their directions. If a direction of a surface loop is the same as the geometrical direction of the edge, its sign is plus; otherwise it is minus.

We classify the shell connection at the edges, using the above notations, into three types according to the types of shells to be combined as shown in Fig. 2 (b): a solid with a solid, a surface with a surface, or a solid with a surface. In the figure, arrows indicate the connectivity before and after connction. When a solid and a solid intersect, there is one degree of freedom to interchange "+" and "−" surface loops because the number of faces for each shell at the intersecting edge is two. Interchanging the link for the loops with different signs means compliance with the criterion of the shell connectivity. But when a surface and a surface intersect, the number of faces is four for each shell. The selection of the link is determined using the direction of the surface normal there to divide the E^3 into appropriate regions. Looking at one side of surface *A*, there is one surface branch of the other surface *B*. So we can link the "+" surface loop of this side of *A* and "−" surface loop of the branch of surface *B*. Lastly, when a surface and a solid intersect, the link may be selected freely, depending on which part of the shape the designers want inside or outside of the solid. This selection should be established beforehand by the designers.

Next we separate the combined shell if possible. At the time, the combined shell is separated without changing the shell connectivity along its edge. By combining and separating the shells, we obtain shapes different from the original ones. In case of that the shells are both solids, this operation generates *union* and *intersection* shapes of the solids. And for the closed shell extraction from the non-manifold complex shell, we obtain positive or negative solids by the separation.

2.2 General algorithm of shell operation

In this section, we present a general algorithm of the shell operation and focus on the roles of geometrical and topological processing. The data structure and lower level Euler operators which execute topological shell operations in the data structure are described in the next section.

Fig. 3 shows a general flow chart of the connection of two shells *A* and *B*. The calculation is repeated between all the surfaces of shell *A* and all the surfaces of shell *B*. First, the intersection curves of two surfaces are calculated by the geometrical tracing method [3]. And the inner parts of the

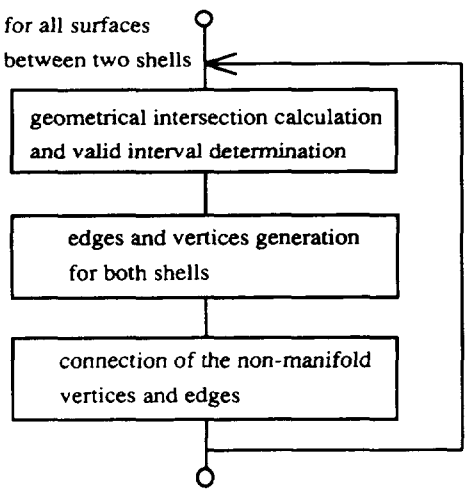


Fig. 3 General flow chart of shell connection

Table 1-1 Generation of topological entities according to end-points pattern of the intersection

Start \ End	Vertex	Edge	Face
Vertex	(a)(c) E,F (mef) (b) E (mekr)	(a) V,2E,F (semv, mef) (b) V,2E (semv, mekr)	(a) E,V (mev)
Edge	(a) V,2E,F (semv, mef) (b) V,2E (semv, mekr)	(a) 2V,3E,F (2semv, mef) (b) 2V,3E (2semv, mekr) (c) V,2E,F (semv, mef)	(a) 2V,2E (semv, mev)
Face	(a) E,V (mev)	(a) 2V,2E (semv,mev)	(a) 2V,(+2-1)E (mev,kemr,mev) (c) V,(+2-1)E,F (mev,kemr,mef)

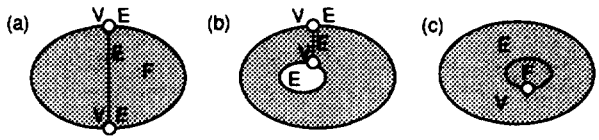
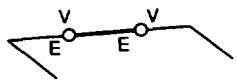


Table 1-2 Generation of topological entities (in case of edge coincidence)

Start \ End	Vertex	Edge
Vertex	No op.	E,V (semv)
Edge	E,V (semv)	2E,2V (2semv)



intersection curves for the both surface loops are determined and regarded as edges. These are performed by geometric routines. If all the intersections are detected without omission by the geometric routines, the second criterion for the shell is satisfied.

Next, the end points of the curves, which are calculated as intersection points among three surfaces and to be vertices, are classified according to a pattern of their locations in each surface. The pattern consists of a point coincident with the original vertex, a point on the edge and an independent point on the face. Also, an edge where two surfaces intersect is classified by whether it corresponds with the original edge or not. According to the geometrical pattern check of the start and end points and the edge coincidence, topological modification of the data structure is determined as shown in Table 1: it is determined whether a new vertex (V), a new edge (E) and a new face (F) should be generated or not. In Table 1-1, the pattern of the intersection curve is divided into three types according to face cycles: (a) within the same face cycle, (b) between different face cycles (e.g., between an outer boundary and an inner boundary of an island), and (c) an intersection curve forming a loop by itself. Appropriate topological modifications for each shell are performed by Euler operators shown in parentheses in the table. Table 1-2 shows the case in which the intersection curve corresponds with the existing edge.

Next, the corresponding vertices and edges are connected by the non-manifold Euler operators, to satisfy the shell connectivity according to the shell connection pattern. After all the intersection calculations and connections are normally completed, the intersection curves make loops or reach the boundary edges of the open shell. This can be checked automatically by a topological routine.

Lastly, the combined complex shell is separated by the non-manifold Euler operators which separate edges and vertices without changing the shell connectivity. Since all the topological shell operations are done using Euler operators, generated shells satisfy the shell topological criterion.

3. Data structure and Euler operation set

In this section, we present a data structure which represents the shells described in Section 2 and an Euler operation set which manipulates the topology of the data structure. Before this, we discuss the mathematical foundation of the non-manifold and introduce an extended Euler-Poincaré formula. To represent the connectivity, we apply the *circularity* and the *connectedness* of the *boundary concept* from mathematics to the data structure and the Euler operation set. *Circularity* means that one goes along a path and eventually reaches the starting point again. *Connectedness* is defined as that a certain path exists between any two arbitrary points on the object. A path which has circularity is called a *cycle*. A cycle which breaks the connectedness of an object and separates it into two objects is called a *boundary cycle* or simply a *boundary*.

We can localize the above discussion, concentrating on a certain element and its vicinity. Its *local cycles* are paths which circle around the element and are included in its arbitrary neighborhoods. Its *local boundary* is defined as a local cycle which breaks the connectedness between the element and the object supporting it and separates it from the object. A local boundary corresponds to the central element in a one-to-one relationship. We distinguish the localized concepts from the previous global ones by adopting the term *local* in this paper.

Conventional surface loops which represent orientations of faces are regarded as local cycles of the face, since they repeat an edge and a vertex alternately around a certain face, as shown by (1) in Fig. 4. The whole of loops at the face is a local boundary, since it separates the face from the model by cutting the face along the loops. Considering the duality of a vertex and a face, the same discussion applies to a vertex. We take an example of a vertex where two vertices are connected into one in non-manifold models, as shown by (2) in Fig. 4. There are local cycles which are dual in relation to face cycles and which repeat an edge and a face alternately around the vertex. The whole of the cycles is a local boundary with duality which retains the connectedness between the vertex and the model. The same applies to edges, as shown by (3) in Fig. 4. Edge cycles repeat a face

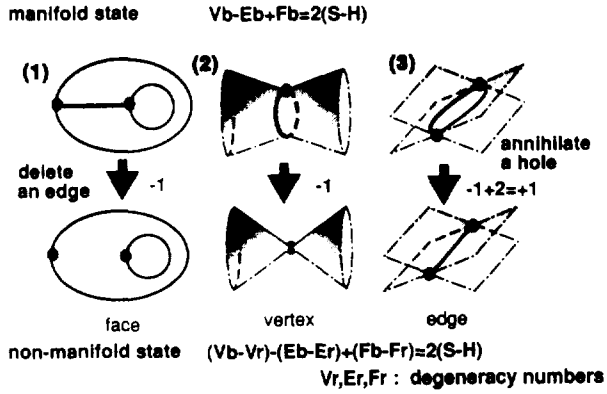


Fig. 4 Transition into non-manifold

and vertex two times around an edge, and the whole of those is a local boundary of the edge. All of these cycles must satisfy the topological shell criterion.

3.1 Extended Euler-Poincaré formula

We introduce an extended Euler-Poincaré formula to satisfy the shell topological criterion in the shell connection and separation, by treating the non-manifolds as an extrapolation of the manifold theory in mathematics. We suppose that a non-manifold state such as a vertex connection or an edge connection is an extreme state of the transition from a certain manifold state due to degeneration of some elements as shown in Fig. 4. The figure indicates a certain element and its neighbourhood in a model.

In the case of (1) a surface loop for a face, one edge disappears through the transition from the upper manifold to the lower non-manifold. We treat the lower state with surface loops as the non-manifold, although it is treated as the manifold in the conventional solid modeling by subtracting a number of loops from the original Euler-Poincaré formula in mathematics:

$$Vb - Eb + Fb - L = 2(S - H), \quad (1)$$

where Vb, Eb, Fb, L, S and H are numbers of vertices, edges, faces, loops, shells and holes.

Instead we consider that the non-manifold state should have a hypothetic edge to be a manifold state. When two vertices are connected as in case (2), an edge degenerates into a vertex, and there exists a hypothetic edge in the non-manifold state. For an edge connection as in case (3), one edge and one hole disappear. We should count a hypothetic edge and hole in the non-manifold state. We obtain the Euler formula of the non-manifold from that of the manifold adding the hypothetic edges and holes:

$$Vb - (Eb + Vr + Er + Fr) + Fb = 2(S - (H + Er)), \quad (2)$$

where Vr, Er and Fr are numbers of the hypothetic edges added to each of the dimensional entities. A transition from a manifold to a non-manifold state in Fig. 4 adds -1, -1 or $+1$ to eq. (1), respectively. Arranging terms of the equation (2) for each of the entities, we obtain

$$(Vb - Vr) - (Eb - Er) + (Fb - Fr) = 2(S - H). \quad (3)$$

where Vr, Er and Fr are numbers of degeneracies, that are hypothetic edges, for each of the entities. As a result, the hypothetic edge corresponds in a one-to-one relationship to each of the degeneracies in the non-manifold and converts the non-manifold state to the manifold state by subtracting a number of degeneracies from a number of corresponding geometric elements.

In the manifold state, H can have the obvious meaning of a number of geometric holes. On the other hand, in the non-manifold state, it is a mere number for logical adjustment and may take a minus value. We cannot determine whether the non-manifold shape (3) in Fig. 4 is made from a shell with a hole or from two shells contacting each other. There are multiple ways to make the transition. Our formula is unique to the transition, but not to the non-manifold shape. This multiplicity is controllable by considering the sequence of dynamic transitions and makes the shell operation able to change the shell interconnections.

3.2 Data structures

We propose a data structure which we call a *cycle structure*. The cycle structure is an extension of the *half edge structure* by Mäntylä [4] to represent non-manifold shells appearing in a shell operation. The half-edge structure represents solid volumes by their boundaries, i.e., shells, but it cannot allow the existence of more than one vertex and edge cycle which are made by a shell connection. So, we add vertex cycle and edge cycle entities to it in addition to face cycle entities, i.e., face loops. This makes the cycle structure symmetrical to the geometric entities and their cycles, while the half-edge structure is hierarchical for only face cycles.

The cycle structure represents global topologies and local topologies of a shell. A global topology of a shell is represented by a child ring of the shell, and a local topology is represented by linking pointers in half-edges. Traversal in the data structure using pointers in half-edges obtains ordered faces in an edge and vertex cycle which divide three-dimensional regions at the vicinity of the geometric element. As a result, the edge cycle corresponds to the radial relation of the radial edge structure [11] and the vertex cycle correponds to the zone proposed by Gursoz et al. [1]. Since the cycle structure is entirely manipulated by Euler operators which satisfy the conditions of the Euler-Poincaré formula when shells are generated and modified by shell operations, topological consistency of the shell is maintained in the structure.

All entities in topology and geometry are mapped onto the hierarchical ring structures, as shown by Fig. 5. The hierarchical ring structure becomes a topological graph structure by local linking of half-edges. Each disjointed *object Ob* has a child ring of *shells Sh*, where only one is a positive shell and the others are negative cavities for a solid shape.

A *shell* is the entity which represents global connectedness in the 2D B-reps model. Therefore, most of the information required by global operations such as the

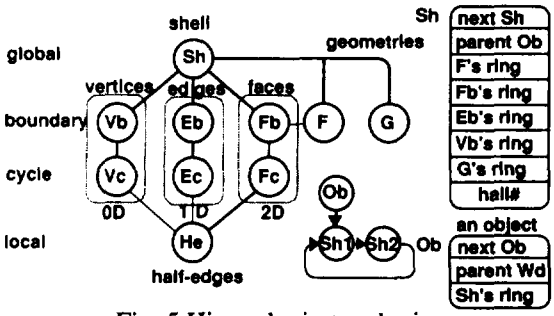


Fig. 5 Hierarchy in topologies

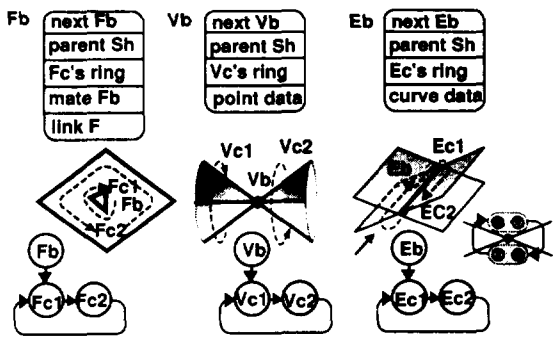


Fig. 6 Local boundaries and local cycles

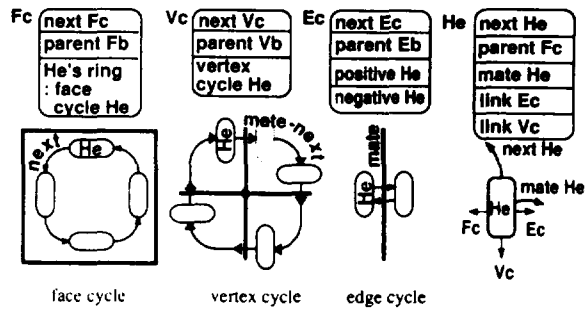


Fig. 7 Local cycles and half-edges

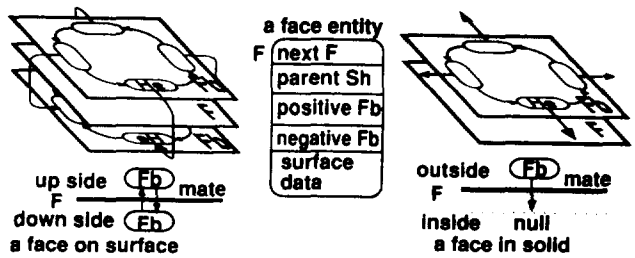


Fig. 8 Face entity

shell operation concentrates at this shell. The shell has a number of holes and five child rings: three local boundaries, a *vertex boundary Vb*, *edge boundary Eb* and *face boundary Fb*, which correspond to 0D, 1D and 2D geometric objects, and faces *F* and geometries *G*. The local boundaries have a child ring of local cycles: *vertex cycles Vc*, *edge cycles Ec* and *face cycles Fc*. Only the local face cycle has a child ring of *half-edges He*. The other local cycles for vertices and edges have no child rings.

The *face boundary* links to all the face cycles belonging to the face as a ring. The vertex boundary similarly holds all the vertex cycles concentrating at the vertex. And the edge boundary links all the edge cycles which are co-axial or radial with the curve. The edge cycles consist of a pair of half-edges and have an ordered circularlity like half-edge rings (see Fig. 6).

The *half-edge* is the entity which represents local connectedness in contrast with the global one of the shell, and indicates three local cycles in its neighborhood by three pointers: *link Ec*, *link Vc* and *parent Fc*. *Link Ec* indicates an edge cycle which involves this half-edge as one of a pair, *link Vc* indicates its starting vertex of the half-edge, and *parent Fc* indicates the face cycle which involves it. The *mate He* indicates another half-edge in the same edge cycle. The local cycles around the geometric entity are shown by Fig. 7, and traversed using the half-edges as follows:

The face cycle is a counterclockwise circular loop of half-edges chained by ***next****.*

The vertex cycle is a clockwise radial loop of half-edges chained by ***mate*** *and* ***next*** *alternately.*

The edge cycle is a loop of half-edges chained by ***mate****, i.e., a pair of half-edges.*

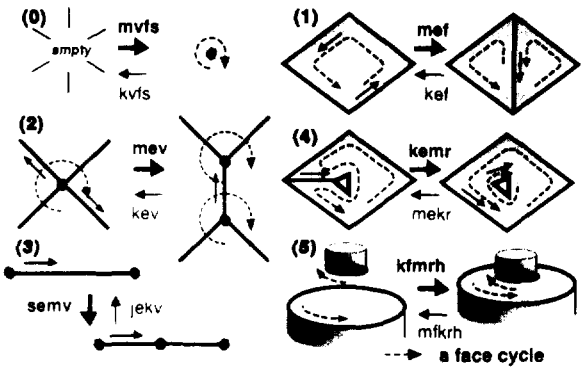
The face cycle entity has a pointer indicating a half-edge which circulates around the face cycle as in Fig. 7. The vertex cycle entity has a pointer indicating a half-edge which radiates from the vertex cycle. And the edge cycle has two pointers indicating a positive half-edge and a negative one according to the direction of the geometric curve.

The *face entity F*, shown by Fig. 8, is introduced for resolving non-uniqueness: *F* should correspond to a pair of face boundaries for the surface shape, but in fact corresponds to only a face boundary for the solid shape and indicates a geometric surface. So it has two pointers indicating a positive face boundary and a negative one according to the direction of the surface normal. The face boundary has the same property as the half edge and can be considered as a *half-face*. The face boundary has a mate pointer indicating another face boundary in the face entity. The mate pointer has a null value for a solid shape, but an active value only for a surface shape.

Geometric entities G contain geometric data such as surface equations, curve equations and point locations which are affected by the coordinate transformation. The vertex boundary and the edge boundary have pointers indicating geometric entities. On the other hand, the face boundary has a pointer indicating the face entity *F*, and refers indirectly to a geometric surface through the face entity.

3.3 Euler operation set

To perform shell operations consistently, we add four operations and their inverses ones to the conventional Euler operation set which Mäntylä [4] has proposed. They comprise one connecting operation for vertices, two connecting operations for edges and a local transformation



Notes: Terminology and notation are conventional: m is making, k is killing, s is splitting, j is joining, r is degenerating (rather than ring), h is hole. And swapping between m and k, s and j, makes inverse operator names. Vertex cycles and face cycles vary in operations, especially in non-manifold operations.

Fig. 9 Set of the conventional Euler operations by Mäntylä

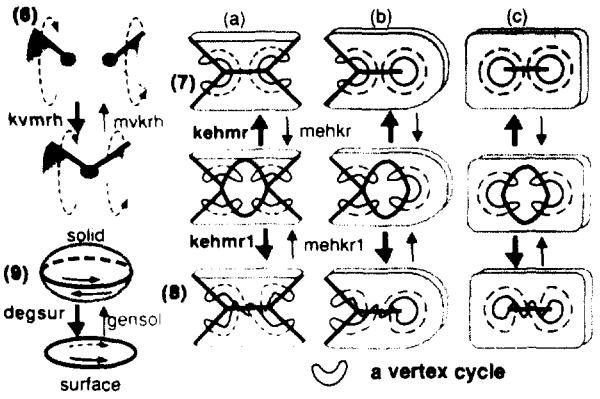


Fig. 10 Set of new Euler operations

between a solid and a surface. As a result, we obtain a whole set of operations for B-reps non-manifold models. The conventional Euler operation set is shown in Fig. 9. Our additions are given in Fig. 10. Conventional operations are applied to the manifold part maintaining correspondence in a one-to-one relationship between a boundary and a cycle in vertices and edges, respectively. And they are applied to a non-manifold surface by transforming the surface to a solid with two faces and transforming it back to a surface after the corresponding manifold operations.

New operations are introduced as follows. We examine the duality between vertices and faces in the conventional operations. The operators *mev* and *mef* obviously have a dual relation to each other, *mvfs* has this in itself and *semv* has no involvement with duality. But operators *kemr* and *kfmrh* have no dual operators in the set. We make their partners according to the duality between vertices and faces. We add *kehmr* and *kehmr1* from *kemr*, and *kvmrh* from *kfmrh*. We discuss the functions of added operators compared to those of the partners. The dual functions maintain the topological shell connectivity in the data structure.

First, we introduce *kehmr* as a partner of *kemr*. In the data structures, *kemr* and its inverse *mekr* swap *next* pointers between two half-edges in the same face cycle or in different face cycles at the same face boundary by deleting or creating an edge, join two face cycles into one, or separate a cycle into two. We can consider the dual functions at a vertex, and make an edge connection operator *kehmr* and its inverse *mehkr* that is an edge separation operator. They swap *mate* pointers between two half-edges in the same vertex cycle or different vertex cycles at the same vertex boundary by deleting or creating an edge, join two vertex cycles into one, or separate a vertex cycle into two. The edge cycles are simultaneously joined or separated, swapping their positive half-edge pointers corresponding to the swapped *mate* pointers. However, these operators cannot conclude at one vertex only, because an edge has two ending vertices and two mated half-edges. As a result, they change a number and pathes of vertex cycles at both vertices, and are divided into three cases, as shown in Fig. 10, according to the states of the vertex cycles at both ends. These operations satisfy shell connectivity at a vertex to divide E^3 into regions.

On the other hand, we introduce another edge connection operator *kehmr1* and its inverse *mehkr1*. They do not swap the *mate* pointers of two half-edges and do not change the vertex cycles there, but edge cycles are joined or separated in the same manner as *kehmr* and *mehkr*. These edge connection operators, as well as *semv*, have no duality for a vertex and a face and their ring manipulations are similar to those of *kfmrh* and *kvmrh*.

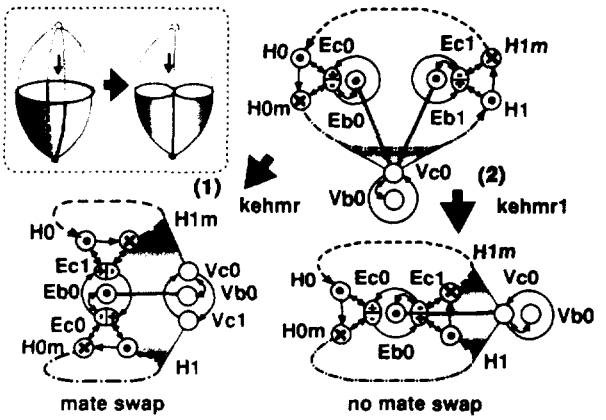


Fig. 11 Edge connections

Fig. 11 shows functions of two edge connection operations at one vertex of two. The shells connected are different in a number of vertex cycles, half-edge pointers in the edge cycles and mate pointers of the half-edges. These operators can be applied only at the place where ending vertices of the two edges are connected into common vertex boundaries. Therefore, they are always used after the vertex connection operations. For applying *kehmr* or *kehmr1* to two edges, they must be arranged in the same direction geometrically.

Next, we introduce *kvmrh* as the partner of *kfmrh*. The

Table 2 Transition vector

		Vb	Eb	Fb	Vr	Er	Fr	H	S
0)	**mvfs**/kvfs	1	0	1	0	0	0	0	1
1)	**mef**/kef	0	1	1	0	0	0	0	0
2)	**mev**/kev	1	1	0	0	0	0	0	0
3)	**semv**/jekv	1	1	0	0	0	0	0	0
4)	**kemr**/mekr	0	-1	0	0	0	1	0	0
5) -a)	**kfmrh**/mfkrh	0	0	-1	0	0	1	0	-1
-b)		0	0	-1	0	0	1	1	0
6) -a)	**kvmrh**/mvkrh	-1	0	0	1	0	0	0	-1
-b)		-1	0	0	1	0	0	1	0
7) -a)	**kehmr**/mehkr	0	-1	0	-2	1	0	-2	0
-b)		0	-1	0	0	1	0	-1	0
-c)		0	-1	0	2	1	0	0	0
8)	**kehmr1**/mehkr1	0	-1	0	0	1	0	-1	0
9)	**degsur**/gensol	0	0	0	0	0	0	0	0

operator *kfmrh* is a connection operation for faces. It deletes one face boundary and joins its face cycle with another face boundary, and degenerates the face. By merely replacing a face with a vertex, we easily obtain the dualized operation *kvmrh* which is a connection operator for vertices. It deletes one vertex boundary, and joins its vertex cycle with another vertex boundary, and degenerates the vertex. We also obtain the inverse operator *mfkrh*.

The operators *kfmrh* and *kvmrh* are divided into two types by their functions. Applying different shells, they join two shells into a single shell. Applying one shell, they make a hole. The inverse operations *mfkrh* and *mvkrh* separate a single shell into two if the target local cycle is a global boundary, or delete a hole if it is not.

The last operations, *degsur* and *gensol*, are a local transformation between a solid and a surface and the opposite. The operator *degsur* is applied between two face boundaries which have the same local structure and share all edges and all vertices; it deletes a face F of one face boundary, conjoins two face boundaries together with another face F, and degenerates a local 3D region into a thin skin. As a result, two face boundaries become the "up" and the "down" side of the surface.

Next, let us consider the transition vectors of these operators. A vector consists of eight tuples such as $(Vb, Eb, Fb, Vr, Er, Fr, H, S)$. Table 2 shows transition vectors of the conventional operations and our adding operations, which indicate only positive ones. The operators *kvmrh* and *kfmrh* are similar to each other and two vectors: one is for the same shell, the other for different shells. The operator *kehmr* has three vectors. The first one is for different cycles at both vertices. The second one is for the same cycle at one vertex and different cycles at the other vertex. The last one is for the same cycles at both vertices. On the other hand, *kehmr1* has only one vector. The operators *kehmr1* and *kehmr* reduce the hole number H. The number of holes H in *kehmr1* is reduced by 1, but in *kehmr* it is reduced by 1 only when two vertex cycles conjoin at each vertex, respectively. This seems to be somewhat complex, but makes natural transitions between non-manifolds and manifolds. Fig. 12 shows an example of the transitions with their vectors from a manifold to two manifolds

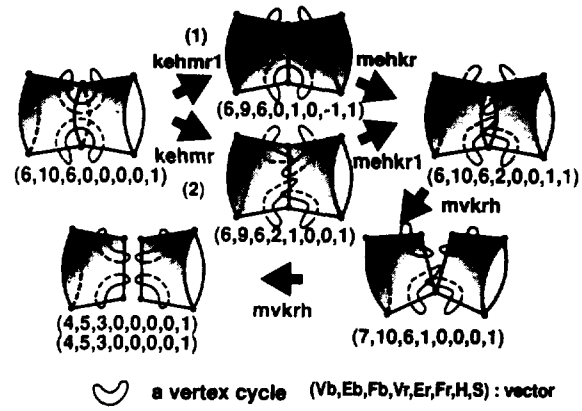


Fig. 12 Example of transitions

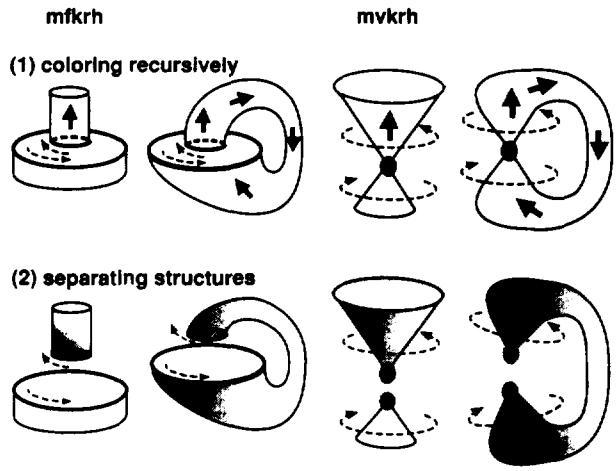


Fig. 13 Global shell separation

through non-manifold states. There, figure (1) is equal to figure (2) geometrically, but not topologically. There are two transition paths.

All operations are accessed with input parameters of a half-edge pointer or a pair of half-edge pointers and return a half-edge pointer as output, because all entities can be indicated locally by a half-edge. Selection of a half-edge in shell operation is performed according to the shell connection pattern in Fig. 2.

3.4 Check and separation of a global shell

The operators *mfkrh* and *mvkrh* are associated with the global shell structure and accessed by inputting a local cycle. It is necessary to check whether the target local cycle is a global boundary cycle or not. The operators separate a shell into two shells if the local cycle is a global boundary, but annihilate a hole without separating, if not a global boundary. These operations consist of two stages, coloring and separating, and can execute this global check automatically as shown in Fig. 13. The coloring indicates a connected shell to the local cycle, setting a global flag at each node. It is recursively performed as follow:

(1) The coloring routine sets the flags of all half-edges

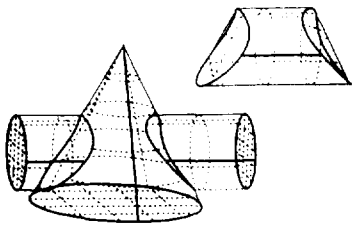


Fig. 14 Union and Intersection of solids

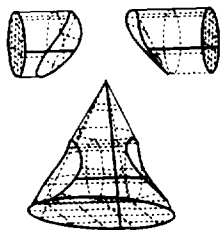


Fig. 15 Exclusive-or of solids

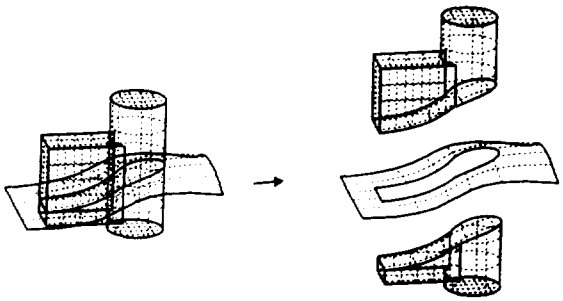


Fig. 17 Separation of a solid by a free-form surface

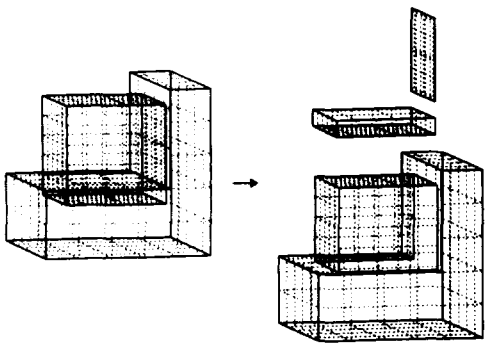


Fig. 16 Regularized set operation

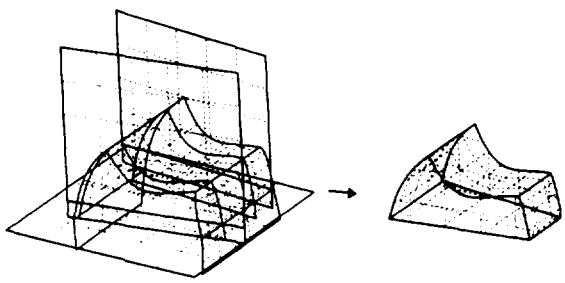


Fig. 18 Generation of a solid by combining surfaces

belonging to the initial local cycle as a barrier against which the routine does not move to any nodes, and applies the color set to the mated half-edges of all the half-edges in the initial cycle,

(2) The coloring routine traces recursively in the hierarchy of topologies from the bottom half-edge nodes to the upper boundary nodes through cycle nodes by utilizing the local links of half-edges, setting the color flag at each node, and terminates if it can go nowhere.

As a result, if the local boundary involving the initial local cycle has been colored, the initial one is a global cycle but not a global boundary. And if not so, it is a global boundary, because the initial cycle is barricaded with the flags at the first step. Next, when the local cycle is a global boundary, the shell is separated into two. The separating routine recursively separates the hierarchy of the structure under the shell according to the resulting color. For the convenience of this process, a shell has many entries to access child rings, such as geometry data and so on.

4. Generation of solids

We present two applications of the shell operation. One is a Boolean set operation and the other a solid extraction from the non-manifold shell which is generated by the shell connection among surfaces and solids.

4.1 Set operation

In the conventional algorithm for the set operation proposed by Mäntylä [4], shells of the given two polyhedron solids *A* and *B* are classified into *AinB*, *AoutB*, *BinA and AoutB* and reconnected using new vertices, null edges and null faces along the intersection curves. Here the algorithm owes to *vertex neigborhood classification* which decides the side of the edges of the vertex cycle for the intersecting plane, and inserts *null edges* and *null faces* for the appropriate edges. Furthermore, the algorithm uses three tables for vertices, edges and faces, and needs detailed classification when planes or an edge and a plane of the two solids coincide.

Instead in our shell operation, we directly connect two shells by non-manifold Euler operators. We do not need temporary tables or entities in the data structure. The vertex neighborhood classification is replaced with a simple algorithm of geometric and topological routines. In the geometric routines we calculate intersections including coincidence among vertices, edges and faces as described in Section 2. In the topological routines, we first generate new vertices, edges and faces on each shell according to the classification in Table 1 in Section 2. Next, we connect the corresponding intersecting edges by the extended Euler operators *kvmrh* and *kehmr*. In the connection, there is no freedom because of the constraint for the surface loops. We need not give any special treatment to *dangling faces* which appear in the non-regularized set operation. The dangling faces are connected and separated automatically by the shell operations. The shell operation separates the combined shell at the edges where there are more than one edge cycle.

We introduce here some examples of set operation of the solids. Fig. 14 shows the results which are *union* and *intersection* of the solids. Fig. 15 shows the results which are *exclusive-or* of the solids and generated after the reverse operation for the subtracting solid. Fig. 16 shows an

example of the regularized set operation which separates a dangling face.

4.2 Solid extraction

We can extract a solid from a non-manifold complex shell when the shell incorpolates a closed shell as a component. First, we check whether the shell incorpolates a closed shell or not. This is done by recursive traversal through the data structure similar to the case of shell separation in Sec. 3.4. If the shell has a *boundary edge* where mated *half-edges* belong to the same face entity, it is not a closed shell. So, we start from an arbitrary *half-edge* and traverse all the connected *half-edges* checking for the existence of a boundary edge. We continue this until all the *half-edges* are checked. Next, we separate the closed shell from the non-manifold shell. This is done in a manner similar to the case of a separation in the set operation for the *half-edges* belonging to a closed shell.

By way of example, Fig. 17 shows the separation of a solid by a free-form surface. In this example, a user specifies that an inner part of a solid is effective for the freedom of the shell connection, and invokes only the solid extraction command. Fig. 18 shows generation of a solid by combining surfaces. An extracted closed shell and the remaining non-manifold shell are generated automatically.

5. Summary

We have proposed a shell operation concept and its algorithm in an integrated framework to enable designers to generate a solid freely combining solids and surfaces. The algorithm is simple, efficient and common to any types of two-dimensional shells as follows:

- Three-dimensional regions are represented in the cycle structure by vertex and edge cycles of shells which are boundaries of the regions. Only shells are manipulated for generating solids uniformly by shell operations, i.e., combining and separating shells.
- The shell operations maintain the topological consistency of shells utilizing geometrical routines and Euler operators. The geometrical routines detect all the intersections of shells, and determine geometric entities to be generated and their topological relations. The Euler operators corresponding to a non-manifold Euler-Poincar é formula modify the data structure.

We are presently conducting further research on the algorithms for a rigorous shell operation to prevent numerical errors and to allow designers to modify the model as they like by direct removal and parameter changes of form-features.

References

1. E.L. Gursoz, Y. Choi, F.B. Prinz: Vertex-based representaton of non-manifold boundaries, in: M.J. Wozny, J.U. Turner, K. Preiss (eds), *Geometric Modeling for Product Engineering*, North-Holland (1990), pp.107-130
2. E.L. Gursoz, Y. Choi, F.B. Prinz: Boolean set operations on non-manifold boundary representation objects, *Computer-aided Design*, Vol. 23, No. 1 (1991), pp.33-39
3. M. Higashi, T. Mori, M. Hosaka: Interference Calculation of Surfaces based on their Geometric Properties, in: F. Kimura, A. Rolstadas (eds), *Computer Applications in Production and Engineering CAPE'89*, North-Holland (1989), pp.275-282
4. M. Mäntylä: *An Introduction to Solid Modeling*, Computer Science Press (1988)
5. M. Masuda, et al.: A mathematical theory and applications of non-manifold geometric modeling, in: F.-L. Krause, H. Jansen (eds), *Advanced Geometric Modelling for Engineering Applications*, North-Holland (1990), pp.78-92
6. A.A.G. Requicha, J.R. Rossignac: Solid Modeling and Beyond, *IEEE Computer Graphics and Application*, Vol. 12, No. 5 (1992), pp.25-37
7. J.R. Rossignac, M.A. O'Connor: SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries, in: M.J. Wozny, J.U. Turner, K. Preiss (eds), *Geometric Modeling for Product Engineering*, North-Holland (1990), pp.145-180
8. J.R. Rossignac, A.A.G. Requicha: Constructive non-regularized geometry, *Computer-aided Design*, Vol. 23, No. 1 (1991), pp.21-32
9. I. Stroud: Modelling with degenerate objects, *Computer-aided Design*, Vol. 22, No. 6 (1990), pp.344-351
10. K. Weiler: Edge-based data structures for solid modeling in curved-surface environments, *IEEE Computer Graphics and Application*, Vol. 5, No. 1 (1985), pp.21-40
11. K. Weiler: The radial edge structure: A topological representation for non-manifold geometric boundary modeling, in: M.J. Wozny, H.W. McLauglin, J.L. Encarnacao (eds), *Geometric Modeling for CAD Applications*, North-Holland (1988), pp.3-36
12. K. Weiler: Boundary graph operators for non-manifold geometric modeling topology representations, in: M.J. Wozny, H.W. McLauglin, J.L. Encarnacao (eds), *Geometric Modeling for CAD Applications*, North-Holland (1988), pp.37-66
13. K. Weiler, D. McLachlan: Selection sets and filters in geometic modeling and their application in a non-maninifold environment, in: J. Turner, J. Pegna, M. Wozny (eds), *Product Modeling for Computer-Aided Design and Manufacturing*, North-Holland (1991), pp.117-139
14. Y. Yamaguti, K. Kobayashi, F. Kimura: Geometric modeling with generalized topology and geometry for product engineering, in: J. Turner, J. Pegna, M. Wozny (eds), *Product Modeling for Computer-Aided Design and Manufacturing*, North-Holland (1991), pp.97-115