Chapter 2

Basic Concepts

This chapter reviews the conceptual operations one expects to be present in the user interface. Also discussed are constructive solid geometry and boundary representation, two major representation schemata used in solid modeling.

In constructive solid geometry (CSG) a solid is represented as a set-theoretic Boolean expression of *primitive* solid objects, of a simpler structure. Both the surface and the interior of an object are defined, albeit implicitly. A boundary representation (B-rep), on the other hand, describes only the oriented surface of a solid as a data structure composed of vertices, edges, and faces. The orientation convention permits us to decide on which side of the surface the solid's interior is located.[1] This suffices to describe the solid's interior and exterior unambiguously, provided the surface and its geometric embedding satisfy certain geometric and topological requirements.

CSG and B-rep have different inherent strengths and weaknesses. For instance, a CSG object is always valid in the sense that its surface is closed and orientable and encloses a volume, provided the primitives are valid in this sense. A B-rep object, on the other hand, is easily rendered on a graphic display system. In consequence, there is a discernible tendency to combine both CSG and B-rep in an effort to take advantage of the different strong points afforded by each. Such modelers are called *dual-representation* modelers.

[1] If only objects of bounded volume are represented, then a surface orientation is unnecessary. However, we shall permit unbounded objects.

To develop an understanding of the properties and algorithmic aspects of these representations, we describe some of the basic operations on them. For CSG, these include classifying points, curves, and surfaces with respect to a solid; detecting redundancies in the representation; and approximating CSG objects systematically. For B-rep, we review the possible surface types, the winged-edge representation schema, and the Euler operators. A more flexible B-rep schema is given later in Chapter 3, where we discuss how to intersect two polyhedra given in this representation.

Given a boundary representation, the question of whether it represents a solid is of obvious practical interest. Algorithms for testing topological validity can be given, but should be based on precise mathematical definitions. We develop formal definitions of what constitutes a valid solid in the topological sense, and derive from it a validity check. This material is intricate and uses methods from algebraic topology. Although the intuitive content is fairly obvious, it is necessary to develop the material carefully, since there are many subtleties that are not apparent at first glance.

There are other solid representation schemata based on spatial subdivision; for example, octrees. We comment on them briefly at the end of the chapter, but do not go into the algorithmic aspects entailed by them.

2.1 Conceptual Operations and Primitives

Irrespective of the representation schema chosen, we must make available conceptual tools for defining objects, for modifying them, and, eventually, for archiving them. We discuss these operations now.

2.1.1 Primitives

A solid design is usually created in several steps that begin with an existing design and modify it, or create a new design from *primitive* objects. The former situation presupposes an earlier design that is retrieved from a database. The latter situation depends on a suitable notion of what constitutes a primitive.

Primitive objects are selected from a universe of possible shapes. A shape is instantiated by assigning values to certain parameters. Some systems allow delaying parameter assignment. We give three examples of primitive object definition.

1. Each primitive is selected from a set of solid shapes and is instantiated by choosing values for certain dimensioning parameters that control the final shape. For instance, a CSG modeler may use blocks, cylinders, spheres, cones, and tori. The parameters in this case include the side lengths of blocks, the diameter and length of cylinders, and so on.

2. A primitive is created by sweeping a contour along a space curve. Both the shape of the contour and the shape of the space curve are defined

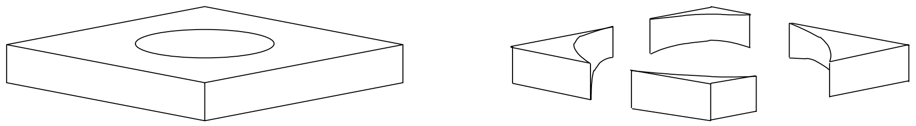

Figure 2.1 Shape Variation Due to Parameter Values

by parameters. For instance, we can sweep a disk of radius r along a line segment of length l, thus creating a solid cylinder. This approach lends itself to generating and verifying cutting paths for numerically controlled machining.

3. All primitives are algebraic halfspaces; that is, point sets defined as

$$\{(x, y, z) \mid f(x, y, z) \leq 0\}$$

where f is an irreducible polynomial.[2] The coefficients of the polynomial can be considered the shape parameters. This approach has been used in several research systems.

We will discuss the first approach in the section on constructive solid geometry (CSG). Note that, in a pure CSG modeler, the instantiation of the shape parameters can be delayed. It is then possible to construct *generic designs*. However, a generic design cannot be displayed or converted to boundary representation, since different parameter assignments could lead to totally different shapes. See also Figure 2.1, where we have varied the diameter of the cylinder defining the hole. In some modelers, the parameters carry default values that can be used to visualize generic designs.

In the second approach, various elementary operations have been proposed for creating primitive solids. A typical example is *sweeping*: We are given an object to be swept, and a path along which to sweep it, and thereby we define some volume. The object S to be swept could be a finite area delimited by a closed curve, or a solid. The path of the sweep typically would be a segment of a space curve C, and could be open or closed. The primitive solid created by this operation consists of the volume swept by S as it is moved along C. An example is shown in Figure 2.2.

The mathematics of sweeping is more delicate and demanding than it might seem at first glance. Foremost, it depends on certain conventions. We

[2]As explained in Chapters 5 and 7, a polynomial is *irreducible* if the point set defined in the text cannot be decomposed into the union of simpler components. Technically, the polynomial f is irreducible if it cannot be factored.

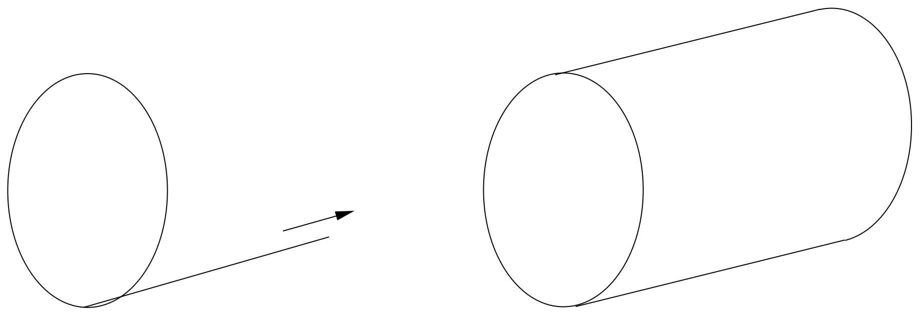

Figure 2.2 A Circular Disk Swept Along a Line

need to fix a reference point on S that will traverse the curve C. We also must define how, if at all, the orientation of S varies as S moves along C, and what the initial orientation is. These conventions must be determined by default or by suitable parameters to the sweep operations. Then, there could be degeneracy problems: If a planar area S with fixed orientation is moved along a path C that has a tangent parallel to the plane containing S, then the interior of the resulting volume could have self-intersections. In the three-dimensional example shown in Figure 2.3, this is indicated by the crease in the center that is the result of a self-intersection. Figure 6.6 in Chapter 6 shows a two-dimensional example. Moreover, if the entire path C is parallel to the plane of S, then we have defined an area instead of a volume. If such cases are considered an error, we need algorithms for their detection. If they are allowed, we need to give proper meaning to the results.

Usually, there is no closed-form mathematical description of the surface bounding the swept volume. For example, the cylinder in Figure 2.2 is bounded by finite areas on two linear surfaces and on one quadratic surface. Moreover, if the surface contains self-intersections or other types of singularities, the areas of interest may not have a simple definition.

Important special cases include sweeping a sphere along a space curve or across a surface of another solid. In the first case, we obtain a partial description of the volume removed by a ball cutter as the cutting tool is

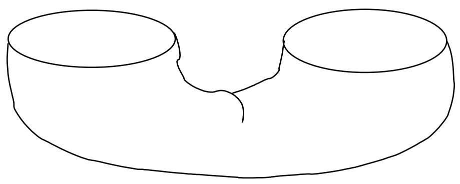

Figure 2.3 Sweep Degeneracies

moved along the path of the sweep.[3] Thus, this case has applications in numerically controlled machining, and can be used to represent the effect of cutting operations, either for automatic generation of cutter paths or for verification that such paths do not interfere with other parts of the solid to be manufactured.

When a sphere is moved across a surface, we obtain a volume bounded by the *offset* of the surface. Offsetting can be viewed as an operation on solids or on surfaces, and has been used to define global *blending* operations on solids in which all edges are rounded or filleted. We return to the subject of offsets and spherical sweeps in Chapters 6 and 7.

The third approach of using algebraic halfspaces as primitives raises difficult algorithmic problems and is the subject of current research. Unless additional restrictions are placed on f, the generality of the primitives can be overwhelming, and general algorithms such as the ones discussed in Chapter 7 should be considered.

Note that variations of the third approach have also been used. For example, we could require that the polynomial f have degree no greater than 2 or 3. Doing so has the advantage that the specialized techniques discussed in Chapter 5 suffice to manipulate the resulting objects.

2.1.2 Local Modifications

Numerous local modifications to solids have been proposed. Most of them operate on a boundary representation, and can be implemented using Euler operations (see also Section 2.3.4). Figures 2.4 through 2.6 show several examples.

If we operate on boundary representations with simple shape elements, then local modifications could be inexpensive, provided that the geometric shapes we manipulate are sufficiently simple. Local modifications do, however, require validity checks to avoid errors such as the one shown in Figure 2.7. Here, the face is extruded too far and interferes with other parts of the solid.

[3]Strictly speaking, a ball cutter cannot physically remove the entire volume swept by the sphere, since we must accommodate the shaft of the cutter.

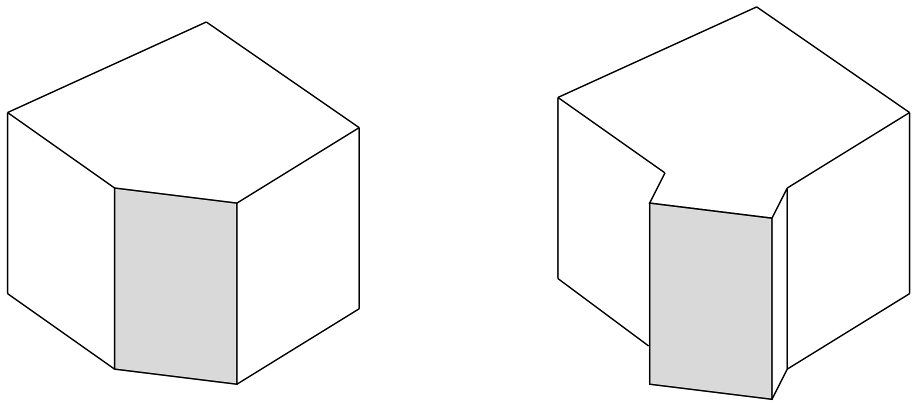

Figure 2.4 Extruding a Face

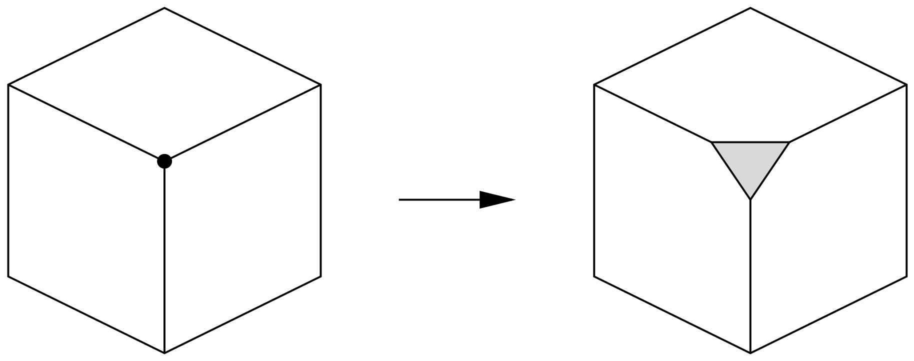

Figure 2.5 Beveling a Vertex

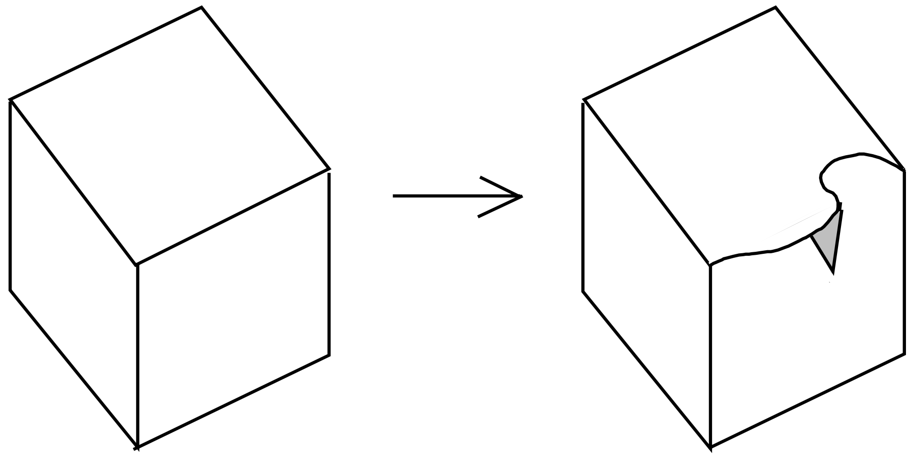

Figure 2.6 Altering Edge and Face Shape

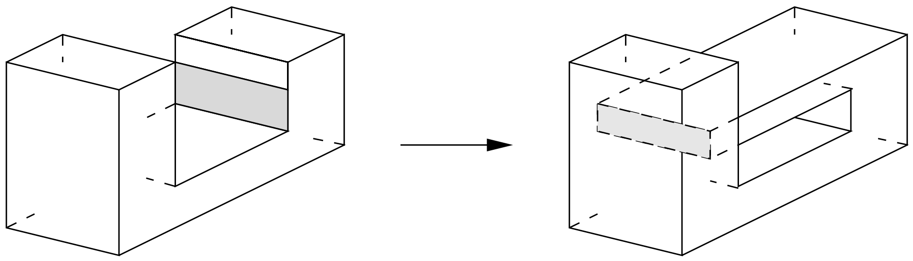

Figure 2.7 Error in Face Extrusion

2.1.3 Global Operations

There are several trivial global operations, including rotating and translating solids. Regularized Boolean operations, explained in Section 2.2, are also considered global operations, as are the operations of offsetting solids, and of rounding all convex and filleting all concave edges.

2.1.4 Undoing and Redoing

Ideally, solid design is an interactive process in which the designer experiments with alternatives, modifies them, corrects errors, and so on. Whereas interactivity demands quick response time, exploring alternative designs and correcting errors requires the possibility of undoing a sequence of operations, or redoing some of them.

Undoing an operation requires minimally a *history* that records all operations leading, in sequence, to the present design. Then, in principle, we could reconstruct the entire sequence of operations, from the beginning up to some prior point. Effectively, this undoes all subsequent operations. If alternatives should be explored and if we wish to return to some of them, then the history record must be a tree. An example of a history tree of designs is shown in Figure 2.8.

Redoing a design from the root up to a specific alternative is inferior to using a more direct *undo* capable of reversing the effect of an operation. The difficulty of the undo will depend on the way objects are represented. The operation is easy in pure CSG. In boundary representation, local modifications are easy to undo, but Boolean operations are not. For difficult undo operations, it is better to *check point*; that is, we store the representation of the design prior to the operation, and then once more after its completion. Then, undoing is simply a retrieval.

The cost of check pointing has to be balanced against the cost of inverting the operation. If an undo is cheap, it is probably better not to check point, especially in B-rep, where the data structures describing the current design may be very large.

Having undone a partial design, we may wish to reconstruct a design alternative previously defined. Since most operations include consistency

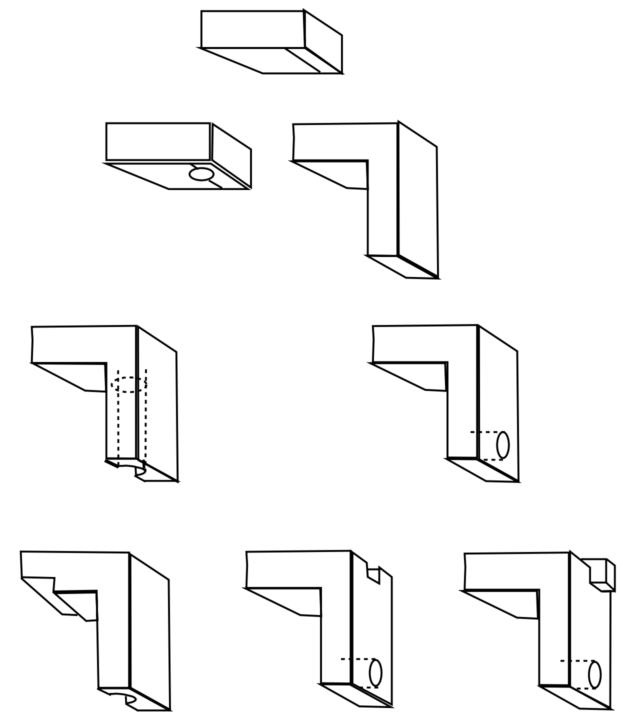

Figure 2.8 History Tree of Design

$$\begin{tikzcd}
 & (0.1) \arrow[dl] \arrow[dr] & \\
(1.2) \arrow[d] & & (1.1) \arrow[d] \\
(2.1) \arrow[d] & & (2.2) \arrow[d] \\
(3.1) & (3.3) & (3.2)
\end{tikzcd}$$

Figure 2.9 Indexing a History Tree

checking for their results, an explicit *redo* operation has the advantage that such checks are not needed. Thus, a redo operation can be faster. Moreover, for expensive operations such as union or intersection, redo can simply access the check-pointed representation and is therefore also cheap.

The history tree should be presented to the user as an aid to remember the various versions of design already explored. The presentation can be augmented by an indexing scheme that generates default names for the design based on the position in the history tree. Thus, instead of issuing a sequence of undo and redo commands to reach a specific alternative, we can retrieve the alternative directly by giving its name. A simple indexing scheme is as follows:

> Each tree vertex is indexed by a pair of numbers (i, j), where i is the depth of the vertex, and j enumerates all vertices of equal depth in the order of creation.

The *depth* of a vertex is determined as follows: The root has depth 0; if v descends from a vertex w, then depth(v) = depth$(w) + 1$. Figure 2.9 shows an example.

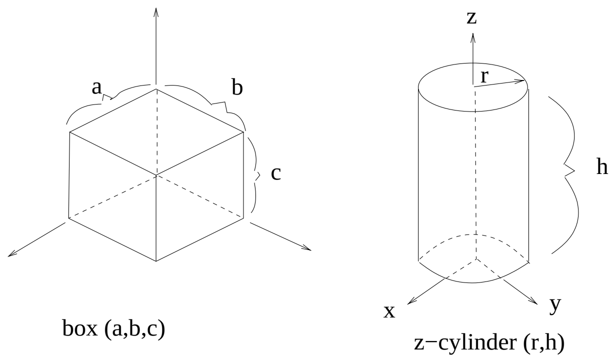

Figure 2.10 Coordinate Frames for Two Standard Primitives

2.2 CSG Representation

In the strict sense, CSG is a method of representation, a design methodology, and a certain standard set of primitive objects. So, a CSG object is "built" from the *standard primitives*, using regularized Boolean operations and rigid motions. We will sketch this methodology first, and will present some of the properties and algorithms it entails. Later on, we will consider the possibility of greatly enlarging the set of allowed primitives.

2.2.1 CSG Standard Primitives

The CSG standard primitives are the parallelepiped (block), the triangular prism, the sphere, the cylinder, the cone, and the torus.[4] They are *generic* in the sense that they represent shapes that must be *instantiated* by the user to chosen dimensions. Thus, to obtain a parallelepiped of edge lengths 1, 1, and 3, one might specify `block`$(1, 1, 3)$, where the lengths are expressed in units depending on conventions, or, perhaps, are given explicitly. Also depending on convention would be the placement of the resulting object in space: With each primitive object there is associated a *local coordinate frame*. Here, we will place this coordinate frame as shown in Figure 2.10. These different local coordinate frames must be related to one another, by placing them with respect to a common *world coordinate frame*, discussed later.

All standard primitives have a finite domain. For example, the cylinder always has a finite radius and a finite length. This convention seems to be rooted in the thought that we always model finite solids. We will see later that it can be convenient to consider infinite solids, at least as intermediate steps, in the process of defining complex, finite solids.

[4]Note that the prism or the parallelepiped is redundant.

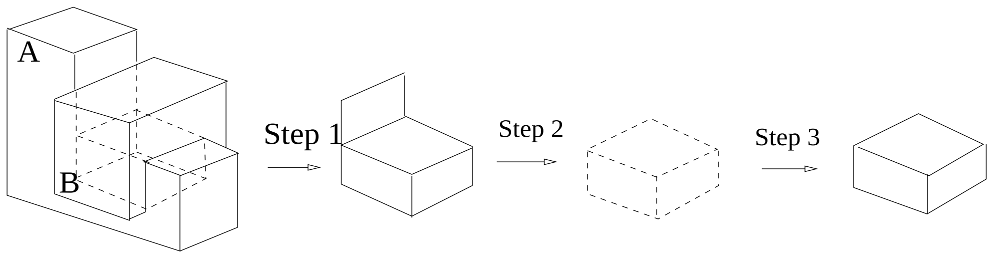

Figure 2.11 Procedure for Regularized Intersection

2.2.2 Regularized Boolean Operations

After instantiation, primitive objects can be combined using *regularized Boolean operations*. The operations are the *regularized union*, denoted $\cup^*$; *regularized intersection*, denoted $\cap^*$; and *regularized difference*, denoted $-^*$. They differ from the corresponding set-theoretic operations in that the result is the closure of the operation on the interior of the two solids, and they are used to eliminate "dangling" lower-dimensional structures. For example, to compute $A \cap^* B$, we proceed conceptually as follows:

1. We compute $A \cap B$ in the set-theoretic sense. The result is a collection of volumes, and additional faces, edges, and vertices. These additional faces, edges, and vertices are lower-dimensional structures that we will eliminate.

2. We now take the *interior* of $A \cap B$. The interior consists of all those points $p \in A \cap B$ such that an open ball of radius ϵ, centered at p, consists only of points of $A \cap B$, for a sufficiently small radius ϵ.

3. We form the *closure* of this interior, by adding all boundary points adjacent to some interior neighborhood. A point q that is not an interior point of $A \cap B$ is *adjacent* to the interior if we can find a curve segment (q, r) of sufficiently small length ϵ, between q and another point r of $A \cap B$, such that all points of this segment are interior points of $A \cap B$, except q. Note that the lower-dimensional structures do not enclose volume and are therefore not adjacent to the interior of $A \cap B$.

The resulting solid is the regularized intersection. Figure 2.11 illustrates the procedure.

Note that, in practice, regularized Boolean operations are not implemented in this manner. Rather, $A \cap^* B$ is implemented by classifying the surface elements of $A \cap B$ and eliminating lower-dimensional structures. This explicit classification is delayed until a geometric query requires it, as explained later, or until a conversion from CSG to B-rep is carried out; see also Section 2.2.6.

Eliminating the lower-dimensional structures is desirable for defining solids. However, in some applications, it may be desirable to retain them, possibly even in the interior of objects. For example, when considering solids as *domains* in finite element analysis, interior lower-dimensional structures might represent certain constraints on how to discretize the domain, or might define the domain discretization outright. At present, this is a research topic, and we do not explore this line of thought further.

Before the two objects are intersected, they must be positioned appropriately with respect to each other. This is done by translations and rotations, as needed. To make this positioning meaningful, we must establish a relationship between the local coordinate frames of the objects. A simple method is to identify the local frames with a single, universal coordinate frame. The universal frame is often referred to as the *world coordinate frame.*

Suppose we have positioned the two primitives, and have constructed an intersection. Then, the resulting object should have a local coordinate frame of its own, needed for subsequent positioning operations we might wish to perform. By convention, we will use (a copy of) the world coordinate frame for this purpose.

2.2.3 Construction of a CSG Object

The CSG representation of the simple bracket shown in Figure 2.12 is easily worked out. We think of the bracket as the union of two blocks of respective dimensions (1,4,8) and (8,4,1) with the hole subtracted by a cylinder of radius 1. Without the hole, we can specify the bracket as

$$\texttt{block}(1,4,8) \cup^* \texttt{x-translate}(\texttt{block}(8,4,1),1)$$

The hole is removed by subtracting a cylinder about the z axis, resulting in the expression

$$(\texttt{block}(1,4,8) \cup^* \texttt{x-translate}(\texttt{block}(8,4,1),1)) -^* \\ \texttt{x-translate}(\texttt{y-translate}(\texttt{z-cylinder}(1,1),2),5)$$

The expression is conveniently drawn as a tree, as shown in Figure 2.13. This tree can be considered to be the *representation* of the object, and is customarily called a *CSG tree.* We see that the leaves of the CSG tree are primitive solids, and the interior nodes are rigid motions and Boolean operations.

In our example, the two blocks joined are touching, and the cylinder length matches the bracket thickness. In practice, this is an unsafe specification for nonintegral dimensions because of the possibility of floating-point inaccuracies. It is thus advisable to allow for a safe amount of overlap when specifying union operations. Here, then, is one place where substratum problems have intruded into the higher design levels.

2.2.4 Point/Solid Classification and Neighborhoods

Having built a CSG object, we might wish to interrogate its geometry in various ways. The most elementary such query is to test whether a point

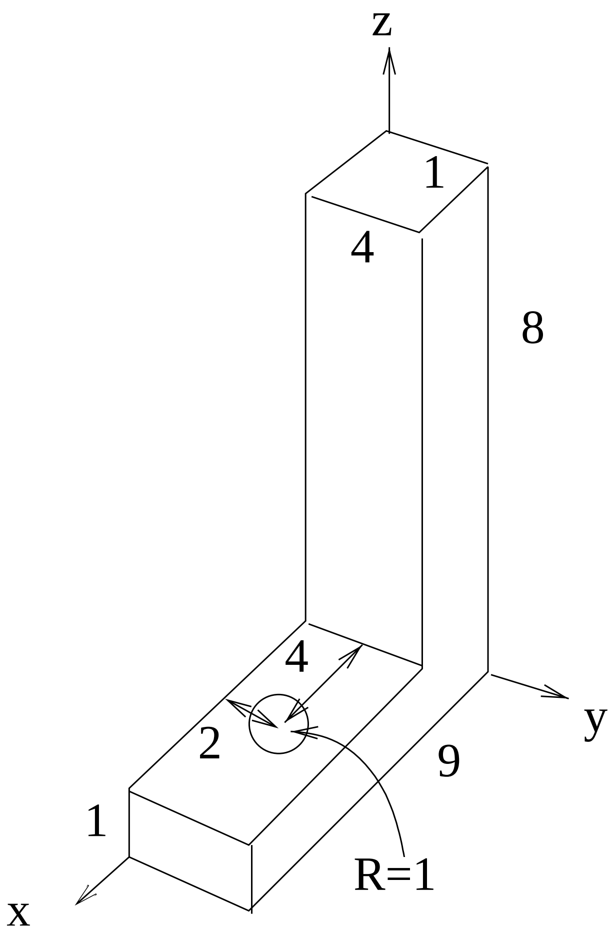

Figure 2.12 Bracket

(x, y, z) is inside a solid, is on its surface, or is outside of it. This query is usually referred to as a *point/solid classification.* Other such queries include a classification of how a line intersects a solid, a classification of how a surface intersects a solid, and a test of whether two solids intersect in a nonempty volume. These operations will be discussed later.

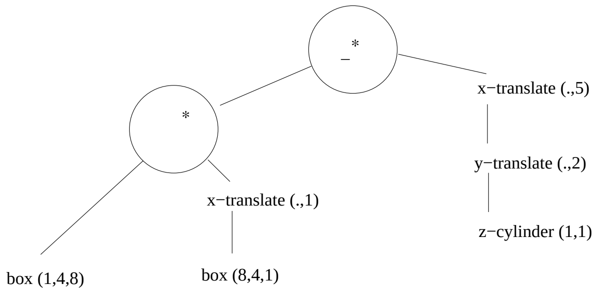

Figure 2.13 Tree Representation of CSG Expression

Point/solid classification can be done with an algorithm that has a simple conceptual structure. Despite its apparent straightforwardness, however, we soon realize that a difficulty may arise when the point lies on the surface of a primitive, and this difficulty necessitates the introduction of *neighborhoods*.

The basic idea underlying this and other such algorithms is to reduce the point/solid classification to a query of the primitives in the CSG tree. The respective answers, one for each primitive, are then collated at each operation node as appropriate. Thus, the algorithm is based on the *divide-and-conquer* paradigm familiar from the literature.

Downward Propagation

Point/solid classification is naturally implemented as a set of recursive procedures, but it might be simpler to think of it as passing messages between the tree nodes. At the outset, the point coordinates are sent to the root of the tree. From there, they are propagated into the tree down to the leaves, possibly altered. At each leaf, the final coordinates describe the same point, but with respect to the local coordinate frame of the primitive solid that the leaf represents.

At the leaf, we classify the point as one of *in*, *on*, or *out*, depending on whether the point is, respectively, in the interior, on the surface, or on the outside of the primitive solid. This classification is passed back up the tree, to the root. At an operation node, the results from the subtree are coordinated. So, we specify the first phase of the algorithm as follows.

1. If (x, y, z) arrives at a node specifying a Boolean operation, then it is passed unchanged to the two descendants of the node.

2. If (x, y, z) arrives at a node specifying a translation or rotation, the inverse translation or rotation is applied to (x, y, z), yielding a new point (x', y', z'), which is sent to the node's descendant.

3. If (x, y, z) arrives at a leaf, then the point is classified with respect to that primitive solid, and the classification is returned to the parent of the leaf.

When classifying the point $(2, 1, 0.3)$ with respect to the bracket, for instance, we classify the point $(2, 1, 0.3)$ with respect to `block`$(1, 4, 8)$, the point $(1, 1, 0.3)$ with respect to `block`$(8, 4, 1)$, and the point $(-3, -1, 0.3)$ with respect to `z-cylinder`$(1, 1)$. The respective classifications are *out*, *in*, and *out*.

Upward Propagation

In the second phase of the algorithm, the messages contain point classifications that must be combined at the Boolean operation nodes. No work is done at nodes representing translation or rotation. Table 2.1 shows what to do for union and intersection operation nodes.

$\cup^*$	*in*	*on*	*out*
in	*in*	*in*	*in*
on	*in*	*on?*	*on*
out	*in*	*on*	*out*

$\cap^*$	*in*	*on*	*out*
in	*in*	*on*	*out*
on	*on*	*on?*	*out*
out	*out*	*out*	*out*

Table 2.1 Naive Neighborhood Combination for Union and Intersection

Neighborhoods

Implemented in this way, the algorithm will be incorrect. For example, classifying the point $(1, 1, 0.5)$ with respect to the bracket yields an incorrect *on*. The problem here is the classification of points that lie on the surface of a primitive solid. These points may lie on a primitive surface area that remains a part of the surface of the solid described by the tree, and then using the table yields the correct result. If, however, the point is on a surface area that is not on the final surface — for example, because it becomes solid interior as the point $(1, 1, 0.5)$ does — then the tables do not suffice. What is needed in addition to the classification as one of *in*, *on*, or *out* is the local geometry of the solid in the vicinity of the point. The additional information is given by a neighborhood of the point, as explained next.

A *neighborhood* of a point $p = (x, y, z)$, with respect to the solid S, is the intersection with S of an open ball of infinitesimal radius ϵ centered at p. We used this concept to define the interior of a solid, and recall that p is *inside* S, iff the neighborhood is a full ball. The point p is on the *outside*, iff the neighborhood is an empty ball. If p is on the surface of S, then the structure of the neighborhood depends on the local topology of S at p. We explain the possible topologies of these neighborhoods by restricting the local geometry to planar surfaces; that is, by considering only polyhedra for the moment.

We decompose the surface of the solid into faces, edges, and vertices. Here, a *face* is a closed subset of the surface all of whose points lie in the same plane.[5] An *edge* is the intersection of two adjacent faces, and a *vertex* is the common intersection of three or more faces. For example, the surface of a cube consists of 6 faces, 12 edges, and 8 vertices.

If a surface point is in the interior of a face, then its neighborhood is a *halfspace* whose surface in the ball is a subset of the face. In Figure 2.14, the ball neighborhood of such a point is shown, and the solid part of the neighborhood has been shaded. Next, consider a point on an edge different from the two vertices. In the simplest case, the edge is adjacent to exactly two faces, so the neighborhood is a *wedge*. For some CSG objects, however, it

[5]Strictly speaking, we may have to split the closed subset into two maximal components such that the solid interior lies locally on the same side of the plane, for each component. In this case, each component is a separate face.

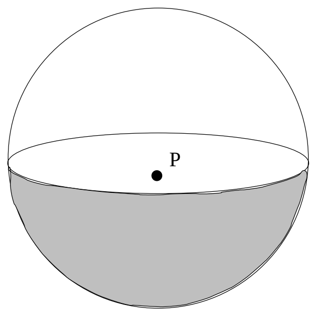

Figure 2.14 Neighborhood of an Interior Face Point

is possible that an edge is adjacent to an even number of faces that is greater than two. In that case, the neighborhood of the point is a union of such wedges, all with the edge in common, as exemplified in Figure 2.15. Again, the solid part of the neighborhood is indicated by the shading.

Finally, consider a vertex. Again the simplest case is when all faces incident to the vertex are edge adjacent in a single cycle. In that case, the vertex neighborhood is a *cone*. Some possibilities are shown in Figure 2.16. In general, the faces incident to a vertex are organized in several cycles. In this general form, the vertex neighborhood consists of a collection of cones, possibly with conical holes and touching along certain edges. All cones have the vertex as common apex; see also Figure 2.17 for an example. Such a neighborhood can be represented as a set of curves on the surface of a sphere. The curves represent the intersection of the cone surfaces with the sphere, and the resulting map on the sphere is two-colorable, with one color representing solid interior, the other representing solid exterior.

In the case of curved surface elements, the neighborhood structure remains topologically the same as in the polyhedral case, but the geometric structure is more complicated. Often, we can approximate the curved surfaces with the tangent planes at p. However, in situations where surface elements match and combine in ways that alter the topology qualitatively, we must consider the curved-surface geometry. Some of these situations are discussed in the next section.

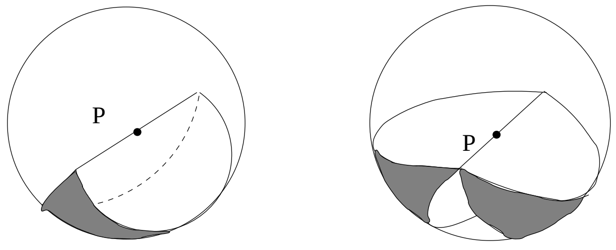

Figure 2.15 Neighborhood of an Interior Edge Point

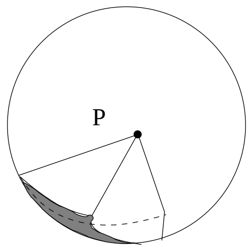

Figure 2.16 Simple Neighborhoods of a Vertex

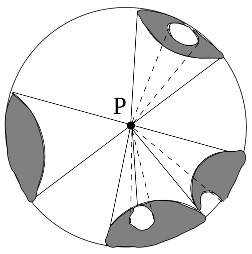

Figure 2.17 General Neighborhood of a Vertex

Refined Upward Propagation

The problem with Table 2.1 is that no geometric information on the neighborhood structure is taken into consideration. Thus, although the union of two halfspaces in general forms a wedge, it may remain a halfspace or become a full solid ball. Since this geometric information is ignored, the tables cannot always produce the correct answer.

Thus, to repair our method for processing the information during the second phase of point/solid classification, we must perform the respective Boolean operation on the neighborhoods themselves. Only then do we obtain correct answers. This requires accounting for the local geometry, devising suitable data structures to represent neighborhoods, and transforming the geometric data appropriately at the rigid motion nodes in the tree. Again, we consider the polyhedral case first.

Representing the neighborhoods of interior or exterior points is trivial. So, let p be a point on the surface of the solid defined by a subtree. If p is in the interior of a face, then the neighborhood can be represented by the plane equation of the face, oriented such that the plane normal points to the exterior of the halfspace. If p is on the interior of an edge, then the neighborhood is represented by a set of sectors in a plane containing p that is perpendicular to the edge. Vertex neighborhoods, finally, are inferred from the adjacent edge neighborhoods. When performing Boolean operations on boundary representations, it will again be useful to think in terms of neighborhoods, so we will discuss this subject again in the next chapter.

At a union node, we must compute the union of the two neighborhoods of p that reach the node from its left and right descendants. Except for the trivial cases where one or the other neighborhood is the empty or the full ball, we must merge the two data structures and inspect the result. We describe this procedure conceptually.

Essentially, the following rules apply for merging neighborhoods at a union node. Let N_L and N_R be the two neighborhoods at the descendants to the left and to the right. Then the neighborhood N at the node is as follows:

1. If N_L is the full ball, then $N = N_L$. If N_L is the empty ball, then $N = N_R$.

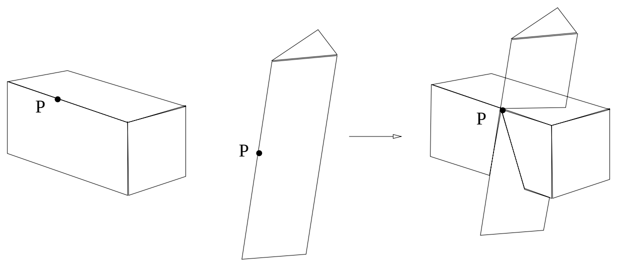

Figure 2.18 Edge-Neighborhood Merge, General Position

2. If N_L and N_R are face neighborhoods, then N is an edge neighborhood unless the two faces coincide. This case includes coplanar faces; then, N will be a face neighborhood or the full ball, depending on how the faces are oriented.

3. If N_L and N_R are edge neighborhoods, then N is in general a vertex neighborhood whose cones are formed from the wedges of N_L and N_R; see Figure 2.18. If the edges coincide, N will be an edge neighborhood, unless the wedges match up to form a single face with p in the interior; see also Figures 2.19 and 2.20.

4. If N_L is a vertex neighborhood, and N_R an edge neighborhood, then N is a vertex neighborhood unless each of its solid cones is contained in a wedge of N_R.

5. If N_L and N_R are vertex neighborhoods, then N is a vertex neighborhood, as shown in Figure 2.21, unless the cones match up to form wedges or a face with p in the interior; see also Figures 2.22 and 2.23.

The remaining cases can be worked out easily, and analogous rules are formulated for the other Boolean operations.

Clearly, the geometric processing required to cover all cases is not trivial, even when we restrict our attention to polyhedral objects only. The vertex-neighborhood merge is inherently complicated because the neighborhood structure can be complex. The other cases gain in complexity because of the exceptions that arise when the various geometric elements are in special positions with respect to one another.

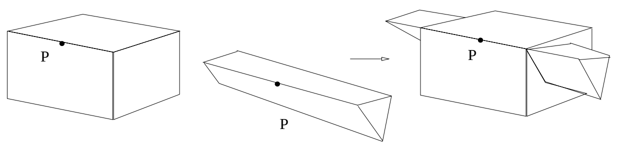

Figure 2.19 Edge-Neighborhood Merge Producing an Edge

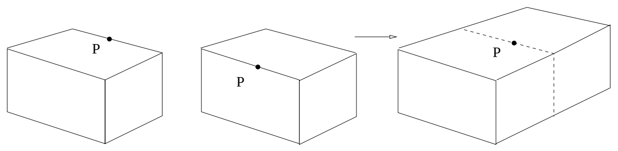

Figure 2.20 Edge-Neighborhood Merge Producing a Face

2.2.5 Curve/Solid Classification

A useful interrogation primitive is the classification of a space curve against a solid. The special case of the straight line can be used to generate shaded images as follows. Consider a line through the view point and a screen pixel. Classify that line against the solid, pick the nearest intersection point, and, recalling which primitive is intersected at that point, compute the intensity from the surface normal and the lighting information. Since this application uses the algorithm a large number of times, it is important to implement it as efficiently as possible. The approximation techniques discussed here provide additional strategies for speeding up the computations.

The algorithm for classifying a line or curve against the solid is organized exactly like the point/solid classification.

1. Send the line or curve description to the leaves. Partition the curve into segments labeled *inside*, *outside*, or *on* the surface of the primitive.

2. Propagate the segments back upward, and merge them appropriately.

To classify a line against a primitive, we may parameterize the line and substitute the parametric form into the implicit surface equations bounding the primitive, thereby deriving a polynomial in one variable for each surface. The roots of the polynomial define the intersection points. Only those points that lie on the primitive are considered further. The points are then sorted along the line and are paired into segments with the appropriate labeling.

Example 2.1: We classify a line against a primitive cylinder. The cylinder is `z-cylinder`$(1, 2)$, and the line is

$$\begin{aligned} x &= 2 - 2\lambda \\ y &= 0 \end{aligned}$$

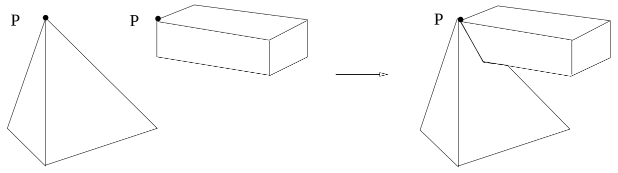

Figure 2.21 Vertex-Neighborhood Merge, General Position

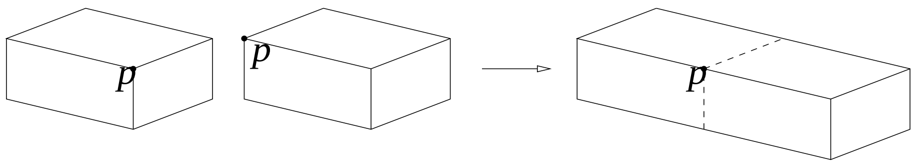

Figure 2.22 Vertex-Neighborhood Merge Producing an Edge

$$z = \lambda$$

The cylinder's perimeter is given by $x^2 + y^2 - 1 = 0$, so the line/perimeter intersection points correspond to the roots of $(2-2\lambda)^2-1 = 0$. The two roots are $\lambda = 1/2$ and $\lambda = 3/2$, corresponding to the points $p = (1, 0, 1/2)$ and $q = (-1, 0, 3/2)$. Both points are on the primitive, since they are above the plane $z = 0$ and below the plane $z = 2$ that bounds the cylinder domain. The intersections of the line with these planes are outside the primitive, and hence are irrelevant. We sort and pair the two intersections found, and conclude that the segment (p, q) is inside the primitive, and the unbounded segments with parametric values $(-\infty, 1/2)$ and $(3/2, +\infty)$ are outside. $\diamond$

Classifying a curve against a primitive can be done in the same way, provided the curve has a parametric form. For such curves, we sort the intersection points by their parameter values. However, even when we consider only those space curves that arise as the intersection of two standard CSG primitives, we need not obtain curves that possess a parameterization. For those curves, more complicated sorting procedures are needed.

Briefly, if the curve lies on a parameterizable surface, then we may equivalently sort the points by sorting the corresponding points in parameter space. That is, instead of considering the point $p = (x(s,t), y(s,t), z(s,t))$ in three-dimensional space, we consider the point $q = (s, t)$ in parameter space. If p is on the intersection with another surface, then q is on a plane algebraic curve C in parameter space. This curve C is considered.

The curve C is decomposed into convex segments not containing any singularities. Points P_k on a specific segment may be sorted by the angle between the secant $\overline{RP_k}$ connecting P_k with a suitable reference point R and a reference direction; see also Figure 2.24. All intersection curves between standard

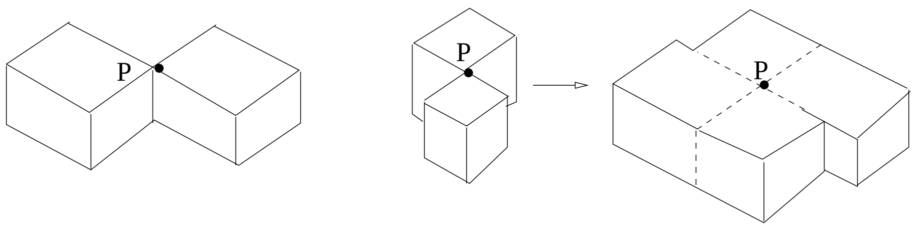

Figure 2.23 Vertex-Neighborhood Merge Producing a Face

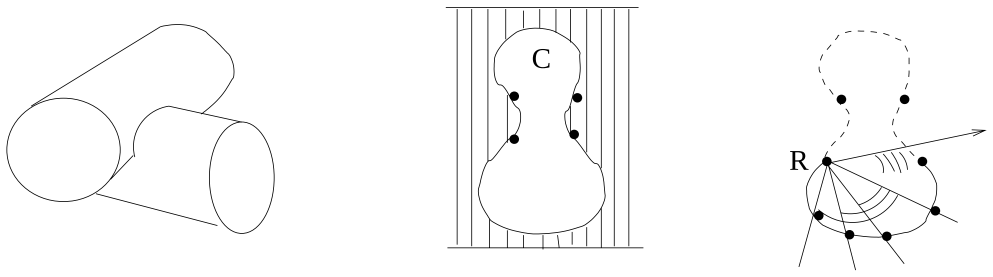

Figure 2.24 Sorting Curve Points on a Parametric Surface

primitives can be processed in this way. See Chapter 6 for information on how to deal with singularities on plane algebraic curves. How to sort points on a general space curve is not well understood.

2.2.6 Surface/Solid Classification, Conversion to B-rep

A surface will intersect a solid in a number of areas. Each such area is bounded by curve segments, where each segment is on the intersection of the surface with one of the primitives of the solid. A general strategy for determining the segments, and from them the respective areas, is therefore as follows:

1. Intersect the surface with each of the primitives from which the solid has been constructed.

2. Classify the resulting curves, thereby determining the bounding edges of those surface areas that are inside or outside the solid, or are on the solid's surface.

3. Combine the segments, appropriately oriented, constructing a boundary representation of the respective surface areas.

Elaboration of this conceptual method leads to many details but is straightforward. The resulting algorithms are similar to, or use outright, the algorithms for point/solid and for curve/solid classification.

Surface/solid classification, in turn, can be used to devise a method for converting from a CSG to a boundary representation. Such a conversion algorithm is based on the *generate-and-test* paradigm: We consider all pairs of intersecting primitives in the CSG object A, obtaining for each a set of space curves in which they intersect. By classifying each curve against the solid, we can determine those segments that are on the surface of A. Each segment will be an edge of the boundary representation. These segments now define, on the surface of the primitives, areas that will be the faces of the boundary representation of A. By considering the neighborhoods, we derive the topological information needed to determine the adjacencies and incidences of the various faces.

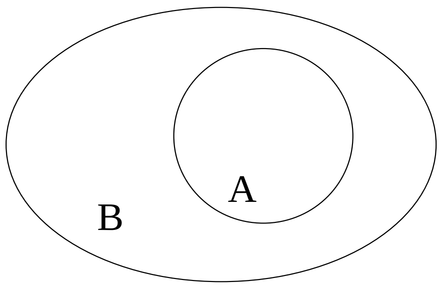

Figure 2.25 A Is Λ-Redundant in $A \cup^* B$

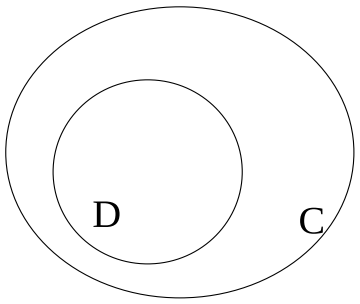

Figure 2.26 C Is Ω-Redundant in $C \cap^* D$

2.2.7 Redundancies and Approximations in CSG Trees

Since a geometric query of a CSG tree grows at least linearly, and in some cases quadratically, with the number of primitives, we investigate whether a given CSG tree contains redundant subtrees that can be eliminated without altering the object defined by the tree. The most blatant redundancy would be a subtree that represents empty space. Such a subtree is said to define the *null object*, Λ, and a detection algorithm for Λ can be used to test whether two CSG objects interfere: Let T_1 and T_2 be two CSG trees defining the objects. Then the two objects do not interfere iff $T_1 \cap^* T_2$ represents the null object.

More generally, a subtree T' of the CSG tree T is *redundant* if replacing T' with the null object Λ, or with the complement Ω of the null object, does not alter the shape defined by T. In the first case, we say that T' is Λ-redundant. In the other case, we say that T' is Ω-redundant.

Example 2.2: In Figure 2.25, the primitive A is Λ-redundant in the CSG expression $A \cup^* B$, because $A \cup^* B = \Lambda \cup^* B$. In Figure 2.26, the primitive C is Ω-redundant in the CSG expression $C \cap^* D$, because $C \cap^* D = \Omega \cap^* D$. $\diamond$

Redundancies arise in contexts other than interference detection. It is possible that a CSG tree T contains redundancies because it was constructed by modification of another CSG tree T_1. Possibly, the object defined by T_1 contains certain parts that are unnecessary for the object defined by T. In such a situation, the designer may simply obliterate the entire unwanted substructure, say by cutting it away using a differencing operation. If the eliminated structure was defined by a complicated subtree in T_1, then that subtree would be redundant.

A general approach to redundancy detection is to approximate CSG objects by enclosing them in simple geometric shapes, and to derive criteria for redundancy based on the approximations. When the approximating shapes are sufficiently simple and are easily constructed, this approach leads to efficient redundancy tests. Based on approximations, however, it can only yield *sufficient* criteria for redundancy. Hence, certain redundancies would remain undetected.

As approximating shapes, we could use spheres or boxes that are oriented in a particular way. The advantage of spheres is that they are invariant under rotation. This would not be true for boxes, whose edges are parallel to the coordinate axes, but there are elegant data structures for using such boxes, as explained in Chapter 3. We describe an approximation algorithm for CSG objects whose structure is independent of the particular choice of approximating shape. However, we shall assume that the CSG object is completely contained within the approximating shape.

We fix a class Σ of approximating shapes. The algorithm begins by approximating all primitives P in the tree with a shape $\sigma(P) \in \Sigma$. By processing the trees from the leaves to the root, we then determine the approximations at all interior nodes by the following three rules.

1. If $T = T_1 \cup^* T_2$, then $\sigma(T) = \sigma(\sigma(T_1) \cup^* \sigma(T_2))$.

2. If $T = T_1 \cap^* T_2$, then $\sigma(T) = \sigma(\sigma(T_1) \cap^* \sigma(T_2))$.

3. If $T = T_1 -^* T_2$, then $\sigma(T) = \sigma(T_1)$.

We eliminate translations and rotations from consideration by distributing them over the leaves. That is, we require that all primitives are positioned with respect to the coordinate system of the final solid. Thus, we need only a method for computing $\sigma(P)$, where P is a primitive, suitably rotated and translated, and an algorithm for approximating the union, intersection, and difference of two approximations. It is now straightforward to show that every point of the object defined by the CSG tree T must be contained in the approximating shape.

Example 2.3: We let Σ be the class of all rectangles whose sides are parallel to the axes. Then the approximation of $(A \cup^* B) -^* (C \cup^* D)$ is as shown in Figure 2.27. The intermediate approximations are also shown. $\diamond$

The approximation algorithm yields a criterion for when a primitive or a subtree in T is redundant. We noted that the approximation at the root contains the entire object. Hence, if T' is any subtree of T, then only points in $\sigma(T') \cap^* \sigma(T)$ can contribute to the object defined by T. In particular, if $\sigma(T') \cap^* \sigma(T) = \emptyset$, then the subtree T' does not contribute to the final shape and can be deleted from T. For example, the primitive D shown in Figure 2.27 is redundant by this criterion, and can be deleted.

2.2.8 Nonstandard Primitives

We can extend the primitives by adding other shapes to our repertoire. For instance, we might add all quadric halfspaces — that is, ellipsoids, paraboloids, hyperboloids, and cylinders and cones with conic base curves. We could require that infinite halfspaces, such as the hyperboloids, be restricted to finite domains, as we did with circular cylinders and cones, or

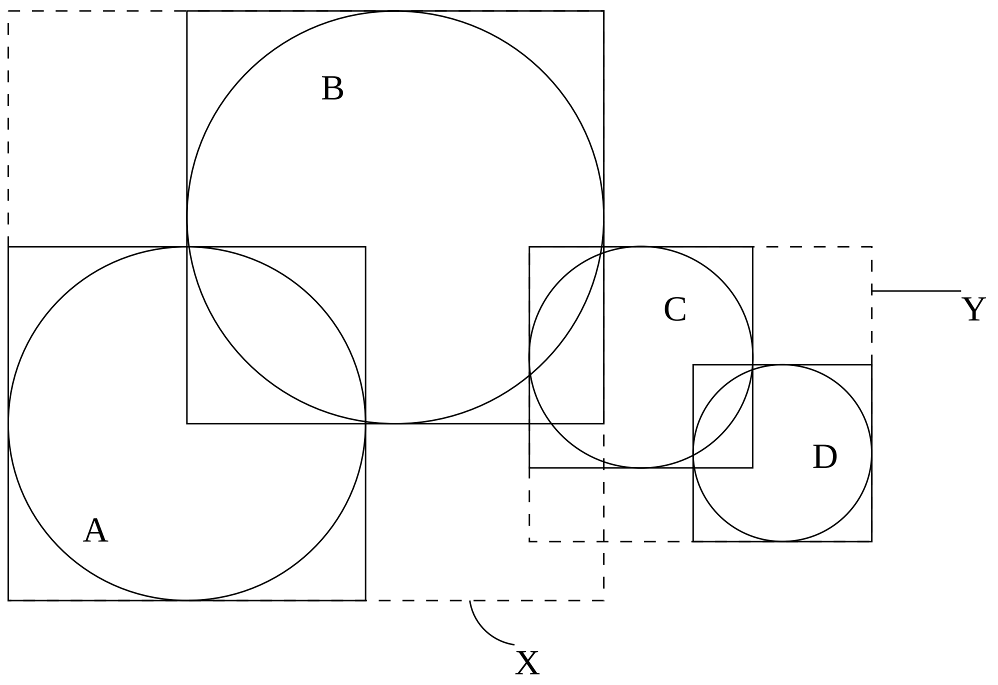

Figure 2.27 Approximation of $(A \cup^* B) -^* (C \cup^* D)$: $Y = \sigma(C \cup^* D)$ and $X = \sigma(A \cup^* B) = \sigma((A \cup^* B) -^* (C \cup^* D))$.

we could work with infinite halfspaces. Less modest extensions might include various classes of sculptured surfaces, or even all irreducible algebraic surfaces.

We can assess the difficulties this enterprise raises by reviewing the basic CSG algorithms we have presented. Recall that the basic classification algorithms follow the divide-and-conquer paradigm. The attractiveness of such a strategy depends on the ease with which we can do the various classifications with respect to primitives, and the algorithmic complexity entailed by analyzing neighborhoods, sorting points on surface intersections, determining adjacencies, and so on. With greater geometric complexities at the primitive level, the difficulty of these operations quickly increases, and even the classification against primitives can no longer be taken for granted.

In such a situation, a case-by-case analysis may become too complex, and more general algorithms will be needed. Such algorithms are the subject of Chapters 5, 6, and 7. They continue to be research topics. In geometric and solid modeling, these algorithms are exercised many times. Each one of them must be sufficiently fast, yield results of adequate accuracy, and exhibit unfailing robustness. How best to negotiate these sometimes conflicting demands is not clear at this time, and probably depends not only on the geometric coverage, but also on the individual applications for which the modeler is needed.

2.3 Boundary Representations

We can represent a solid unambiguously by describing its surface and topologically orienting it such that we can tell, at each surface point, on which side the solid interior lies. This description has two parts, a *topological* description of the connectivity and orientation of vertices, edges, and faces, and a *geometric* description for embedding these surface elements in space. Historically, the representation evolved from a description of polyhedra.

Briefly, the topological description specifies vertices, edges, and faces abstractly, and indicates their incidences and adjacencies. The geometric representation specifies, for example, the equations of the surfaces of which the faces are a subset. The equations have been written such that, at a point p in the interior of a face f, the surface normal points to the exterior of the solid. More details are given later in Section 2.3.2, in this chapter, and in Section 3.2, in Chapter 3.

2.3.1 Manifold Versus Nonmanifold Representation

A large segment of the literature requires that the surface represented by a boundary representation be a closed, oriented manifold embedded in 3-space. Intuitively, a *manifold* surface has the property that, around every one of its points, there exists a neighborhood that is homeomorphic to the plane. That is, we can deform the surface locally into a plane without tearing it or identifying separate points with each other. Thus, surfaces that intersect or touch themselves are excluded.

A manifold surface is *orientable* if we can distinguish two different sides. The procedure for deciding orientability can be thought of as follows. Pick any point p, and define arbitrarily a clockwise orientation around it. Maintaining this orientation, move along any closed path on the surface. If there exists a path such that it is possible to return to p with an opposite orientation, then the surface is not orientable; otherwise, it is orientable. Examples of nonorientable surfaces include the Möbius strip and the Klein bottle. Orientable surfaces include the sphere and the torus. Closed, orientable manifolds partition the space into three regions that we may call the *interior*, the *surface*, and the *exterior*, respectively. In Section 2.4.2 we explain these concepts in greater detail.

The topological properties of manifolds are well understood. Thus, restricting attention to manifold solids has the advantage that one can draw on a rich mathematical theory for such objects. However, systematic work to relate this topological theory to specific representation schemata is relatively recent.

It is only recently that the requirement for manifold surface objects in B-rep is being revised, partly because a regularized Boolean operation on two manifold objects may yield a nonmanifold result. An example is shown in Figure 2.28, where we have taken the regularized union of two L-brackets. The problem is the edge (P, Q) that is adjacent to four faces. Three ap-

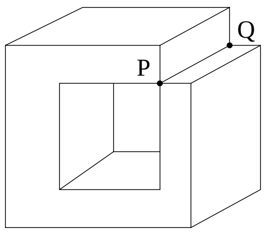

Figure 2.28 A Nonmanifold Object

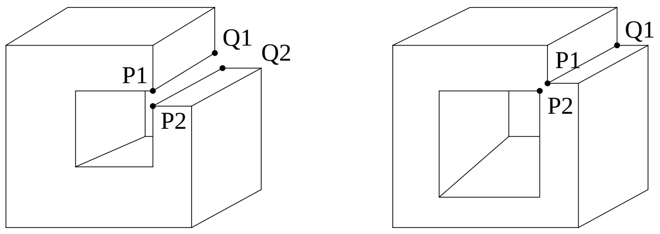

Figure 2.29 Two Possible Topologies

proaches to treating nonmanifold structures have been developed:

1. Objects must be manifolds, so operations on solids with nonmanifold results are not allowed and are considered an error.

2. Objects are topological manifolds, but their embedding in 3-space permits geometric coincidence of topologically separate structures.

3. Nonmanifold objects are permitted, both as input and as output.

Not much needs to be said about the first approach. It is straightforward and appears to be satisfactory in many applications. Note, however, that it unduly restricts modelers carrying out Boolean operations. Moreover, depending on the internals of the modeling system, operations that produce a nonmanifold object as an intermediate result might be disallowed even when the final result would be a manifold object. Such restrictions might not be convenient for the user.

In the second approach, we must give a topological interpretation of the nonmanifold structures. In the example of Figure 2.28, we must interpret the nonmanifold edge as two separate edges that happen to coincide. Two possibilities exist, and Figure 2.29 shows them side by side. Which interpretation should be chosen is discussed in Section 2.4. In this example, the left interpretation is more natural. Briefly, we choose an interpretation in which the surface is triangulable without degenerate triangles. Note that such a triangulation is possible for the left, but not for the right, interpretation shown in Figure 2.29: In the right interpretation, the triangulation of the front face must include an edge (P_1, P_2), since those two points are topologically distinct. They are, however, geometrically coincident; hence, this edge has zero length — that is, the adjacent triangles are degenerate.

From a robustness point of view, the second approach is likely to lead to difficult geometric problems, and analyzing them in the presence of geometrically coincident but topologically separate surface elements could be intricate. These difficulties would be further exacerbated in the curved-surface domain, in which the numerical problems are more severe. Moreover, no efficient general algorithm for triangulating curved faces is known.

In the third approach, nonmanifold edges and vertices are accepted. It is our experience that this approach ultimately leads to the simplest algorithms because it requires neither testing for the presence of the disallowed configurations, nor special processing that derives topological disambiguations. The algorithm for Boolean operations on polyhedra described in the next chapter is based on this approach.

2.3.2 Winged-Edge Representation

The oldest formalized schema for representing the boundary of a polyhedron and its topology appears to be the *winged-edge* representation. It describes manifold polyhedral objects by three tables, recording information about vertices, faces, and edges. We will describe a nonmanifold representation scheme in Chapter 3.

The topological information is as follows. Each face is bounded by a set of disjoint edge cycles, one of which is the outside boundary of the face, the others bounding holes. In the face table, therefore, a representative edge of each cycle is recorded. Each vertex is adjacent to a circularly ordered set of edges, so the vertex table specifies one of these edges for each vertex. Finally, for each edge, the following information is given:

1. Incident vertices

2. Left and right adjacent face

3. Preceding and succeeding edge in clockwise order (explained later)

4. Preceding and succeeding edge in counterclockwise order

The edge is oriented by giving the two incident vertices in order, the first being the *from* vertex, the second the *to* vertex. Left and right, as well as clockwise and counterclockwise, are interpreted with respect to viewing the oriented edge from the solid exterior. The information is shown schematically in Figure 2.30. Various restrictions may be placed on faces. For example, we may require that each face be bounded by a single cycle of edges, or even that each face be triangular. Such restrictions might be imposed to increase the uniformity of data structures, or to simplify processing for certain operations on B-rep objects.

The geometric information consists typically of coordinates of the vertices and plane equations for the faces. Each face equation has been written such that its normal, at an interior face point, is directed toward the outside of the solid. Thus, if two faces lie on the same plane $P = 0$, but in opposite orientation, then both $P = 0$ and $-P = 0$ must be specified.

The geometric information may also include parametric equations for specifying the edges in 3-space. In the case of curved solids, other information may be required to avoid ambiguities, as discussed in Chapter 5.

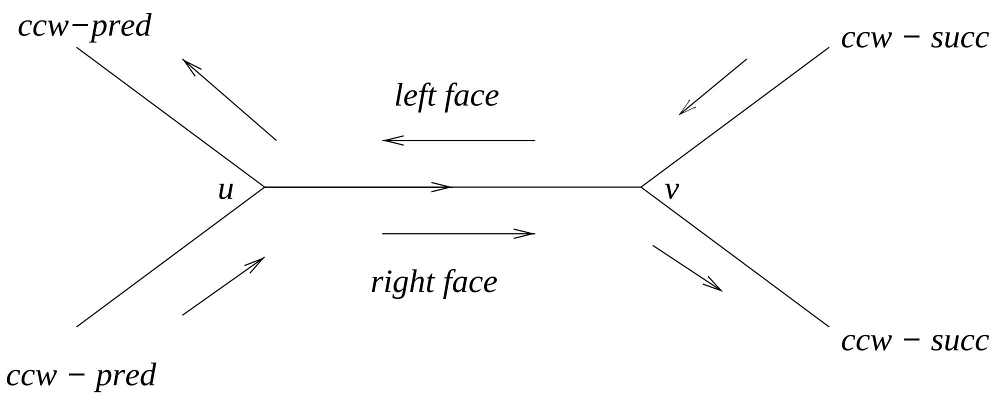

Figure 2.30 Winged-Edge Data Structure

Figure 2.31 shows a tetrahedron, where vertices, edges, and faces are labeled as shown. The topological data in its winged-edge representation is summarized in Table 2.2.

2.3.3 The Euler–Poincaré Formula

As we have seen, the topological data of a B-rep solid is symbolic information. Unless care is exercised, this prescribed topology might be inconsistent in the sense that there cannot exist a manifold solid whose vertices, edges, and faces satisfy the prescribed incidence relationships. This problem becomes especially acute when the topological data are derived from geometric information that is only approximate, due to floating-point errors. Hence, there is interest in maintaining consistent topological data, and a number of formulae have been found that must be obeyed by the number of vertices,

Edge	Vertices		Faces		Clockwise		Counter-clockwise	
Name	from	to	left	right	pred	succ	pred	succ
a	1	2	A	D	d	e	f	b
b	2	3	B	D	e	c	a	f
f	3	1	C	D	c	d	b	a
c	3	4	B	C	b	e	f	d
d	1	4	C	A	f	c	a	e
e	2	4	A	B	a	d	b	c

Table 2.2 Edge Table of the Tetrahedron, Winged-Edge Methodology

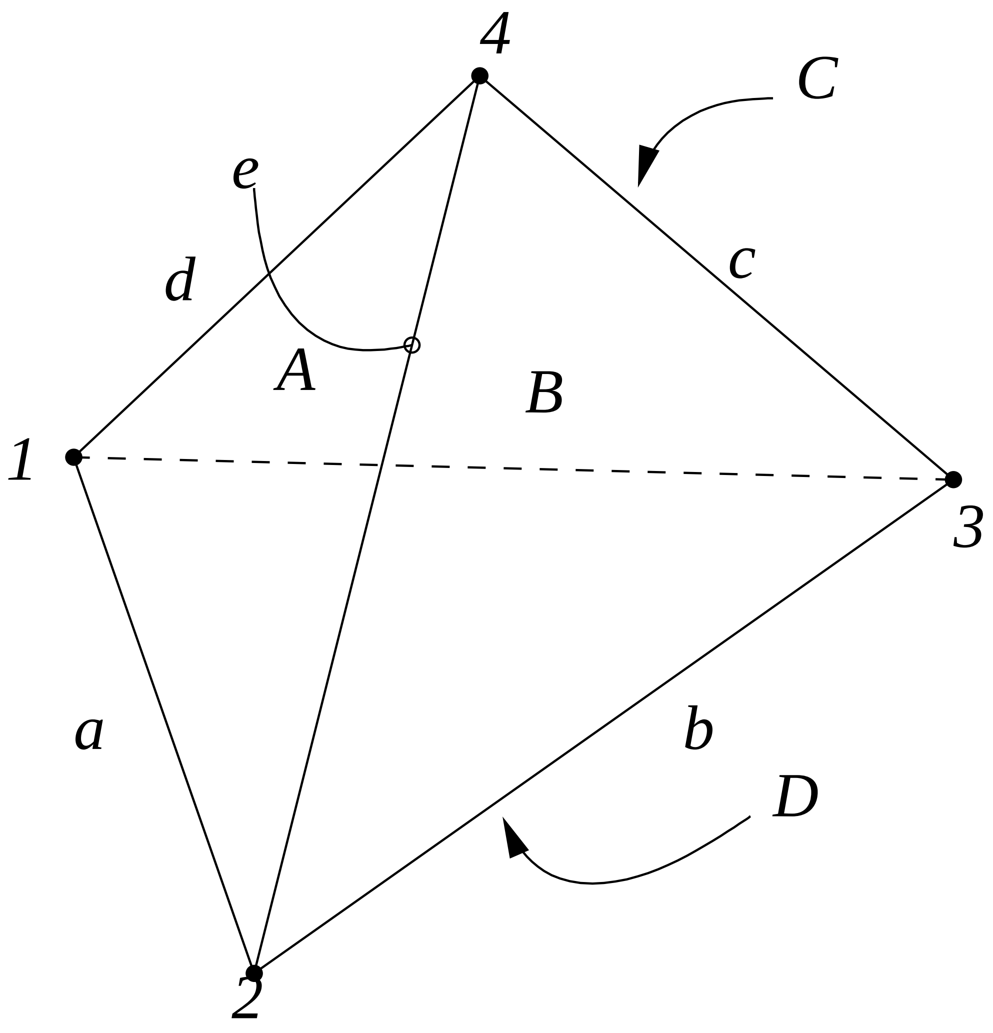

Figure 2.31 Tetrahedron

edges, and faces. Note that these formulae provide necessary conditions, but not sufficient ones. In Section 2.4, we will derive necessary and sufficient conditions.

From a topological viewpoint, the simplest solids are those that have a closed orientable surface and no holes or interior voids. We assume that each face is bounded by a single loop of adjacent vertices; that is, the face is homeomorphic to a closed disk. Then the number of vertices V, edges E, and faces F of the solid satisfy the *Euler* formula:

$$V - E + F - 2 = 0$$

This fact is easily proved by induction on the surface structure. Extensions to this formula have been made that account for faces not being homeomorphic to closed disks, the solid surface not being without holes, and the solid having interior voids, as reviewed next.

We consider the possibility that the solid has holes, but that it remains bounded by a single, connected surface. Moreover, each face is assumed to be homeomorphic to disk. For example, the torus has one hole, and the object in Figure 2.32 has two. It is a well-known fact that such solids are topologically equivalent, i.e., *homeomorphic*, to a sphere with zero or more handles. For example, the object of Figure 2.32 is homeomorphic to a sphere with two handles, the latter shown in Figure 2.33. The number of handles is called the *genus* of the surface. In general, with a genus G, the numbers of vertices, edges, and faces obey the *Euler–Poincaré* formula:

$$V - E + F - 2(1 - G) = 0$$

Next, we further generalize by adding the possibility of internal voids. These voids are bounded by separate closed manifold surfaces, called *shells*. The number of shells will be denoted by S. Finally, we relax the requirement that a face is bounded by a single loop of vertices, but require that each face can be mapped to the plane. Thus, a sphere missing at least one point can be a face. In Figure 2.34, a face is shown with four bounding loops. Note that one of these loops consists of a single vertex, and another one of two vertices connected by an edge. To account for faces of this complexity, we must count, for each face, the number of bounding vertex loops. For the face in Figure 2.34, this number is four. With L the total number of loops, the relationship among the number of faces, edges, vertices, loops, and shells, and the sum G of each shell's genus, is then

$$V - E + F - (L - F) - 2(S - G) = 0$$

An example solid illustrating this relationship is shown in Figure 2.35.

We may think of the quantities V, E, F, L, S, and G as existing in an abstract six-dimensional space. The relationship among them is then the equation of a hyperplane. Since the values of the variables must be non-negative integers, we might view the relation as defining a lattice on this hyperplane. For each solid with a given topological structure, there corresponds a point in this lattice.

Although a manifold solid must satisfy the extended Euler–Poincaré formula, not every surface satisfying the formula will be the surface of a manifold solid. For example, the cube is a manifold object with 6 faces, 12 edges, and 8 vertices. It has a single shell surface of genus zero. However, the surface shown in Figure 2.36 has the same number of faces, edges, and vertices, yet it is not the surface of any manifold solid, since it has a "dangling" quadrilateral face attached to the prismatic part by a nonmanifold edge.

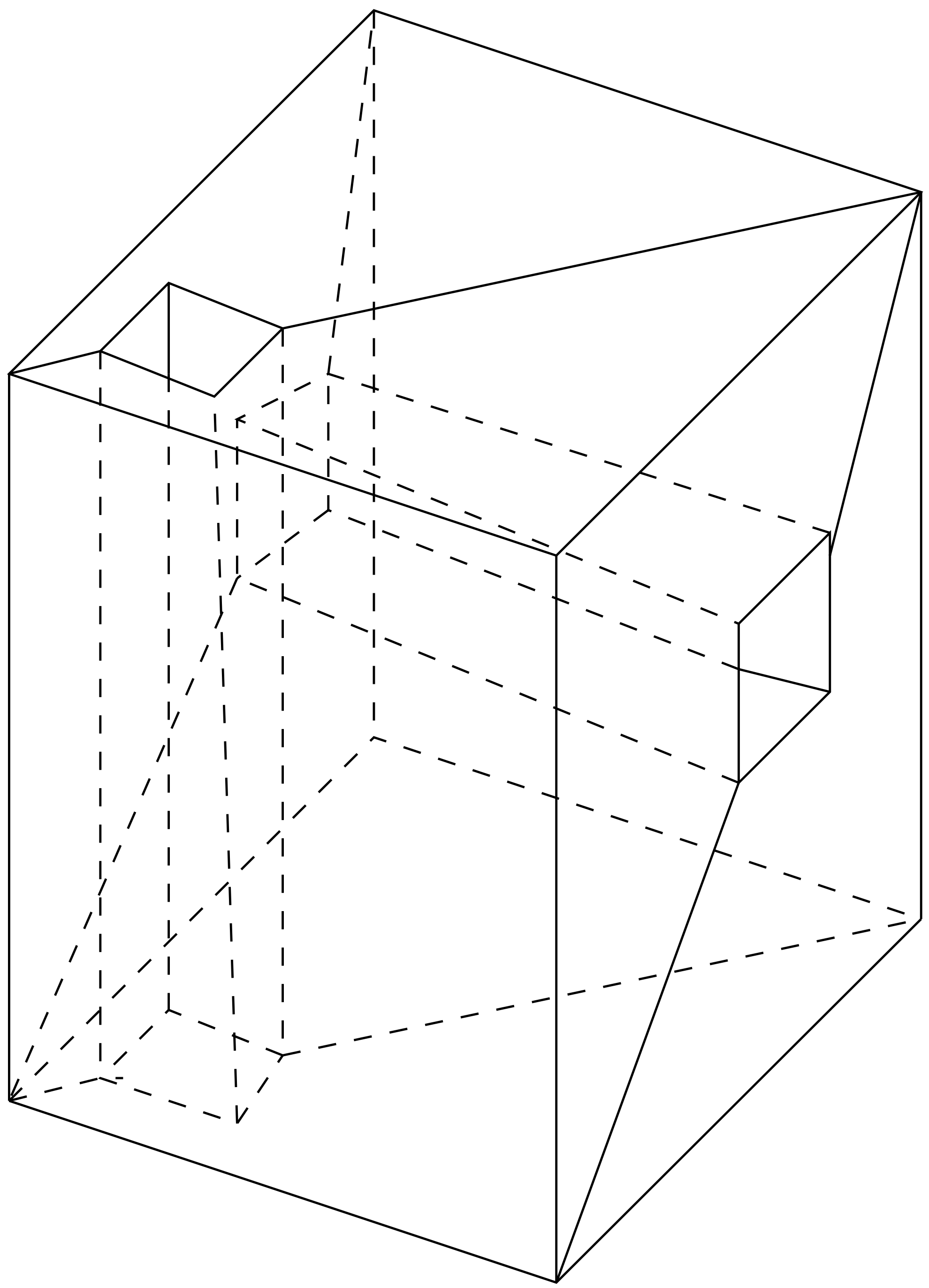

Figure 2.32 An Object with Two Holes and with Faces Homeomorphic to Disks

2.3.4 Euler Operators

Conceptually, *Euler operators* can be thought of as creating and modifying consistently the topology of manifold object surfaces. In particular, they can create closed surfaces, and modify these surfaces by adding or deleting faces, edges, and vertices. They also modify the surface genus by adding or deleting

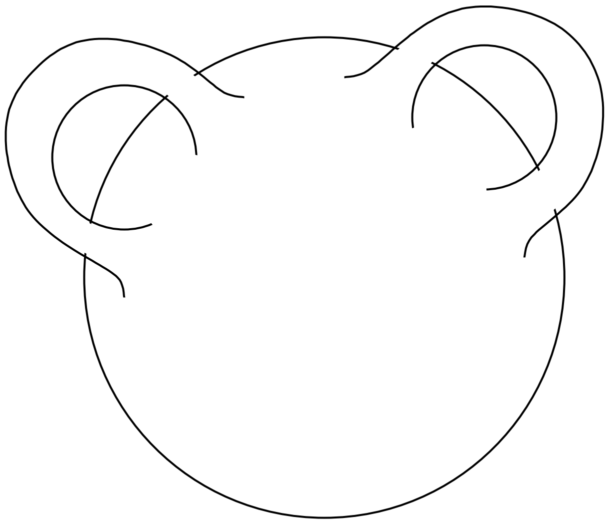

Figure 2.33 A Surface of Genus 2

handles. Euler operations are traditionally named by a string of the form *mxky*, where *m* stands for *make*, and *k* stands for *kill*. The strings *x* and *y* name the topological element types that are created or destroyed. The types are vertex, edge, loop, face, and shell. Ordinarily, only one new element of each type is created or destroyed, but sometimes several elements of the same type are created or destroyed. For example, *mekfl* adds an edge and deletes a face and a loop, whereas *mefl* adds an edge, a face, and a loop.

Euler operators are used as an intermediate language in some modeling systems. Using them has the advantage of insulating, to a degree, the operations implemented on top of them from details of the data structures used to represent the surface topology. Thus, in principle, the underlying representation could be changed with minimal impact on the modeling system's implementation. Another advantage is that Euler operators ensure topologi-

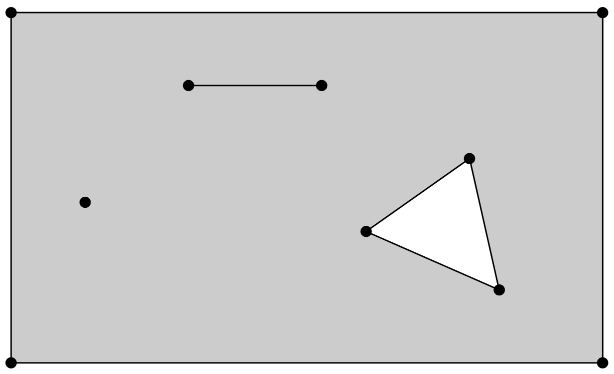

Figure 2.34 A Face with Four Bounding Loops

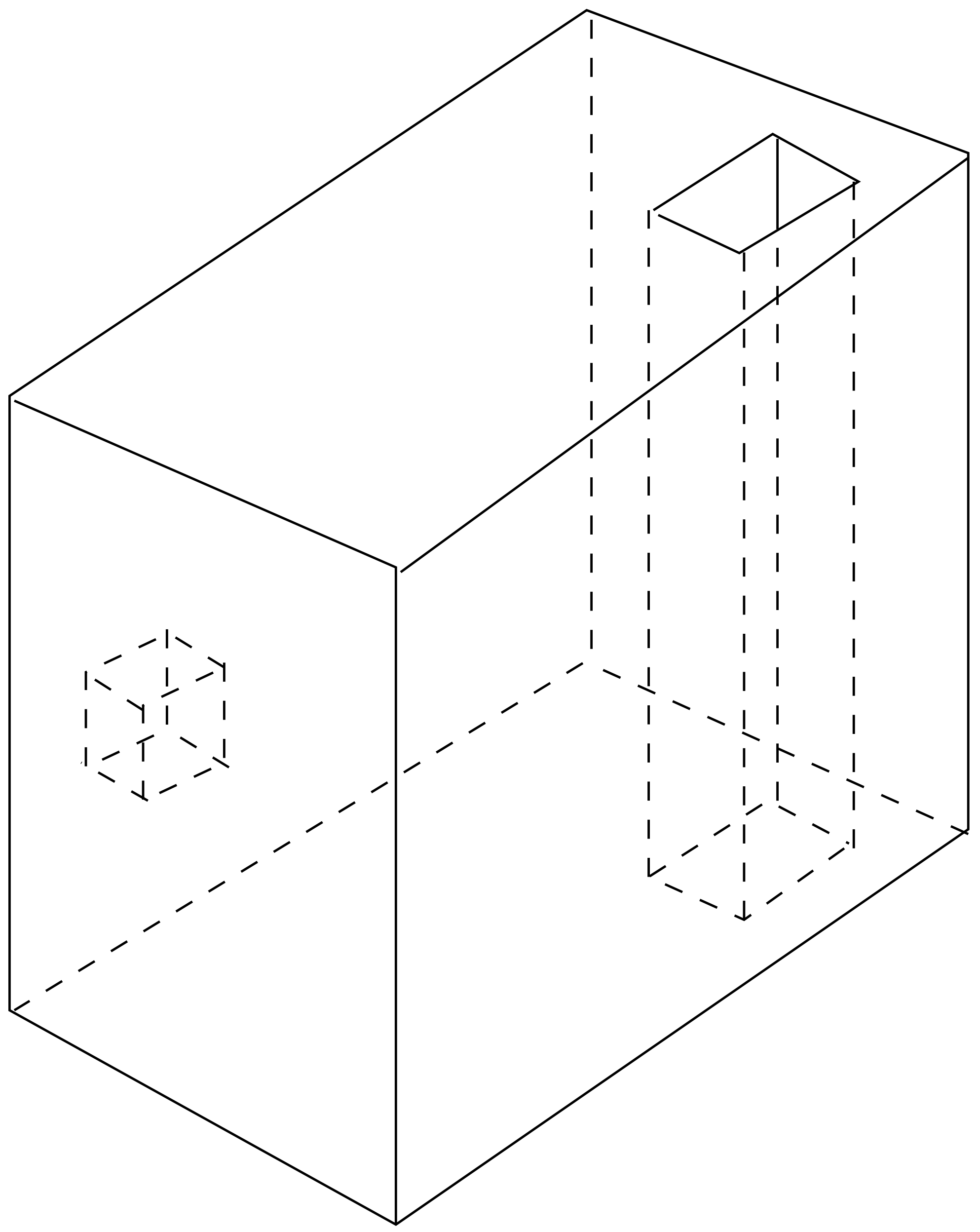

Figure 2.35 Solid with 24 Vertices, 36 Edges, 16 Faces, 18 Loops, 2 Shells, and Genus Sum 1

cal consistency throughout the modeling process. This can be advantageous when the precise topology of the result of a modeling operation may be in doubt because of imprecision of the numerical model data, as discussed in Chapter 4.

As example of specific Euler operators, consider the operation of adding an edge between two existing vertices. Depending on the two vertices designated, this operation has differing topological effects. In consequence, different Euler operations would be used to implement the operation. The

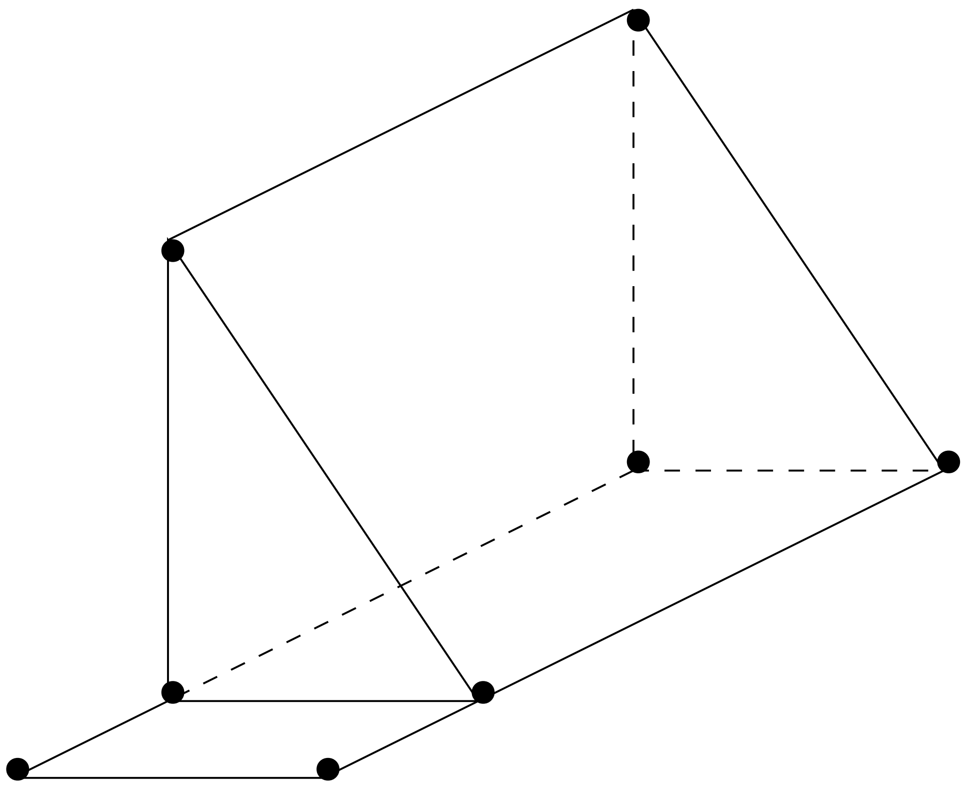

Figure 2.36 Surface with 8 Vertices, 12 Edges, and 6 Faces

possibilities are as follows.

1. The new edge closes off one part of a face from the rest. In this case, the operation is called *mefl*. Its effect is to increase the number of edges, faces, and loops by one each. An example is shown in Figure 2.37.

2. The new edge connects two different loops bounding the same face. In this case, the operation is called *mekl*. Here we have added one edge and deleted one loop; see also Figure 2.38.

3. The new edge connects two vertices on two different shells. In this case, the operation is called *meks*. It merges the two shells, which includes deleting a face on each shell and creating a new face that makes a connection between the two surfaces. Figure 2.39 shows an example of the *meks* operation. The interior shell is connected with the exterior shell, opening the interior void to the outside by a conical face. Note that for polyhedra, more than one edge would have to be created.

Note that these operations need additional specifications to ensure an unambiguous placement of the new constructs. For example, in the *meks* operation, it is not clear which faces should be deleted on each shell. Suitable

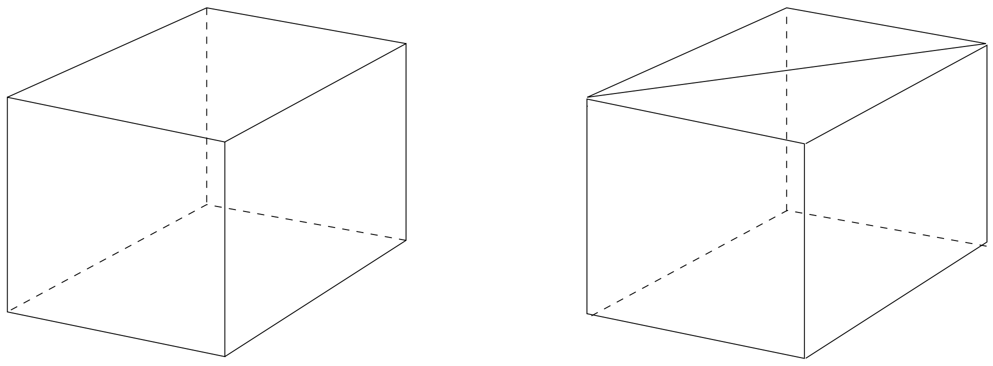

Figure 2.37 *mefl* Operation

conventions for communicating this geometric information to the operation are readily worked out, say via certain parameters.

The implementation of Euler operations increases in difficulty with the geometric coverage of the modeling system. We mentioned that there are no efficient general algorithms that triangulate curved faces. Clearly, triangulation of curved faces can be based on the *mefl* operation. Hence, this operation will be difficult to implement unless the geometric coverage is suitably restricted.

2.4 Topological Validity of B-rep Solids

A basic assumption underlying solid modeling is that we deal with topologically valid solid objects. The meaning of *topological validity* needs to be made precise, for otherwise we cannot be assured that the computer representations of solids and the algorithms using them are correct. This task is especially important in B-rep, where we must infer, from a description of the two-dimensional boundary, that a solid is defined. In this section, we give a definition of topological validity. Based on this definition, it is possible to derive an algorithm that tests whether a given data structure, intended as a boundary representation of a solid, does in fact describe a solid. Such an

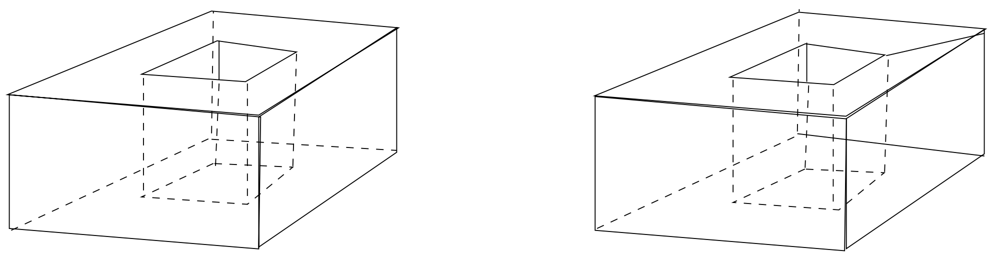

Figure 2.38 *mekl* Operation

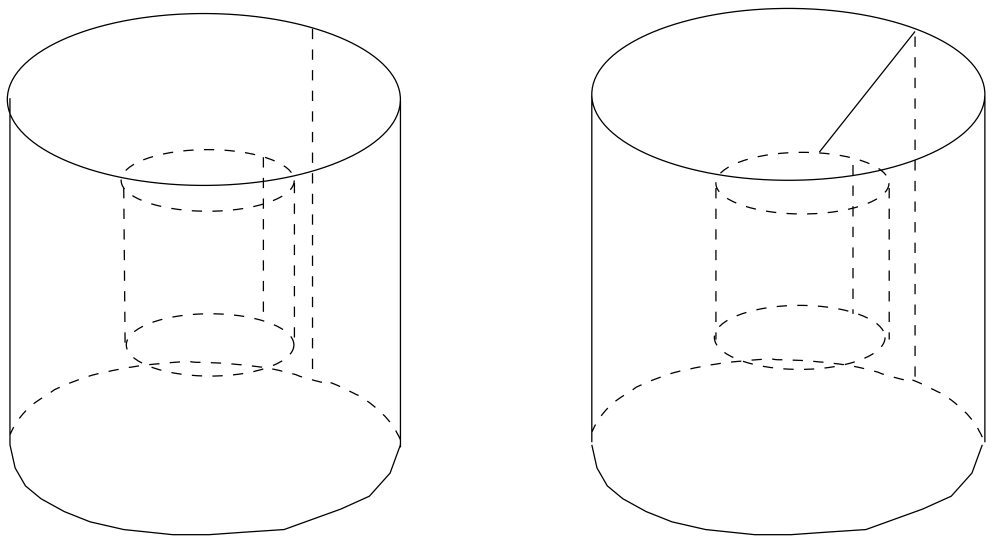

Figure 2.39 *meks* Operation

algorithm is also sketched, but no deep consideration has been given to its implementation. Rather, it serves to elucidate the various aspects of topological validity.

Initially, we consider manifold solids. Thereafter, we discuss how to characterize nonmanifold solids topologically. The material is fairly detailed, as is necessary: The tools provided by topology are very general, and their naive use can lead to subtle errors. Hence, it is important to develop the material carefully and explicitly.

Checking topological validity has a geometric dimension. For instance, each face in a B-rep must consist of manifold points. In the case of planar faces, this condition is trivial; for curved faces, however, it is by no means a straightforward computation. Moreover, the geometric dimension naturally suggests broadening the topological-validity problem to a proper mathematical definition of the term *solid* that encompasses the other aspects as well. Although we do not develop such a comprehensive definition, we discuss some of the issues that are needed for for such a task. These issues arise from the interaction of geometric and topological factors.

2.4.1 Topological Polyhedra

Our objective is to characterize a solid as a topological polyhedron. We initially think of a solid as a 3-manifold with boundary, and then impose a triangulation to grasp better the structure of these manifolds. The resulting definition of a topological solid is preliminary because we characterize the solid as an object without accounting for the surrounding space. By characterizing subsequently the relationship between the solid and Euclidian 3-space, we refine this definition to a definition of manifold solids in the sense discussed in Section 2.3.1.

Topological Spaces

A *topological space* (X, T) is a set X along with a system of subsets T, called the *open sets* of X. The system T must satisfy the following two properties:

1. The intersection of finitely many sets in T is again in T.

2. The union of sets in T is also in T.

Note that infinite unions of open sets are permitted. Therefore, T must be closed under finite intersection and arbitrary union. A subset of X is *closed* if its complement is open.

In the following discussion, we specialize the set X and assume that it is the n-dimensional Euclidian space $\mathbf{E}^n$ or a subset thereof. $\mathbf{E}^n$ consists of all points $(x_1, ..., x_n)$, where the coordinates x_k are real numbers. In $\mathbf{E}^n$, we consider the *natural* topology, using as our system of open sets all those sets that can be obtained as the union of *open balls.* The open ball $B(p, r)$, of radius $r > 0$ centered at the point $p = (x_1, ..., x_n)$, is defined as

$$B(p,r) = \{q = (y_1, ..., y_n) \mid d(p,q) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2} < r\}$$

That is, $B(p, r)$ consists of all points whose Euclidian distance from p is less than r.

A *neighborhood* of a point p is any open set U that contains p. Note that this definition of neighborhood has a different meaning from that introduced in Section 2.2.4. It can be shown that a subset of $\mathbf{E}^n$ is open precisely when X contains a neighborhood of every point p in X.

The *interior* of a set U, denoted $int(U)$, consists of all points $p \in U$ such that U contains a neighborhood of p. The *closure* of a set U, denoted $cl(U)$, is the complement of the interior of the complement of U. Let $\neg U$ denote the set-theoretic complement of U. Then

$$cl(U) = \neg int(\neg U)$$

A map f from a topological space (X, T) to another topological space (X', T') is *continuous* if every neighborhood of $f(p)$ in (X', T') is mapped by f^{-1} to a neighborhood of p in (X, T). If f is *bijective* (i.e., is one to one and onto), and if both f and its inverse f^{-1} are continuous, then f is a *homeomorphism.* Two topological spaces are *topologically equivalent* if there is a homeomorphism between them.

In $\mathbf{E}^n$, we identify topological subspaces. A subspace is a subset Y of $\mathbf{E}^n$ along with the *relative topology* consisting of the intersection of the open sets of $\mathbf{E}^n$ with Y. Examples include open balls of any radius, but also closed sets such as the unit cube consisting of all points $p = (x_1, ..., x_n)$ such that, for $k = 1, ..., n$, we have $0 \leq x_k \leq 1$. Note that an open set in the relative topology need not be open in the containing topological space. For example, the (relatively) open set U in the unit cube in $\mathbf{E}^3$, obtained as the intersection with the open ball of radius 1 centered at $(1, 1, 1)$, is not an open set in $\mathbf{E}^3$ since no neighborhood of $(1, 1, 1)$, in $\mathbf{E}^3$, is contained in U.

An *n-manifold* M in $\mathbf{E}^m$, where $m \geq n$, is a subspace that is *locally* homeomorphic to $\mathbf{E}^n$. That is, for every point p of M, there exists a neighborhood U of p that is homeomorphic to $\mathbf{E}^n$. An *n-manifold with boundary* is a subspace whose boundary point neighborhoods are locally homeomorphic to the positive halfspace

$$\mathbf{E}^{n+} = \{(x_1, ..., x_n) \in \mathbf{E}^n \mid x_1 \geq 0\}$$

and whose interior point neighborhoods are locally homeomorphic to $\mathbf{E}^n$. The hyperplane $x_1 = 0$ is the *boundary* of $\mathbf{E}^{n+}$.

Note that, in an n-manifold M with boundary, we can distinguish between interior and boundary points: A point $p \in M$ is an *interior point* if there is a neighborhood U of p that is homeomorphic to $\mathbf{E}^n$. A point $p \in M$ is a *boundary point* if it has a neighborhood U that is homeomorphic to a neighborhood of the point $(0, ..., 0)$ in $\mathbf{E}^{n+}$. In contrast, an n-manifold consists of only interior points.

The boundary of an n-manifold with boundary can be shown to be homeomorphic to an $(n-1)$-manifold without boundary. Intuitively, a manifold is *connected* if it cannot be decomposed into two disjoint manifolds.

A set in $\mathbf{E}^n$ is *bounded* if it is contained in an open ball. When a set is both closed and bounded, it is *compact*.

We wish to define a solid as a connected 3-manifold with boundary where, in addition, the boundary is compact. This definition is too restrictive in that it excludes nonmanifold solids. At the same time, it is also too general, because it places no requirements on the space surrounding the solid.

Simplicial Complexes

We explain how to construct 3-manifolds combinatorially. The basic building blocks are simplices of various dimensions that are put together in particular ways to obtain manifolds. We explain how this is done.

Let p_0 and p_1 be two distinct points. The *convex combination* spanned by p_0 and p_1 is the set

$$\langle p_0, p_1 \rangle = \{\lambda p_0 + (1 - \lambda) p_1 \mid 0 \leq \lambda \leq 1\}$$

Geometrically, $\langle p_0, p_1\rangle$ is the closed line segment $[p_0, p_1]$ in Euclidian space. Similarly, we define the convex combination spanned by three distinct points as

$$\begin{aligned}\langle p_0, p_1, p_2\rangle &= \{(\lambda p_0 + (1-\lambda)p_1)\mu + (1-\mu)p_2 \mid 0 \leq \lambda, \mu \leq 1\} \\ &= \{\mu q + (1-\mu)p_2 \mid q \in \langle p_0, p_1\rangle, 0 \leq \mu \leq 1\}\end{aligned}$$

If the p_i are not collinear, then $\langle p_0, p_1, p_2\rangle$ is a triangle with vertices p_0, p_1, and p_2. The notion of convex combination generalizes to arbitrary dimension: The convex combination of the points $p_0, ..., p_{d+1}$ is

$$\langle p_0, ..., p_{d+1}\rangle = \{\lambda q + (1-\lambda)p_{d+1} \mid q \in \langle p_0, ..., p_d\rangle, 0 \leq \lambda \leq 1\}$$

We say that $d+1$ points in d-dimensional real space $\mathbf{R}^d$ are *linearly independent* if none of the points is contained in the convex combination of the others. It is not difficult to show that we can define $\langle p_0, ..., p_d\rangle$ equivalently as

$$\langle p_0, ..., p_d\rangle = \{\sum_{k=0}^{d} \lambda_k p_k \mid \lambda_k \geq 0 \text{ and } \sum_{k=0}^{d} \lambda_k = 1\}$$

Here, the numbers λ_k are the *barycentric coordinates* of the point $\sum_{k=0}^{d} \lambda_k p_k$. If the p_k are linearly independent, then it can be shown that the barycentric coordinates of every point in $\langle p_0, ..., p_d\rangle$ are unique.

A *d-simplex* is the convex combination of $d+1$ linearly independent points. Moreover, d is the *dimension* of the d-simplex. Clearly, a 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle, and a 3-simplex is a tetrahedron.

The *boundary* of a d-simplex S consists of all $(d-k)$-simplices contained in S, where $k > 0$, and is denoted ∂S. Every simplex in the boundary of S is a *face* of S. A k-simplex that is a face is also called a *k-face*. The following theorem is elementary.

Theorem

A d-simplex contains exactly $\binom{d+1}{k+1}$ k-simplices as faces.

Moreover, two d-simplices are homeomorphic.

A *simplicial complex* C is a finite set of simplices satisfying the following restrictions:

1. Let S be a simplex in C, and let S' be one of its faces. Then S' is also in C.

2. Let S_1 and S_2 be two simplices of C. Then their intersection is either empty or is a simplex of C.

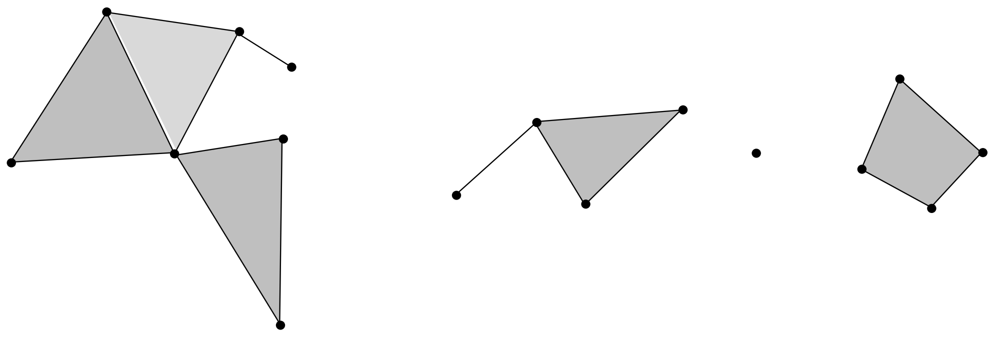

Figure 2.40 A Simplicial Complex C

Figure 2.40 shows a simplicial complex; Figure 2.41 shows a set of simplices that do not form a simplicial complex.

The *dimension* of the simplicial complex C is defined as the maximum dimension of the simplices in C. The dimension of the simplicial complex in Figure 2.40 is 2. It can be proved that the dimension of a simplicial complex is invariant under continuous maps. If S is a d-simplex and $d > 0$, then the boundary ∂S of S is a simplicial complex of dimension $d - 1$.

A subset of $\mathbf{E}^n$ is *triangulable* if it is homeomorphic to a simplicial complex. A triangulable set is also called a *topological polyhedron.* Note that the term is not used in a geometric sense, because a homeomorphism may map a linear surface to a curved surface. Hence, the closed unit ball in $\mathbf{E}^3$ is a topological polyhedron since it is homeomorphic to a 3-simplex. Figure 2.42 shows several topological polyhedra of dimension 2. Moreover, given a topological polyhedron M, the homeomorphism from a simplicial complex onto M is a *triangulation* of M.

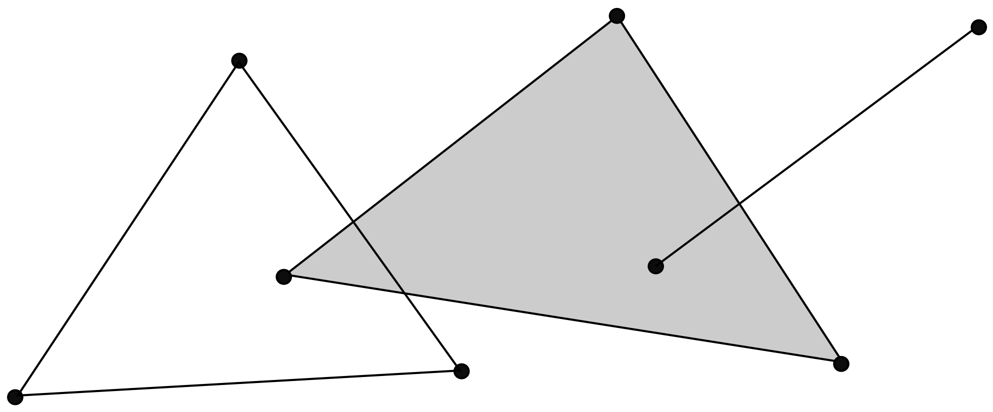

Figure 2.41 Simplices Not Forming a Simplicial Complex

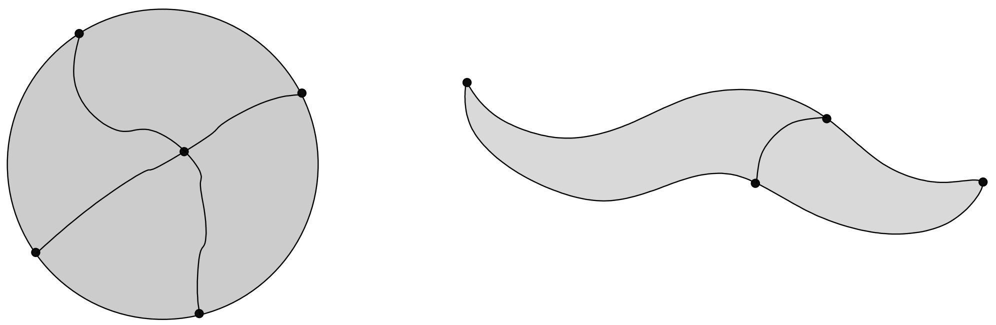

Figure 2.42 Topological Polyhedra of Dimension 2

Abstract Simplicial Complexes and Geometric Realization

Since we defined simplices as convex combinations of points, it is conceivable that this definition is too narrow. That is, when constructing a simplicial complex, can we obtain more complicated structures using simplices that are only homeomorphic to convex combinations? From a topological point of view, the answer is no, and is justified as follows.

We define an *abstract* simplex S as a finite set of points, called the *vertices* of S. Every proper subset of S is a face of S. If S consists of $d+1$ points, then we say that it has the dimension d. An *abstract* simplicial complex C is defined as follows:

1. There is a finite set of *vertices* V.
2. C is a set of subsets S of V with the property that all subsets of S are in C.

Intuitively, the subsets S are the simplices in C.

It can be proved that every abstract simplicial complex C has a geometric realization $|C|$ in Euclidian space as a complex of simplices that are convex combinations. That is, given an abstract complex C with vertices $\{v_1, ..., v_m\}$, we can find m points in Euclidian n-dimensional space E such that, for every abstract simplex $S = \langle p_0, ..., p_d \rangle$ in C, the points in E corresponding to the p_k are linearly independent and hence define a simplex $|S|$ in E that is a convex combination of those points.

Theorem
If C is an abstract simplicial complex of dimension n, then C can be realized by a corresponding concrete simplicial complex $|C|$ in $\mathbf{E}^{2n+1}$, where the vertices are points and the simplices are convex combinations of them.

In other words, the abstract complex C has a "nice" piecewise linear realization in a Euclidian space of sufficiently high dimension. Thus, we do not lose generality by using the concrete definition of simplices as convex combinations.

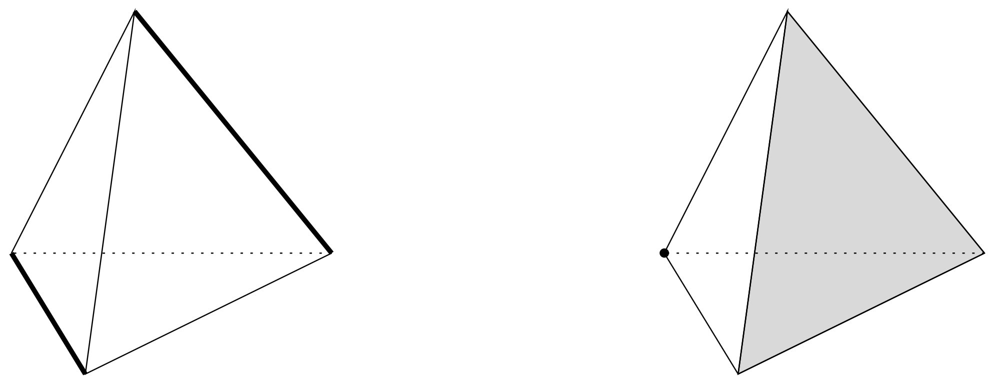

Figure 2.43 Opposite Faces in a 3-Simplex

Manifold Triangulations

We return to the problem of characterizing manifolds as topological polyhedra, and describe the local structure of a simplicial complex triangulating the manifold. Because of the above, we may assume that the simplicial complexes are piecewise linear in a suitable Euclidian space.

Let S be a d-simplex with vertices $p_0, ..., p_d$. A proper subset $q_0, ..., q_r$ of the vertices of S defines an r-simplex that is a face S_1 of S. Let $q_{r+1}, ..., q_d$ be the remaining vertices of S. Then these vertices define another face S_2. We say that S_1 and S_2 are *opposite faces* of S. Figure 2.43 shows examples in the case of $d = 3$. Let S and S' be two simplices in a simplicial complex. Then S and S' are *adjacent* if they have a common face. If S'' is a face in which S and S' are adjacent, then S and S' are *incident* to S''. Finally, a simplicial complex C is *connected*, if for all pairs of simplices S and S' in C, we can find a sequence of simplices $S_1, ..., S_r$ in C such that, for $1 \leq k < r$, we have $S = S_1$ and $S' = S_r$; and S_k is incident to S_{k+1}, or vice versa.

Let S be a simplex in some simplicial complex. S will be incident to a finite set of simplices $S_1, ..., S_r$ in C. For each simplex S_i of which S is a face, let T_i be the face of S_i opposite S. The set of all such opposite faces is the *link* of S in C; see Figure 2.44 for an example. We are now in a position to characterize 2- and 3-manifolds in terms of simplicial complexes. Although stated as definitions, these characterizations can be proved formally.

A *2-manifold without boundary* is homeomorphic to a simplicial complex C of dimension 2 satisfying the following restrictions:

1. Every 1-simplex in C is incident to exactly two 2-simplices.

2. The link of every 0-simplex in C is a triangulation of the circle.

See also Figure 2.45 for an illustration of the vertex structure in a 2-manifold without boundary.

Similarly, a *3-manifold without boundary* is homeomorphic to a simplicial complex C of dimension 3 satisfying the following restrictions:

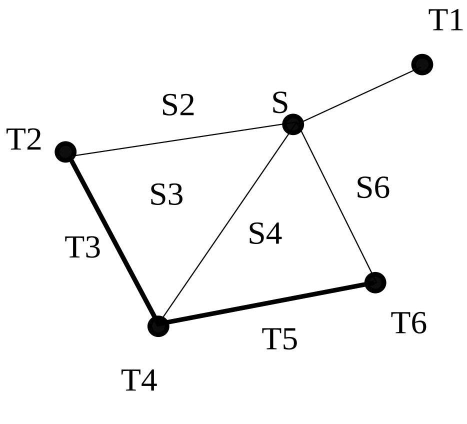

Figure 2.44 Link of S

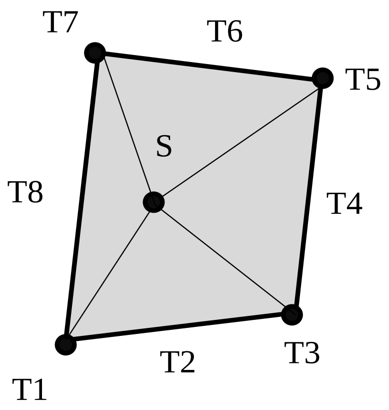

Figure 2.45 Vertex Structure in Triangulated 2-Manifold

1. Every 2-simplex in C is incident to exactly two 3-simplices.

2. The link of every 0-simplex in C is a triangulation of the sphere.

To so characterize 3-manifolds with boundary, we need to distinguish between simplices that are on the boundary of the manifold and ones that are interior. We discuss only 3-manifolds with boundary.

Let S be a 2-simplex in the complex C. Then S is an interior face if it is incident to exactly two 3-simplices of C, and is a boundary face if it is incident to exactly one 3-simplex. Similarly, a 0-simplex is interior if its link is a triangulation of the sphere, and is a boundary point if its link is a triangulation of the disk. Note that we do not give an analogous characterization of 1-simplices; such a characterization is not needed.

Formally, then, a *3-manifold with boundary* is homeomorphic to a simplicial complex C of dimension 3 satisfying the following restrictions:

1. Every 2-simplex is adjacent to one or two 3-simplices.

2. The link of every 0-simplex is a triangulation of the disk or the sphere.

This completes the explanation of the polyhedral structure of 2- and 3-manifolds.

2.4.2 Manifold Solids

We wish to define a topological solid as a 3-manifold with boundary, where, in addition, the boundary should be compact. The intuition is that we should have a finite surface, but that infinite volumes are permitted. As pointed out before, this definition will not constrain the relationship between the topological solid and the surrounding 3-space. Moreover, since in B-rep a solid is implicitly described by a specification of its surface, we must also

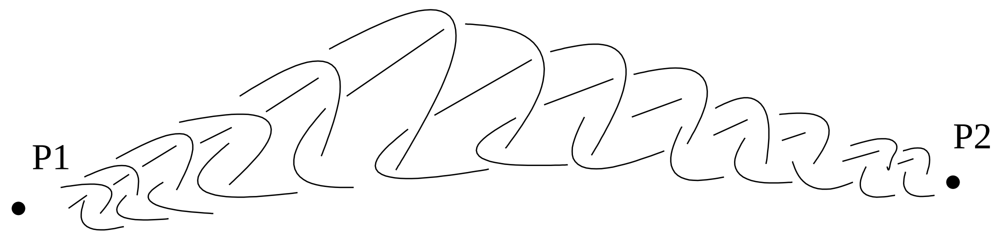

Figure 2.46 A Wildly Embedded Simple Arc

clarify the relationship between a topological solid and its surface. We do this now.

Embeddings in $\mathbf{E}^3$

We consider the relationship between a 3-manifold with boundary and the surrounding space. This relationship needs to be examined because there are "unreasonable" 3-manifolds in $\mathbf{E}^3$ that are homeomorphic to well-behaved topological polyhedra. For example, a manifold can be unreasonable because it has a fractal-like surface. The difficulty is rooted in the fact that the characterization of manifolds as topological polyhedra requires only that there be a homeomorphism from a simplicial complex of suitable structure onto the manifold, without consideration of whether this homeomorphism can be extended to the surrounding Euclidian space.

An *embedding* of the topological space (X, T) into a topological space (Y, T') is a homeomorphism between (X, T) and a subspace Y' of Y. In particular, let X and Y' be subspaces of $\mathbf{E}^n$. Depending on whether and how the homeomorphism extends to a homeomorphism of the entire surrounding space we call embeddings *tame* or *wild*. Wild embeddings lead to unreasonable 3-manifolds in $\mathbf{E}^3$. A simple example is depicted in Figure 2.46, which shows a wildly embedded arc in $\mathbf{E}^3$. By giving "thickness" to the arc, we obtain a 3-manifold with compact boundary that can be shown to be homeomorphic to a sphere. Clearly, this would not be a reasonable solid.

In view of this fact, we require, in addition, that a topological solid be tamely embedded into $\mathbf{E}^3$. Rather than giving a formal definition, we will argue later on that the primitives used in CSG and in B-rep guarantee that we are working with tame embeddings.

Orientability

So far, we have characterized a topological solid in its entirety; that is, as a three-dimensional object. In B-rep, however, we have only a description of the boundary. We need to make a connection between the topological structure of the solid and the topological structure of its boundary. This connection requires the concept of *orientability*. Briefly, we can prove the following theorem.

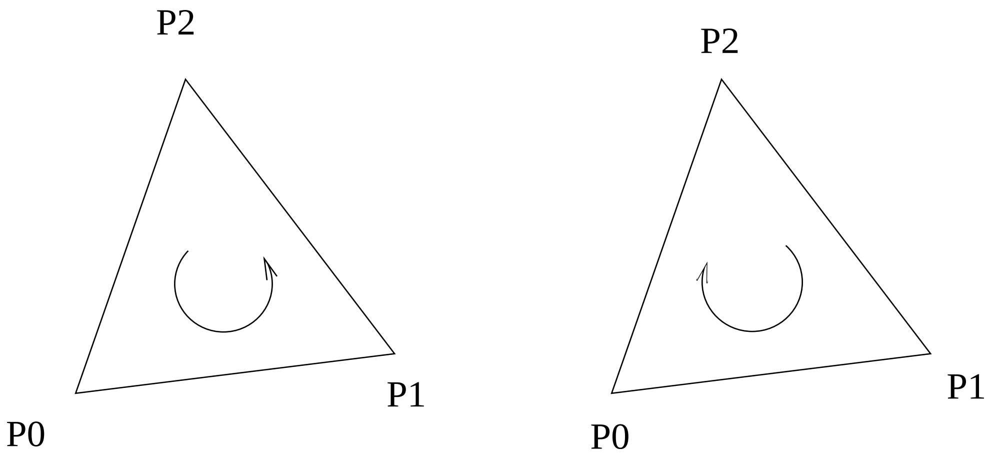

Figure 2.47 Orientations (p_0, p_1, p_2) and (p_0, p_2, p_1) of a 2-Simplex

Theorem
Let M be a connected 3-manifold with boundary embedded in $\mathbf{E}^3$. Then the boundary of M is an embedded orientable 2-manifold without boundary.

Note that we do not need to assume a tame embedding. A converse of this theorem can be formulated which allows us to conclude from boundary properties that a topological polyhedron is enclosed. However, such a theorem requires additional assumptions. These additional assumptions will involve geometric properties of the embedding and are explained later.

Orientability is best visualized in terms of a triangulation of the manifold. It can be shown that the orientability of a manifold is independent of the particular triangulation. That is, if the manifold is orientable, then every one of its triangulations is orientable. Conversely, if the manifold has an orientable triangulation, then the manifold is orientable. So, we explain orientability of 2-manifolds by orienting simplicial complexes that are a triangulation of the manifold.

We orient a 2-simplex by cyclically ordering its vertices. For example, the simplex $\langle p_0, p_1, p_2 \rangle$ can be oriented (p_0, p_1, p_2). Figure 2.47 shows the two possible orientations of the 2-simplex. Note that the orientation of a 2-simplex induces an orientation of every one of its 1-faces. More generally, any d-simplex can be oriented in exactly one of two ways.

Let S and S' be two adjacent 2-simplices in a complex C, and assume that they are adjacent in a 1-face S''. Then S and S' are *coherently oriented* if the orientations of S'' induced by the orientations of S and S' are opposite. Figure 2.48 shows two pairs of adjacent 2-simplices. The left pair is oriented coherently, whereas the right pair is not.

Let C be a triangulation of a connected 2-manifold with or without boundary. Then C is orientable iff all of its adjacent 2-simplices can be oriented coherently. It can be shown that, if C is orientable, then it can be oriented in

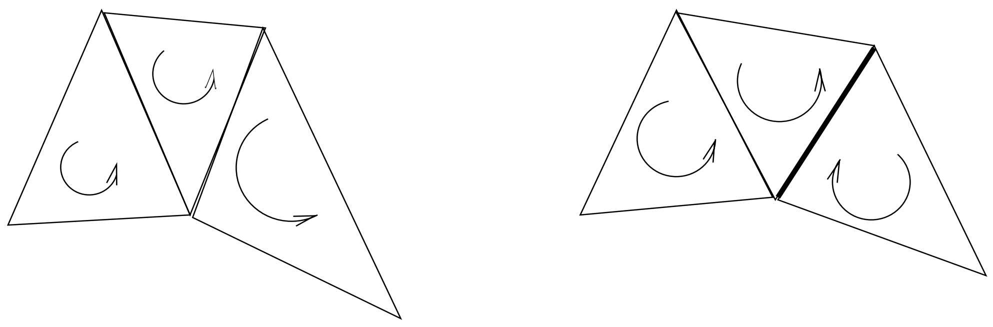

Figure 2.48 Coherent and Incoherent Orientations

exactly two ways, and for this reason there is a simple algorithm for testing orientability. Roughly speaking, we pick any 2-simplex and orient it arbitrarily. Thereafter, we orient an adjacent 2-simplex coherently, continuing iteratively until the manifold has been coherently oriented, or until we reach a 2-simplex S adjacent to two other simplices that are already oriented in a way that precludes orienting S coherently with both.

We return to the problem of formulating the converse of the preceding theorem. We would like to obtain a characterization that is roughly as follows.

Theorem
Let M' be a compact, connected 2-manifold without boundary that is tamely embedded into $\mathbf{E}^3$ and is oriented. If M' satisfies a property $\mathcal{P}$, then M' is the boundary of a connected 3-manifold that is tamely embedded in $\mathbf{E}^3$.

Sufficient properties $\mathcal{P}$ are readily formulated. One possibility is to require that $\mathbf{E}^3$ is triangulated in its entirety such that this triangulation also triangulates M'. However, rather than making $\mathcal{P}$ a global property, we formulate a more local property of the embedding.

Intuitively, wild embeddings (and some tame embeddings) are unsatisfactory because the manifold is “ruffled.” It appears that there is no purely topological characterization of this intuitive notion, and we need to introduce geometry to make the idea precise. Let M be an embedded 3-manifold with compact boundary in $\mathbf{E}^3$. We say that M is of *bounded variation* if every line in $\mathbf{E}^3$ intersects M in finitely many segments and every plane in finitely many areas. Analogously, we say that a compact 2-manifold in $\mathbf{E}^3$ is of bounded variation if every line intersects it in finitely many points and every plane in finitely many curves.

Throughout this book, we assume that the boundaries of solids can be described by finitely many subsets of algebraic surfaces. It is easy to show that algebraic surfaces are always of bounded variation;[6] hence, we do not

[6]Cf. Bezout's theorem in Section 5.3.3 of Chapter 5.

have to make additional requirements on B-rep to achieve embeddings of bounded variation. This simplifies verifying topological validity. Note also that embeddings of bounded variation are always tame.

Topological Validity of Manifold Solids

Consider a 3-manifold M with boundary embedded in $\mathbf{E}^3$. Its boundary is an embedded orientable 2-manifold M' without boundary, but M' need not be connected. Thus, when given an embedded oriented 2-manifold M' without boundary, we need to satisfy a compatibility condition so that its connected components will collectively define a solid. The compatibility condition depends on the given orientation of the components.

Let M be an oriented, connected 2-manifold in $\mathbf{E}^3$. We call a point p not on M *interior* to M if there is a triangulation C of M satisfying the following conditions:

1. There is a 2-simplex S of C oriented as (p_0, p_1, p_2) such that the corresponding points on M are seen from p in a counterclockwise orientation.

2. There is a point q in the image of S such that the line segment (p, q) does not intersect M except at q.

Figure 2.49 illustrates the definition. It is not difficult to show that, with this definition, the manifold partitions $\mathbf{E}^3$ into two open sets, one consisting of all points that are interior, the other consisting of points that are neither interior nor on M. The latter set is the set of *exterior* points.

Note that the definition of interior and exterior points agrees locally with the convention of outward-pointing normals explained in Section 3.2 of Chapter 3. Let (p_0, p_1, p_2) be an oriented 2-simplex. For an acute angle, say at p_0, define a normal direction as the cross product of the vectors $\overline{(p_0, p_2)}$ and $\overline{(p_0, p_1)}$. Then this normal direction points locally to the exterior.

Given this definition of interior and exterior points, we can define a manifold solid as follows:

Definition
A *single-shell manifold solid* A is a connected 3-manifold with boundary embedded in $\mathbf{E}^3$. The boundary A' of A is a compact, connected 2-manifold without boundary, embedded in $\mathbf{E}^3$ with bounded variation, and is oriented such that A consists of the set of interior points of A' along with A'.

Now assume that we have two disjoint connected oriented 2-manifolds, M_1' and M_2', in $\mathbf{E}^3$. Each defines a set of interior points. We say that M_1' and M_2' are oriented *consistently* if M_1' consists of interior points of M_2' whenever M_2' consists of interior points of M_1'. Figure 2.50 illustrates the definition for the two-dimensional case. The bounding cycles on the left are inconsistently oriented, those on the right are consistently oriented. Consequently, the following is a definition of a multishell manifold solid.

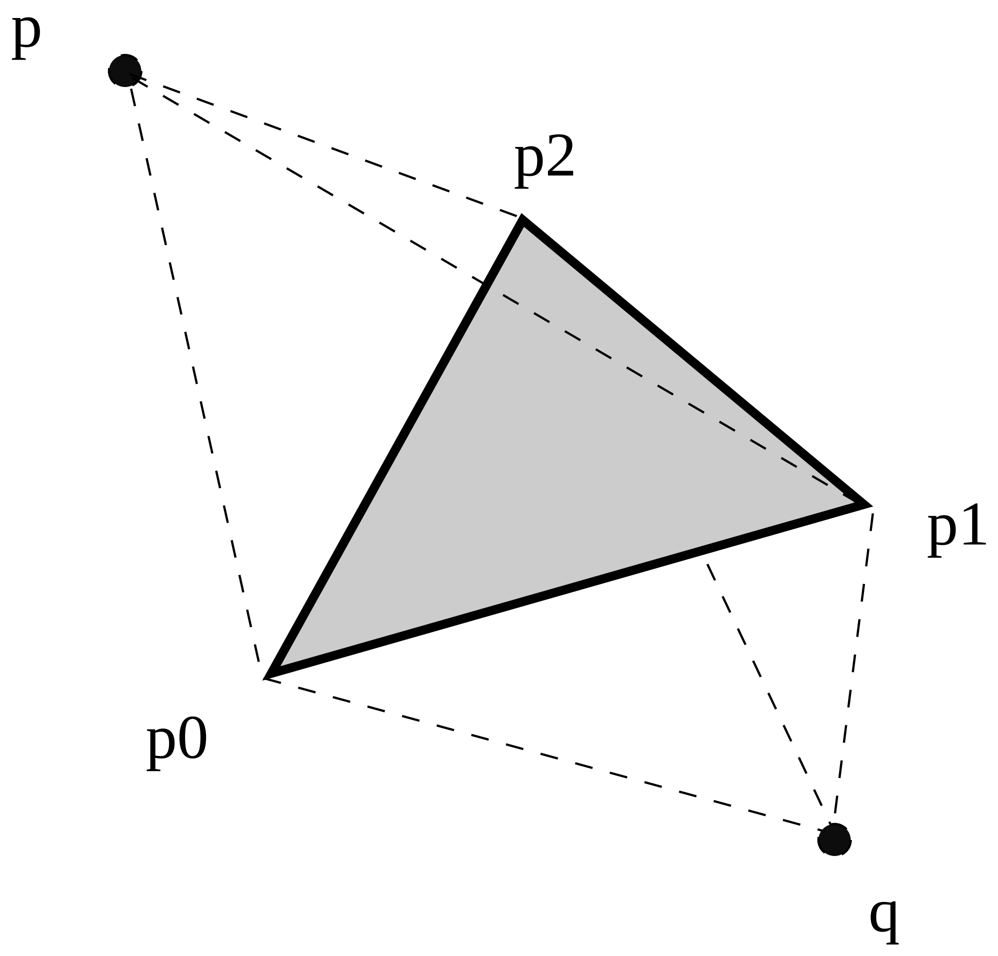

Figure 2.49 Interior Point p and Exterior Point q

Definition
A *multishell manifold solid* A is a 3-manifold with boundary in $\mathbf{E}^3$. The boundary A' of A is a compact, oriented 2-manifold without boundary, embedded in $\mathbf{E}^3$ with bounded variation. Let $A'_1, ..., A'_r$ be the connected components of A'. Then, for each pair (i, k), where $1 \leq i < k \leq r$, the components A'_i and A'_k are consistently oriented.

Topological validity of a manifold B-rep solid is verified as follows. We assume that we are given a boundary representation specifying finitely many vertices, edges, and faces. Moreover, we assume that each face is a compact subset of an algebraic surface, and that edges and vertices are intersections of faces. These assumptions guarantee that the surface is of bounded variation and that it is compact.

Let $A'_1, ..., A'_r$ be the connected surface components described by the representation. Perform the following steps.

1. Verify that the A'_k are pairwise disjoint and that they do not self-

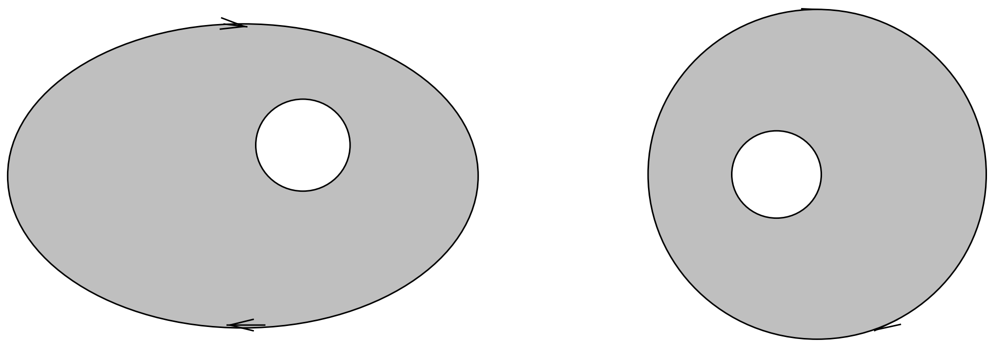
Figure 2.50 Inconsistently and Consistently Oriented Boundaries of Orientable 2-Manifolds

intersect.

2. Triangulate each A'_k and verify that it is a 2-manifold without boundary, using the triangulability conditions explained previously.

3. For each pair A'_i and A'_k of components, verify that their orientation is consistent.

A topologically valid B-rep must satisfy all these criteria.

Step 1 tests that the geometric embedding into $\mathbf{E}^3$ makes sense. Self-intersections create nonmanifold points on the boundary; hence, they must not occur. This is primarily a geometric property of the embedding. Step 2 verifies that the boundary is a 2-manifold without boundary. The assumptions on faces imply that the boundary is compact. It can be proved that a closed 2-manifold embedded in $\mathbf{E}^3$ must be orientable. Hence, orientability need not be tested explicitly. Note, however, that the triangulation of the surface is computationally easy only for planar faces. Step 3, finally, tests whether the individual shells are oriented such that they collectively define the solid's interior and exterior.

All algebraic surfaces have bounded variation, and a B-rep describes finitely many compact subsets of algebraic surfaces. Therefore, the theorem above implies that a B-rep satisfying the conditions checked for in the algorithm defines a topologically valid solid.

2.4.3 Nonmanifold Solids

We sketch how to characterize nonmanifold solids from a topological perspective. Briefly, we consider a nonmanifold solid as an *immersion* of several manifold solids; that is, we allow the manifolds to intersect in $\mathbf{E}^3$. However, we restrict intersections to sets of dimension 1 or 0. This point of view was explained in Section 2.3.1 as the second approach to defining valid boundary representations. That is, each such intersection is considered a geometric coincidence of topologically different parts of the manifold boundary.

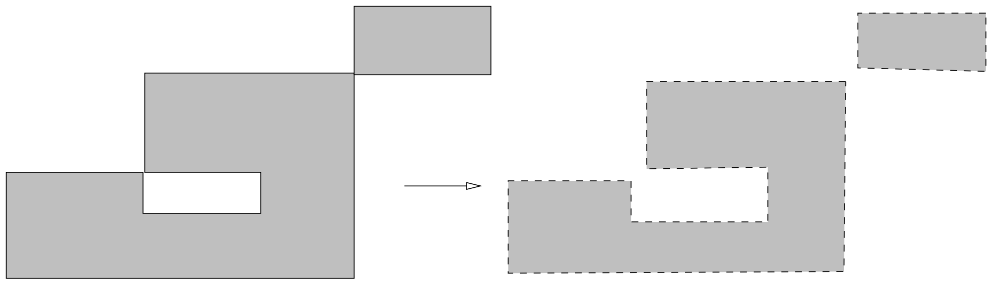

Figure 2.51 Resolution of Nonmanifold Structures

To identify the various manifold solids, we triangulate the nonmanifold solid and consider the interior points of the resulting simplicial complex. They decompose into several connected, open sets, each of which can be understood as the interior of an immersed 3-manifold with boundary.

Reversing this process, we *define* a nonmanifold solid as homeomorphic to an oriented simplicial complex of dimension 3, such that the interior points are an embedded 3-manifold of bounded variation. The closure of the set creates a boundary that is compact and of bounded variation. In particular, this implies that the surface of a nonmanifold solid can be triangulated without degenerate 2-simplices of zero area.

We adapt the validity test for manifold solid boundary representations to test whether a given B-rep describes a valid nonmanifold solid. Briefly, we must "split" nonmanifold edges and vertices by locally considering the solid interior, as implied by the given surface orientation. In effect, we construct a set of surface components, each a 2-manifold without boundary. Each of these components are then tested as described previously.

Intuitively, the splitting process "shrinks" the volume of the solid infinitesimally. After shrinking, the nonmanifold edges and vertices disappear. See also Figure 2.51 for a two-dimensional example. Note, however, that this topological resolution of nonmanifold edges and vertices will be different in a solid A and its complement solid, $\neg A$, because we shrink the interior of the solid. Hence, shrinking the complement solid $\neg A$ is equivalent to expanding the solid A.

2.5 Spatial Decomposition

Apart from CSG and B-rep, there exist a number of solid representation schemata, based, loosely speaking, on spatial decomposition. We briefly mention these schemata now. Most of them play a peripheral role in solid modeling because of certain limitations. Nevertheless, many of them have important special applications, including numerical analysis and geographic databases.

The simplest decomposition schema is a *uniform subdivision* of space into

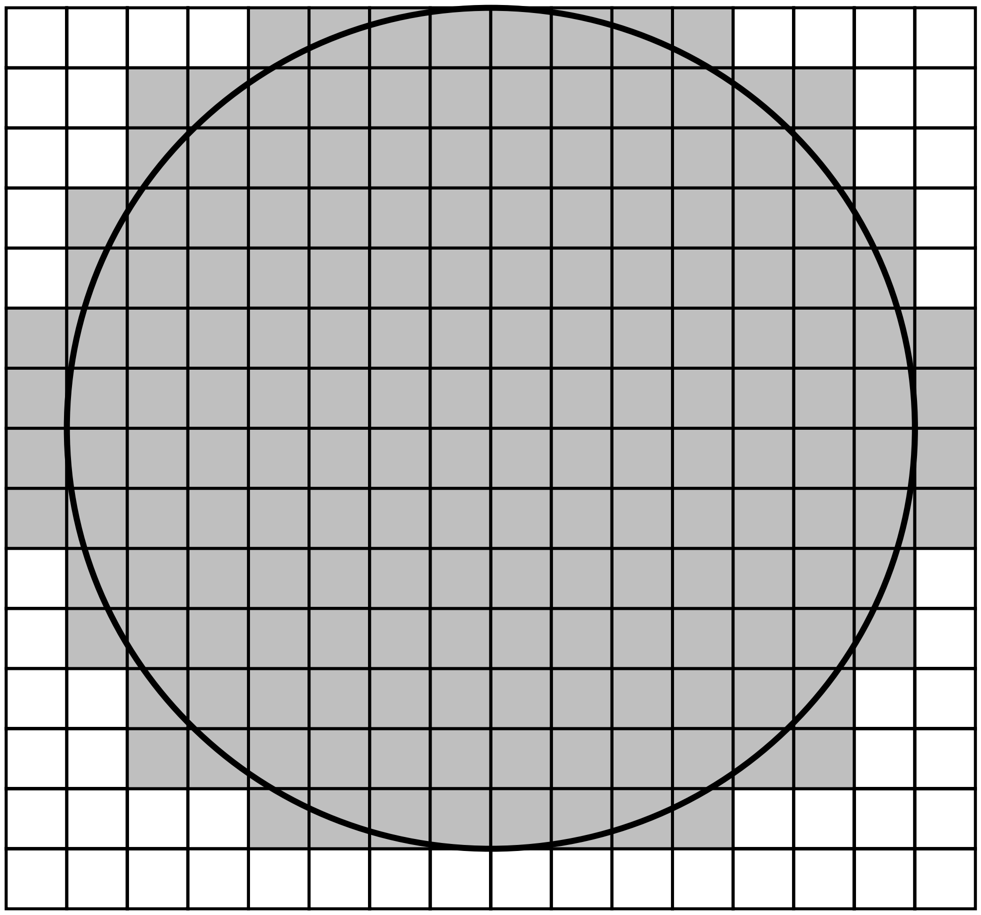

Figure 2.52 Circle Represented by Uniform Subdivision of Space

a grid of cubes of specified size, and *marks* all cubes that intersect the solid interior. The solid would thus be represented by all marked cubes. Figure 2.52 illustrates the idea in two dimensions. This representation is approximate, and the size of the cubes will define the degree of accuracy. In principle, only the marked cubes need to be stored. Adjacency of two cubes could be represented explicitly, or could be inferred from a cube's location in space. Representations of this type are used in numerical analysis for domain discretization.

When a high degree of accuracy is required, the number of cubes may be too great for this data structure to be considered convenient. *Octrees* ameliorate this problem by aggregating certain marked cubes into larger ones. Conceptually, we partition space by several grids, each with mesh size twice that of the previous one. Eight adjacent marked cubes are combined into a larger marked cube, provided the larger cube lies in the next larger mesh. Figure 2.53 illustrates the idea in two dimensions. Octrees are stored as trees that essentially record this adjacency information. The interior nodes represent cube aggregations where not all component cubes are marked, whereas

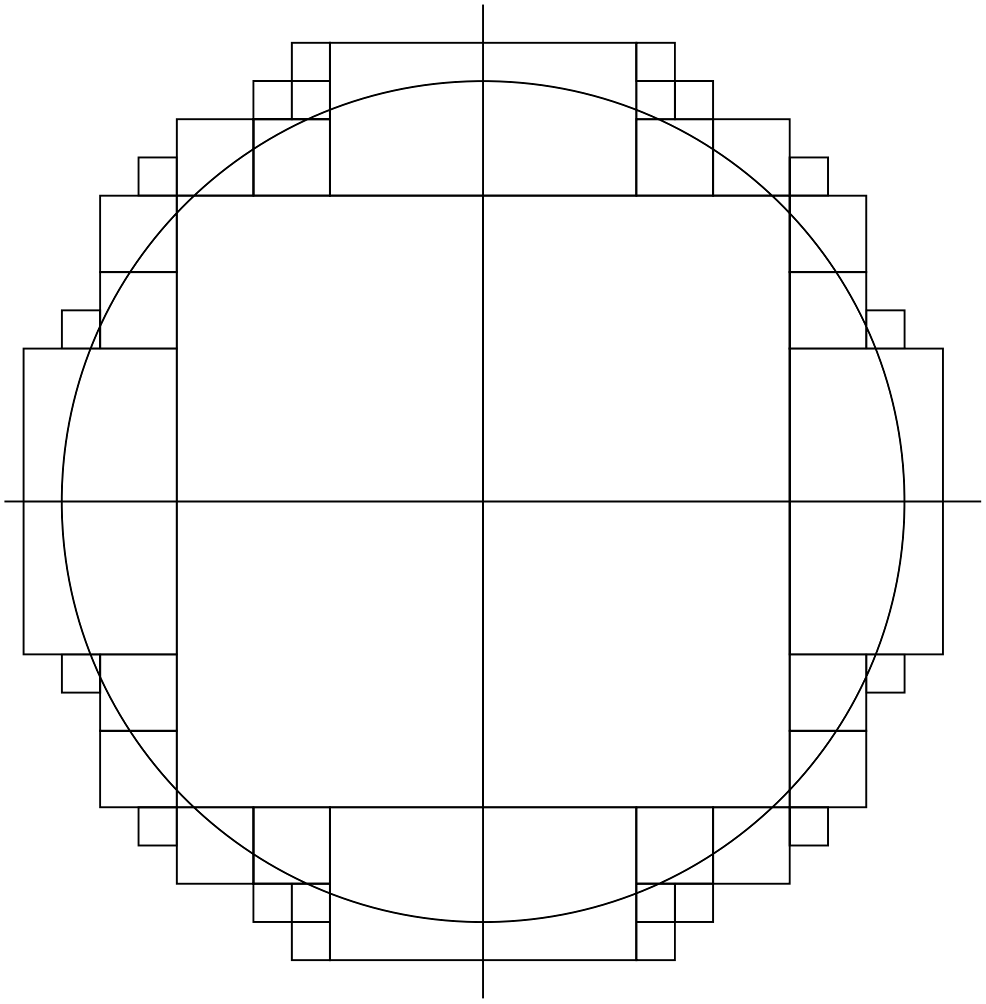

Figure 2.53 Octree Subdivision of Space

the leaves represent cubes that are marked or that do not intersect the solid interior at all.

So far, we have discussed space decompositions in which the space elements are in a specific implicit relationship with the coordinate system. This makes it unattractive to rotate objects so represented. At the expense of more complex adjacency and shape processing, we can drop this relationship and allow irregular shape elements. In finite element analysis, triangular and tetrahedral space elements are used, as illustrated by Figure 2.54.

2.6 Notes and References

Early work on constructive solid geometry includes the work on TIPS by Okino, Kakazu, and Kubo (1973), and the work on PADL by Voelcker et al. (1974) and Requicha and Voelcker (1977). Most of the CSG material

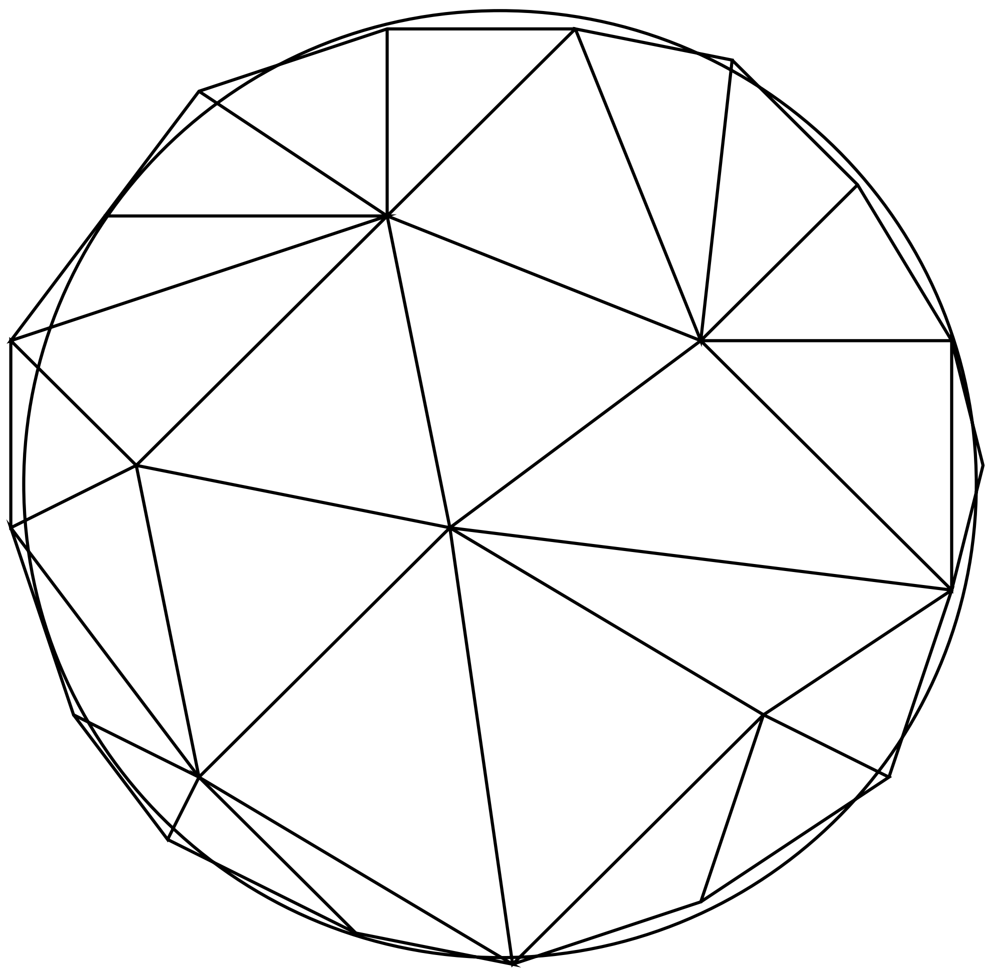

Figure 2.54 Irregular Subdivision of Space

presented here follows the various publications by that group, which relied heavily on an algebraic formulation of the algorithms. See the bibliography under the author names Brown, Tilove, Requicha, Rossignac, and Voelcker.

The approximation algorithm to CSG objects is due to Cameron (1985). By considering different approximation strategies, including approximating only some of the primitives, Cameron devises several redundancy tests. Cameron's algorithm also includes a downward phase in which the approximation at the root of the CSG tree is propagated downward toward the leaves, possibly further refining the approximation. Rossignac and Voelcker (1988) consider redundancy determination without approximating primitives. Their idea is to identify those surface areas of a primitive P that contribute to the surface of the final solid defined by a CSG tree T. By analyzing the volumes defined by the subtrees descending from the nodes on the path from P to the root of T, they derive a description of a volume that must contain the surface area of interest. They call this volume the *active zone* of P, and show how knowledge of the active zone can be used to improve conversion from CSG to

B-rep, detection of redundancies, and other operations on CSG trees. The relationship between the active zone and the approximation approaches to redundancy testing is described in Cameron and Rossignac (1988).

The winged-edge style of boundary representation is due to Baumgart (1975). Many variants of the method, as well as several alternatives, have been proposed and used in B-rep–based modeling systems since then. For a survey of the various representation schemata, see Weiler (1986). Weiler's thesis also contains much material on Euler operators, and on the problem of whether a specific representation method is minimally topologically complete. Nonmanifold boundary representations were apparently first proposed by Wesley (1980). They were again advocated by Weiler (1986) and by Hoffmann, Hopcroft, and Karasick (1987). In each case, the motivation seems to have been the observation that the internals of a number of geometric operations on polyhedra simplify when nonmanifold structures are permitted.

Paoluzzi et al. (1986 and 1988) implement Boolean operations on B-rep solids by disambiguating the topology, as discussed in Section 2.3.1. They assume that all faces have been triangulated before constructing the intersection or union of two polyhedra. With this restriction, they obtain a uniform data structure representing polyhedra. They show that the needed storage is at most 50 percent more than that of a winged-edge representation with untriangulated faces.

Mäntylä (1984) proves that Euler operations form a complete set of modeling primitives for manifold solids. That is, every topologically valid polyhedron can be constructed from an initial polyhedron by a finite sequence of Euler operations. In Mäntylä (1988), the use of Euler operations to implement Boolean operations on polyhedra is explained. The explanation of Euler operators in Chiyokura (1988) is more explicit on the interaction of the topological and geometric aspects.

The section on topological validity reviews standard material from algebraic topology. Good sources on the subject include Aleksandrov (1956), Hocking and Young (1961), Schubert (1964), and Seifert and Threlfall (1947). Hocking and Young (1961) give pictures of wildly embedded manifolds, including Alexander's horned sphere and Antoine's necklace.

The Euler–Poincaré formula can be generalized to manifolds of arbitrary dimension; it is then called the Euler–Poincaré characteristic of the manifold. It is related to the dimensions of the homology groups of the manifold.

Requicha (1977) defines solids not only topologically but also geometrically. The topological characterization is similar to ours. Requicha also shows how the Euler–Poincaré characteristic can be derived from homology computations. Octree and other spatial subdivision schemata are presented in depth in Samet (1989a,b).