UNIVERSITY OF WISCONSIN MADISON

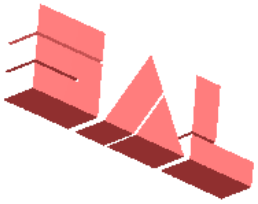
TECHNICAL REPORT
SAL 2001-2

Solid Modeling

To be published in
Handbook of
Computer Aided Geometric Design
G. Farin, J. Hoschek, M.-S. Kim, eds.
Elsevier Science Publishers, 2001

Vadim Shapiro
vshapiro@engr.wisc.edu

October 2001



SPATIAL AUTOMATION LABORATORY

UNIVERSITY OF WISCONSIN - MADISON
1513 UNIVERSITY AVENUE
MADISON, WI 53706-1572

Contents

Solid Modeling

Vadim Shapiro[a] *

[a] Mechanical Engineering & Computer Sciences, University of Wisconsin
1513 University Avenue, Madison, Wisconsin 53705 USA, vshapiro@engr.wisc.edu

Solid modeling is a consistent set of principles for mathematical and computer modeling of three-dimensional solids. The collection evolved over the last thirty years, and is now mature enough to be termed a discipline. Its major themes are theoretical foundations, geometric and topological representations, algorithms, systems, and applications.

Solid modeling is distinguished from other areas in geometric modeling and computing by its emphasis on informational completeness, physical fidelity, and universality. This article revisits the main ideas and foundations of solid modeling in engineering, summarizes recent progress and bottlenecks, and speculates on possible future directions.

1. Introduction

1.1. A premise of informational completeness

The notion of solid modeling, as practiced today,[2] was developed in the early to mid-1970's, in response to a very specific need for *informational completeness* in mechanical geometric modeling systems. This important notion has been promoted largely through the work at the University of Rochester [127] and remains central to understanding the nature, the power and the limitations of solid modeling. The concept may appear academic and redundant in the context of any one particular geometric application or algorithm, because it simply implies that the computed results should always be correct. But solid modeling was conceived as a *universal* technology for developing engineering languages and systems that must guarantee and maintain their integrity in the following precise sense.

- Any constructed representation should be *valid* in the sense that it should correspond to some real physical object;
- Any constructed representation should represent *unambiguously* the corresponding physical object;
- The representation should support (at least in principle) *any and all geometric queries* that may be asked of the corresponding physical object.

*This survey was written by the author during his sabbatical stay at the Dip. Informatica e Automazione, Universita' Roma Tre, Italy

[2] It is possible to trace the origins and techniques of solid modeling to the early beginning of computerized geometry systems in 50's and 60's [87]; there are also interesting connections to the well documented evolution of engineering drawings and descriptive geometry [12], and even to the earlier methods of synthetic geometry employed by Greeks and Egyptians more than two thousand years ago.

Informational completeness is often expressed as the last of these requirements because in fact it is assumed to imply the first two. Implicit in the above requirements is recognition that the same physical object may be represented in more than one way, and any two such valid representations must be consistent. The difficulties arise because all requirements are stated in terms of 'physical objects' that cannot be used for objective tests or unambiguous comparisons. To be clear, a *human* operator may indeed be able to render an ambiguous and subjective judgement of whether a computer program performs as desired, but no computer program can check if given computer representations are informationally complete until the notion of physical object is *defined* in terms of computable mathematical properties and independently of any particular computer representation. Without such definitions, there can be no guarantees, no automation, no standards, and no solid modeling.

Similar reasoning led Requicha and Voelcker to propose a modeling paradigm in Figure 1 that shaped the field of solid modeling as we know it today [81, 86]. The real world artifacts and the associated processes are abstracted by *postulated* mathematical models. The space of mathematical models and operations serves as the definition for the corresponding data type (class) that can be represented on a computer (in more than one way) by a representation 'scheme'. Formally, a representation scheme can be defined as a mapping from a computer structure to a well-defined mathematical object [81]. Finally, representation schemes and accompanying algorithms are organized into systems and software applications that emulate the behavior of the real world artifacts and processes.

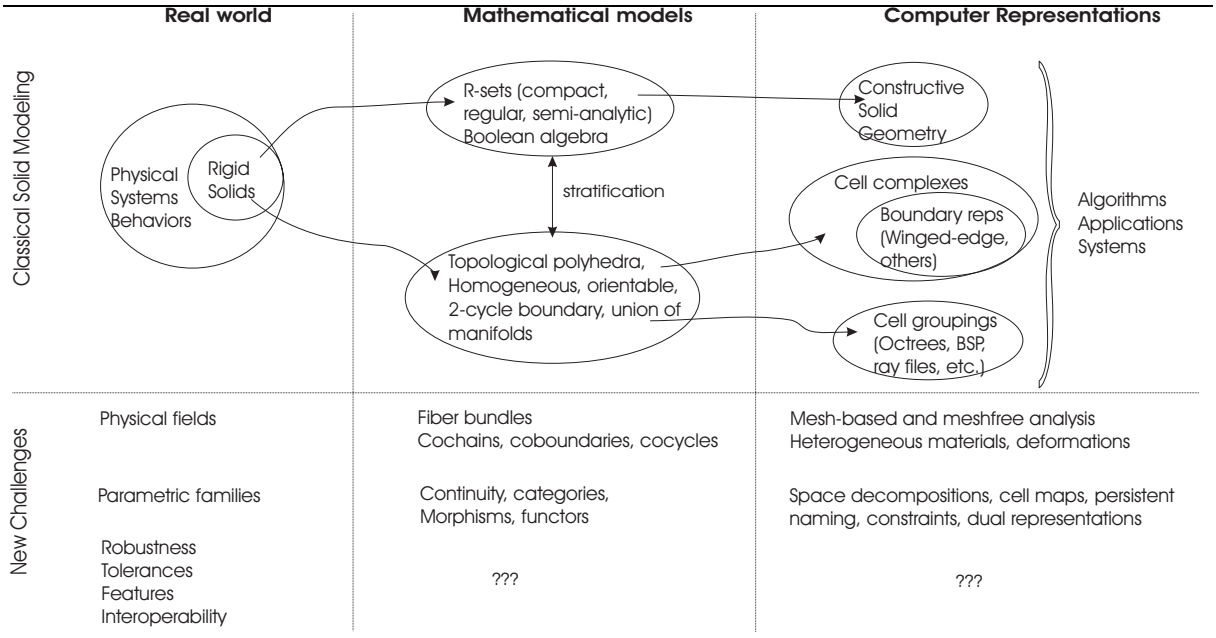


Figure 1. Modeling paradigm from [81] updated to reflect current understanding of concepts, mathematical modeling, and computer representations in solid modeling.

1.2. Outline

Following the modeling paradigm in Figure 1, it is common to survey the field of solid modeling in terms of developments related to mathematical models, representations and representation conversions, algorithms, systems, and applications. This paper adopts a similar approach. However, solid modeling is now a mature field with hundreds of relevant papers published every year in each of the above categories; many of these developments are covered in other chapters of this volume or other recent surveys. Accordingly, this chapter focuses on those aspects of solid modeling that distinguish it from other areas in geometric computing – specifically, on informational completeness, physical fidelity, and universality of representations and algorithms. As pointed out in [95], graphics, visualization, video, imaging, and many other scientific and consumer applications use and rely on solid modeling, but until now they have not driven the development of this field, perhaps because they do not appear to be critically dependent on its key characteristics. The concluding section provides a brief summary and speculates on the future of solid modeling.

Readers who are interested in a more traditional exposition to solid modeling techniques will do well by reading the landmark paper by Requicha [81], the earlier surveys of solid modeling [32, 34, 73, 79, 87, 88, 93–95], and the Proceedings of the ACM Symposia on Solid Modeling and Applications [1]. Several monographs treat important subtopics in solid modeling [33, 55] at various levels of detail, but the field has developed rapidly and no comprehensive up-to-date text on solid modeling is available as of this writing.

2. Mathematical Models

2.1. First postulates

Early efforts in solid modeling focused on replacing engineering drawings with geometrically unambiguous computer models capable of supporting a variety of automated engineering tasks, including geometric design (shaping) and visualization of mechanical components and assemblies, computation of their integral (mass, volume, surface) properties, simulations of mechanisms and numerically controlled machining processes, and interference detection. These developments are described in several often cited articles [87, 88, 127] and culminated in the Requicha's paper [81] postulating the desired properties of solid objects. All manufactured mechanical components have *finite size*; they should also have *well-behaved boundaries* that can be displayed and manipulated on a computer; the initial focus was on the *rigid* parts made of *homogeneous isotropic* material that could be *added* or *removed*. These postulated properties can be translated into properties of subsets of the three-dimensional Euclidean space E^3.

To have a finite size, the subsets must be bounded, and rigidity is readily formulated in terms of congruence under rotations and translations.[3] The requirement of well-behaved boundary is usually interpreted to mean that the set's boundary can be described by a finite collection of piecewise smooth patches, or equivalently can be finitely triangulated. In addition, the collection of the sets should be closed under several set operations: material addition and removal roughly correspond to the set union and difference operations, while interference between two such sets can be modeled by a set intersection. The class of *semi-analytic* sets satisfies all these

[3]It should be clear from the other chapters of this handbook that more general transformations in the four-dimensional projective spaces offer great computational advantages for representation and visualization of curves and surfaces in three-dimensional Euclidean space.

requirements and is defined to include all those sets that can be represented by finite Boolean combinations of inequalities of the form $f_i(x, y, z) \geq 0$, where f_i is an analytic function (in the sense of admitting the Taylor series expansion about any point in space). By definition, semi-analytic sets are closed under the Boolean set operations and include the subclass of semi-algebraic sets, as well as all sets represented by polynomial and rational equalities and inequalities. Closure, projection, and connected components of a semi-analytic set are all semi-analytic, and bounded semi-analytic sets are finitely triangulable [30, 53, 80].

But not all bounded semi-analytic subsets of Euclidean space correspond to the intuitive notion of "solid". Semi-analytic sets may be open, closed, or neither; they may also be heterogeneous in dimension. A proper solid should be homogeneous in dimension and should contain its boundary. This notion of solidity can be characterized in more than one way mathematically. The two common approaches to defining solidity rely respectively on the point-set topology and the (combinatorial) algebraic topology. Both are important because they give rise to complementary models and computer representations: the point-set model defines the local test for solidity, while the combinatorial model specifies how solids can be built up from simple pieces (cells).

2.2. Continuum point set model of solidity

For any subset X of the three-dimensional Euclidean space E^3, the points of E^3 can be classified according to their neighborhoods[4] with respect to X: a full ball neighborhood indicates that the point belongs to the interior of X; points with partial neighborhoods belong to the boundary of X. For X to be considered solid, it should contain only interior points and all those boundary points that have some interior points nearby; all other points with eroded lower-dimensional neighborhoods indicate the lack of solidity. See example in Figure 2.

Formally, set X is called *closed regular*[5] if

$$X = closure(interior(X)) \tag{1}$$

Based on this definition, introduced and studied in [80, 84, 120], we can now check the neighborhoods of individual points in the set X to see if they pass the neighborhood test. If all points pass, X is indeed solid; otherwise the set is not solid, but we can *regularize* (and therefore solidify) any set X by taking the closure of its interior, as shown in Figure 2. Obviously, regularization can both add points to and/or remove points from the otherwise non-regular set.

But now we appear to have a problem: Figure 2 shows that the intersection of two closed regular sets need not be regular, and the set difference of two closed regular sets is usually not even closed. The problem is solved by requiring that the results of the usual set operations are followed by the additional step of regularization. These new regularized set operations are denoted by $\cap^*$, $\cup^*$, and $-^*$ respectively; they indeed guarantee the solidity of the results, even if the outcome may sometimes be counter-intuitive. For example, the regularized intersection of two solids with overlapping boundaries is empty by this definition, as long as their interiors do not intersect. There may seem to be little reason to expect that these new operations should

[4] In this context, the neighborhood of a point p in set X is an open ball of sufficiently small radius, centered at p, intersected with X.

[5] The dual definition of solidity using *open regular* sets was advocated in [5] to allow overlap of boundaries for assembly modeling, but it did not catch on.

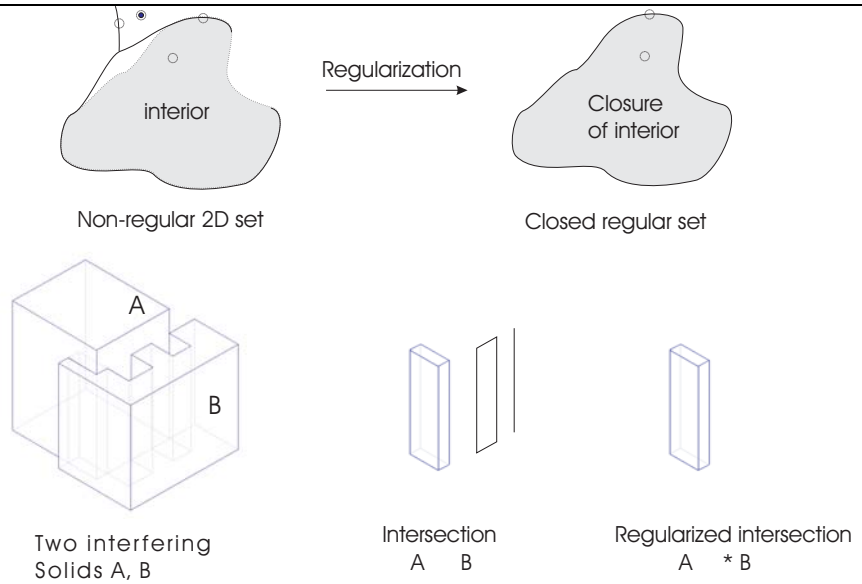


Figure 2. Closed regular sets capture the continuum notion of solidity in terms of neighborhoods of individual points in the set. Solids are not closed under non-regularized set opertions, but closed regular sets form a Boolean algebra with regularized set operations.

possess any algebraic properties related to the usual non-regularized set operations, but they do: closed regular sets form a new Boolean algebra under the regularized set operations $\cap^*$, $\cup^*$, and $-^*$ [44, 58, 84]. [6]

Closed regular semi-analytic and bounded sets are called r*-sets* following Requicha [80, 81]. The same formalism conveniently applies to planar shapes and surface patches (two-dimensional solids), solid lines and curve segments (solid curve), and so on – by simply changing the dimension and/or type of the reference universal set, which in turn modifies what we mean by a neighborhood or a ball.

2.3. Combinatorial model of solidity

Another way to characterize the set as solid is combinatorially, i.e. as composed of many solid but simple pieces (not necessarily of the same dimension), usually called *cells.* As subsets of Euclidean space, all cells are required to be orientable, with one of two orientations corresponding to an arbitrarily chosen sense of direction. Theoretically, the particular choice of cells is not very important, because a solid is characterized by the mere existence of a non-unique decomposition into primitive solid cells.[7] For example, all semi-analytic sets can be decomposed into very coarse disjoint manifold subsets of various dimensions as shown in Figure 3; the resulting submanifolds are called *strata* and the corresponding decomposition a *stratification* [131]. Alternatively, any semi-analytic set may be *triangulated,* i.e., decomposed into a collection of curved triangles (points, curve segments, triangular surface patches, and tetrahe-

[6] Note that the commonly used term "Boolean operations" is ambiguous in the larger context of geometric modeling because it is used to refer to either standard, or regularized, or both types of set operations.

[7] One should not confuse this theoretical issue with practical representational and algorithmic consideration in *representing* solids on a computer using cellular structures which we consider in section 3.2.

dral elements) [53]; triangles often play the role of the simplest common denominator cells in the sense that all other cells may be further triangulated.

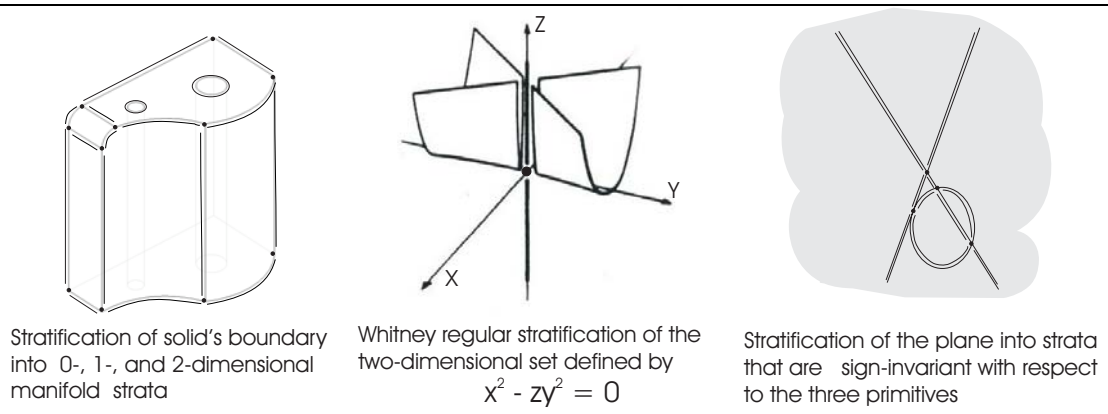


Figure 3. A minimal Whitney regular stratification of a set into connected manifold strata satisfies the frontier condition: the boundary of every stratum is a union of other strata.

Because a combinatorial model is defined in a cell-by-cell fashion, all geometric computations are reduced to presumably simpler computations on individual cells. The cells are usually chosen to be disjoint (for open cells) or to have disjoint interiors (for closed cells) and properly joined together into a *cell complex* so that they can also provide finite 'spatial addresses' for points in an otherwise innumerable continuum.[8] Formally, the proper joining of cells amounts to satisfying the *frontier condition* which requires that the (relative) boundary of every cell is a finite union of other cells in the complex. Intuitively, this means that all points in any stratum are alike: their neighbourhoods are homeomorphic to each other and they all meet the same other strata. This combinatorial model of solidity is usually summarized by saying that, in addition to being semi-analytic bounded subsets of Euclidean space, solids are homogeneously n-dimensional topological polyhedra [3, 4, 80, 81].

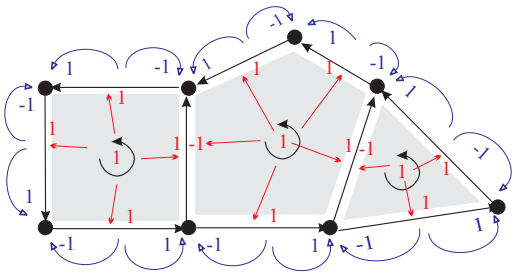
Arguably the most important property of a topological polyhedron is its combinatorial *boundary* which is itself a lower dimensional polyhedron and can be obtained by a pure algebraic computation using the concept of *chains*. Given a cell complex K, a *p-dimensional chain*, or simply *p-chain*, is a formal (i.e., cell-by-cell) sum

$$a_1\sigma_1 + a_2\sigma_2 + \ldots + a_n\sigma_n, \tag{2}$$

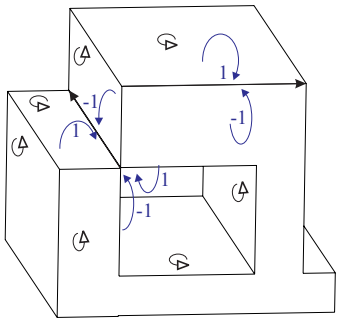
where σ_i are p-dimensional cells of K and a_i are integer coefficients. Two p-chains on a cell complex can be added together by collecting and adding the coefficients on the same cells. The collection of all p-chains on K form a group, and using chains we can replace incidence, adjacency, and orientation computations with a simple algebra. In particular, Figure 4 illustrates

[8]The notions of open, closed, closure, interior, etc. are all relative to the larger set; in our context, usually a curve, a surface, a volume, or a k-manifold.

how the oriented boundary ∂ operation can be defined algebraically in terms of elementary chains using only three coefficients from the set $\{-1, 0, +1\}$ [3, 31, 63].



Boundary operation on k-chain transfers the coefficients from every k-cell to all incident (k-1)-cells with +/- sign depending on relative orientation. Addition cancels the interior k-cells, and yields the (k-1)-boundary. Repeating the operation produces a (k-2)-chain with all 0-coefficients.



Boundary of a three-dimensional solid must be a 2-chain whose boundary is 0, i.e. a 2-cycle. It does not have to be 2-manifold, but every 1-cell (edge) is shared by an even number of 2-cells (faces).

Figure 4. Combinatorial model of solidity allows algebraic definition of topological properties in terms of k-dimensional chains with coefficients 0,1,−1.

Starting with a 3-dimensional solid represented by a 3-chain, the boundary operation ∂ produces an oriented 2-chain (and the corresponding 2-dimensional cell complex) that defines the 2-dimensional boundary of the original solid. If we apply the boundary operation again to the 2-chain, we obtain a 1-chain with all zero coefficients. In other words, the fundamental property of ∂ is that

$$\partial(\partial(X)) = 0 \tag{3}$$

and guarantees that the boundary set is one or more "closed surfaces" sometimes also called "shells." Formally, such a set whose boundary is a zero chain is called a 2-cycle [31, 81].

For historical reasons [8, 16], a more restrictive model of solidity has often been adapted where topological polyhedra are restricted to the orientable three-dimensional *manifolds* with boundary.[9] With this restriction, the combinatorial boundary, defined as above, is not only a 2-cycle, but also a 2-manifold. This manifold model of solidity claims two theoretical advantages: (1) it rules out non-manifold sets that are sometimes deemed non-physical, and (2) the connected manifold boundaries are completely classified in terms of their Euler characteristics (see section 4.1). As we explain below, manifold models are also easier to represent on a computer. But unfortunately, this restriction to manifold solids violates one of the important postulated properties – that solids be closed under the regularized set union and intersection. (The union of two manifold sets touching at a point is a non-manifold set.) On the other hand, it is important to recognize that every 2-cycle is indeed a union of 2-manifolds.

[9] In a k-manifold with boundary, every point has a neighborhood that is homeomorphic to either a k-dimensional open ball (if the point is an interior point), or a k-dimensional half-ball (if the point lies on the boundary.

2.4. Generalizations

More general computer representations of mechanical artifacts often include unbounded and trimmed curves and surfaces (used for aesthetic, reference, manufacturing, and other purposes), which are combined with traditional solid models (to represent cracks, or material heterogeneity, to simulate surface forming, and so on). This in turn means that the paradigm in Figure 1 needs to be expanded to include more general mathematical models, both continuum and combinatorial. The generalizations are relatively straightforward in principle, because they essentially amount to relaxing some of the original constraints. General continuum models correspond to closed semi-analytic subsets of E^3, while the general combinatorial model is readily seen as an arbitrary collection of cells from some dimensionally heterogeneous cell complex [64, 129].

Thus there appear to be two competing mathematical theories of solid modeling: point-set continuum and algebraic topological combinatorial. Fortunately, the two theories are entirely consistent, and we can use the two mathematical models interchangeably [80], relying on either continuum or combinatorial properties whenever needed. The key fact: every closed semi-analytic set is a topological polyhedron that can be represented by some finite cell complex. The class of closed regular subsets of E^d coincides precisely with that of homogeneously d-dimensional polyhedra, and furthermore, it can be shown that every 2-cycle in E^3 bounds some unique solid. We know immediately that every solid may be represented unambiguously by its boundary, and that the boundary has a combinatorial structure of a 2-d homogeneous polyhedron whose points all have homogeneously 2-dimensional neighborhoods.

3. Computer Representations

There are several ways to group various computer *representation schemes*, including Requicha's widely accepted description of six 'pure' representation schemes [81], but as more new and hybrid representation schemes are being proposed, their complete classification appears impractical. This survey takes a slightly different view that there are really only two fundamental ways to represent a point set, closely related to the two mathematical models of solidity identified above.

- *Implicit* (and constructive) representations give rules for *testing* which points belong to the set and which are not; such representations are naturally supported by the point set continuum model of solidity; and

- *Enumerative* (and combinatorial) representations specify the rules for *generating* points in the set (and no other points); such representations are closer in spirit to the combinatorial view of solidity.

This classification of representation methods is probably the coarsest possible, but recall that any informationally complete representation should in principle support any and all geometric queries. Thus, one (even if inefficient) way to test if a point belongs to the set or not is to see whether the point is among the points being generated by some enumerative representation; and a perfectly valid way to generate points in the set is to test all candidate points against some implicit representation to determine whether they belong to the set or not. Such views are not as extreme as they may appear at a first glance, and in fact all representations are 'misused' in similar fashion to support applications for which they were not originally designed. In our

higher-level view of representation schemes, the representation rules (implicit or enumerative) are essentially computable implementations of semi-analytic functions.[10]

3.1. Implicit and Constructive

A very general method for defining a set of points X is to specify a predicate A that can be evaluated on any point p of space:

$$X = \{p \mid A(p) = true\} \tag{4}$$

In other words, X is defined *implicitly* to consist of all those points that satisfy the condition specified by the predicate A. The simplest form of the predicate is the condition on the sign of some real-value function $f(p)$, resulting in the familiar representations of sets by equalities and inequalities [65, 90]. For example, if $f = ax + by + cz + d$, conditions $f(p) = 0$, $f(p) \geq 0$, and $f(p) < 0$ represent respectively a plane, a closed linear halfspace, and an open linear halfspace respectively.

More complex functions can be employed to define progressively more complex geometric shapes, giving rise to the whole discipline of implicit modeling [10]. A natural approach for semi-analytic sets is to define more complex predicates *constructively* using logical combinations of simple "primitives", which is equivalent to using the standard set operations ($\cap$, $\cup$, $-$) on the sets defined by primitive predicates. (See examples in Figure 5.) Furthermore, the theory of R-functions [98, 103] (see also the Chapter on Finite Element Approximation with Splines by Klaus Hollig) allows conversion of such representations into a single function inequality for any closed semi-analytic set.

3.1.1. Representation by point classification

Given an arbitrary point, there are at least two distinct methods for deciding its membership in the represented set. One could replace the constructive representation with a single inequality predicate whose truth is determined by testing the sign of some real-valued function at the given point. This approach leads to increasingly complex arithmetic computations for what is essentially a logical computation, and therefore appears to be a poor computational strategy except when such functions are constructed directly. A more attractive alternative is to represent the constructive geometric representation on a computer using the usual tree data structure, with the primitive predicates (defining the primitive halfspaces) stored in the leaves of tree and the logical (set) operations are stored at the interior nodes. Then, the algorithm for point membership query on this data structure can be implemented naturally by proceeding recursively down the tree and "inverting" every set construction (op) in terms of the corresponding logical operation ($\odot$):

$$p \in (X op Y) \Longrightarrow (p \in X) \odot (p \in Y) \tag{5}$$

Such algorithms are appealing because they require only arithmetic computations (at the leaves of the tree) and logical operations (at the internal nodes); the corresponding Boolean structure facilitates various speed–up techniques for pruning, localizing, restructuring, and parallelizing the computations [96, 119].

[10] In this sense, the two representation methods correspond to the two common methods for describing functions: implicitly and parametrically, the latter being essentially the continuous form of enumeration.

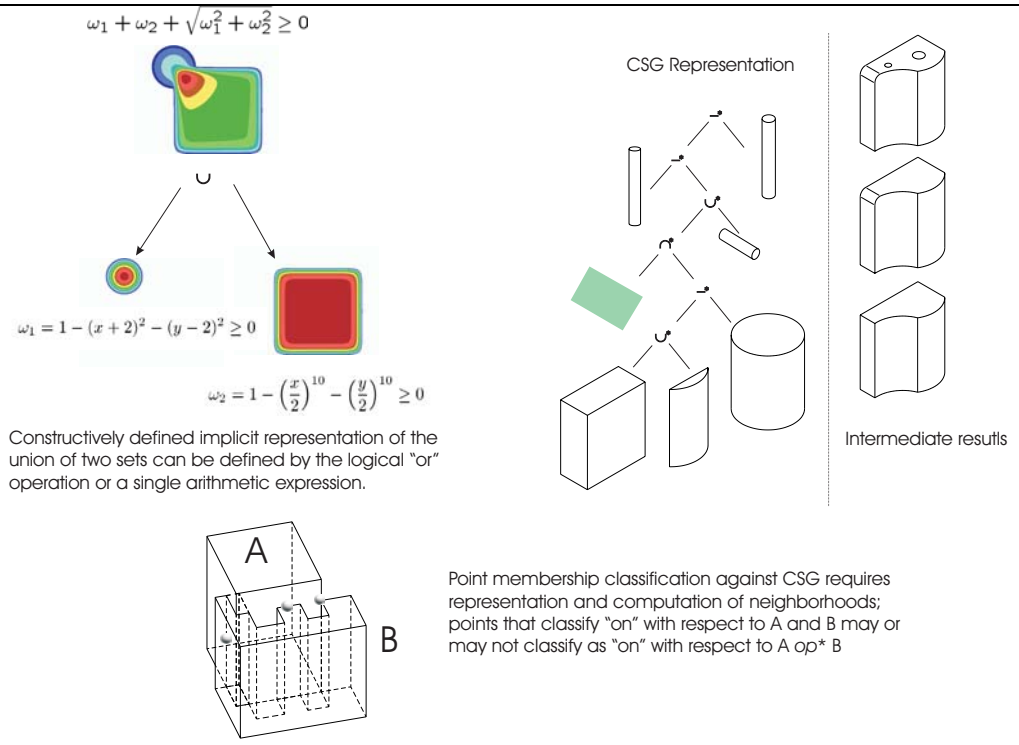


Figure 5. Constructive representations and Constructive Solid Geometry.

3.1.2. Constructive Solid Geometry

Using such constructive representations for solid modeling requires several extensions, as dictated by the mathematical properties postulated for solids in section 2. Rigid body motions of the constructively defined sets are represented by the usual method of coordinate transformations that can also be stored as an intermediate node in the constructive tree; point membership test against a set transformed by motion M is also inverted in a straightforward fashion:

$$p \in M[X] \Longrightarrow M^{-1}[p] \in X \tag{6}$$

Recall that semi-analytic sets are closed under the standard set operations but do not guarantee solidity; to implement the algebra of closed regular sets, the constructive representations are modified in two ways: (1) every primitive set is required to be closed regular; and (2) only regularized set operations are allowed. The latter may appear to be a relatively minor matter of replacing every set operation (op) by the corresponding regularized op^*. But in fact, this dramatically changes the computational properties of the constructive representations: point membership queries against the regularized constructions require substantially more work than is implied by Equation (5). Specifically, combining results of two classifications with respect to sets A and B requires not only the logical information, but also representing and combining the neighborhoods of p with respect to A, B, and Aop^*B [118, 120] (see the example in Figure 5.)

The constructive representation scheme relying on closed regular primitives, rigid body motions, and the regularized set operations is called *Constructive Solid Geometry* or CSG [85]. By design, the use of CSG is limited by availability of solid primitives and by the necessity to

represent and maintain the neighborhood information for points on primitives and their combinations. The latter task is particularly non-trivial for points with non-manifold neighborhoods and/or lying on high-degree tangent surfaces. Complete solutions have been worked out for solids bounded by planar and second degree surfaces, with only limited results available for more complex solids. The attractive properties of CSG include conciseness, guaranteed validity (by definition), computationally convenient Boolean algebraic properties, and natural control of the solid's shape in terms of high-level parameters defining the solid primitives and their positions and orientations. The relatively simple data structures and the elegant recursive algorithms further contributed to the popularity of CSG in academia and early commercial systems.

3.1.3. Other constructive representations

In principle, many other constructions may be added to the lexicon of implicit representations, notably offsetting [92], blending [133] , convolutions [11], and other skeletal-based representations, Minkowski operations [26, 62], and sweeping [37, 114]. Such constructions have numerous applications in mechanical design, analysis, and planning tasks; they also flourished in computer graphics [112] where computational time and guarantee of correctness are often deemed less important than the visually pleasing results. But while such formal definitions are sometimes straightforward, they do not always guarantee the solidity of the result and do not always support a clear point membership query – which came to be regarded as a formal test for any unambiguous representation. The most popular of these constructions is the sweep representation shown in Figure 6 (considered a distinct representation scheme in [81]), defined for a given set X and continuous motion $M(t)$ by the infinite union operation:

$$sweep(X, M) = \bigcup_{q \in M(t)} X^q \tag{7}$$

where X^q denotes set X positioned at the configuration q. In other words, $sweep(X, M)$ is the set of all points swept (or occupied) by X at some time during the motion. Sweeps are relatively well understood [2] and are useful for variety of representational tasks: computing space occupied by a moving object, material removed by a moving cutter, extrusion of a planar cross-section along one-dimensional path, and so on. The point membership test for sweep follows naturally from the studies of the dual infinite intersection operation and also inverts the construction: $p \in sweep(S, M)$ if and only if the inverted trajectory of the point $sweep(p, M^{-1})$ intersects the original solid S [39] (see Figure 6).

By definition, the constructive approach can be utilized for representing any and all semi-analytic sets, relying only on a finite set of analytic primitives and set operations. The operations of *closure* and *connected component* preserve semi-analyticity and may also be used effectively [104]. Particular representation schemes have been proposed to include extension of the classical CSG representations using topological operations in [83] and constructive representations for n-dimensional semi-analytic sets [13].

All constructive approaches are limited by their ability to compute and manage topological neighborhood information of the points in the represented sets. A related significant drawback of CSG and all other implicit representations is the lack of explicit representation and parameterization of the solid's interior and particularly its boundary.[11] This leads to several practical

[11] Observe that explicit representation of the boundary also implies explicit representation of the neighborhood information for all points of the solid.

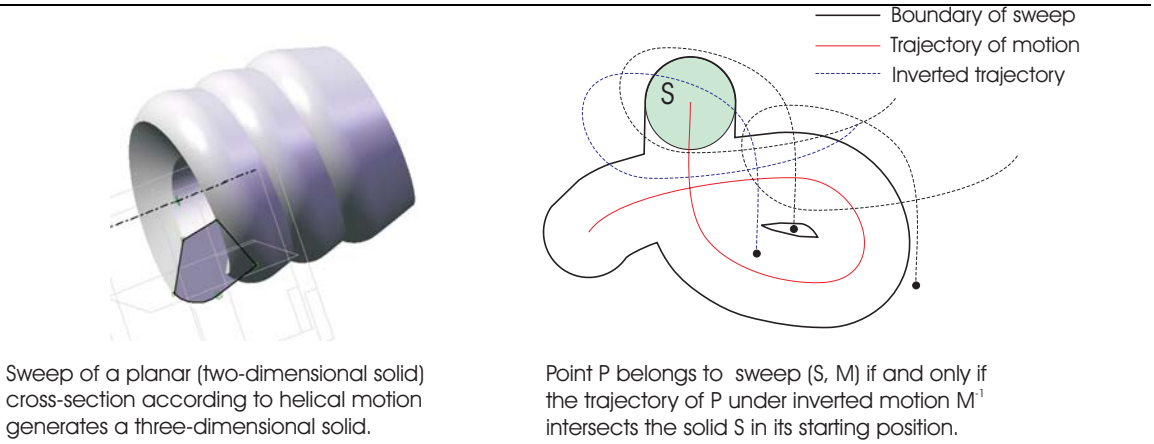


Figure 6. Sweep(S,M) is a constructive representation for a set of points occupied by S under motion M; a point membership procedure inverts the construction.

complications, including computational difficulties in generating ordered points for the purpose of display and/or engineering analysis. Without the explicit representation, spatial locations of points in the solid or its portions are not known *a priori*. In other words, the represented subsets are not *spatially addressable,* and therefore cannot be referenced persistently, for instance for attaching labels and engineering attribute information. Comparison of implicitly-represented solids is also problematic: the same solid admits infinitely many constructive representations, and even deciding if the represented set is empty may require non-trivial computations [119].

3.2. Enumerative and Combinatorial

3.2.1. Representation by enumeration

A seemingly more direct way to define which points belong to the solid and which do not is to enumerate the points by an explicit parametric rule. Thus, a planar curve is commonly defined by a mapping $[0,1] \rightarrow E^2$. For every value of $t \in [0,1]$, the points on the curve are defined by the pair of functions $x(t), y(t)$. Generating points in the defined curve segment is a matter of marching along the unit interval in small increments and evaluating the coordinate functions for every value of t. Testing if a given point belongs to the curve segment requires a more complex numerical procedure but is clearly well-defined and computable task. Similarly, one can define parametric surface patches and tri-variate solids by mappings from the unit square or cube into E^3. Many chapters in this handbook deal with non-trivial issues related to representation of such parametric curves and surfaces for complex shapes. As the next best thing, we can represent a complex solid by enumerating not individual points but simple solid chunks or *cells*, relying on the combinatorial properties of solids as topological polyhedra. This approach leads to persistent and spatially addressable data structures that support development of cell-by-cell *traversal* algorithms and can be controlled locally and incrementally, which is particularly convenient for point generations and local modifications. The point membership query reduces to a search procedure aimed to determine which of the represented cells (if any) contains the given point. The main drawbacks of the combinatorial data structures have to do with their size and apparent lack of means to create, validate, and manipulate such structures

directly.

Broadly, all combinatorial representations can be classified according to (1) the choice of the cells; and (2) restrictions on how the cells must fit together. What makes a good cell? The common requirements include dimensional homogeneity, connectedness, boundedness, and semi-analyticity. Cells may be relatively open or closed (depending on how they fit together to form a closed set); they may or may not be required to be smooth or simply-connected; but their representation must support one or both of the two fundamental computations: point testing or point generation. In other words, the cells should be simple enough to be unambiguously representable: either implicitly or enumeratively, or preferably both. Depending on the type of cells, solids may be assembled from cells in at least one of two distinct ways described below: as groupings or as cell complexes.

3.2.2. Groupings

Groupings[12] of solid cells of the same geometric type and dimension is probably the simplest way to represent the set. Example of groupings include: collections of three-dimensional cubes (called voxels) [110], ray-files of ray segments (finite size rectangular columns) [27], union of overlapping spherical balls, and files of two-dimensional polygons that are commonly used in computer graphics. The common principle underlying all groupings is that they are assembled from closed cells representing small chunks of space. The points in the interior of a cell are characterized by the *constant neighborhood*: all points have the same type of neighborhood with respect to E^3, and the cells are chosen simple enough to admit both implicit and parametric representations. Because all cells are of the same geometric type and dimension, groupings support development of simple and brute-force algorithms. For example, the point membership query reduces to a point-cell test that is repeated for every cell. Depending on their particular geometric type, groupings may be organized into more compact and efficient computational structures (trees, hierarchical graphs, etc.) supporting efficient queries, processed by parallel algorithms and specialized hardware. One of the most popular representations in this category is called *octree:* a hierarchical method for representing a grouping of orthogonal three-dimensional boxes (usually cubes), where each box intersecting the boundary of the solid is subdivided into eight smaller boxes, and so on until the desired level of resolution is reached[18, 59]. A grouping of convex cells defined implicitly by intersecting linear halfspaces may be more efficiently represented by a *binary space partition* (BSP) tree; each BSP node corresponds to a particular sequence of halfspaces and thus to a convex subregion of space[117].

By construction, groupings are guaranteed to be valid solids, but only some solids – those with geometry representable by the unions of cells in the grouping – can be represented exactly. Other solids can be approximated, for example, by a multi-resolution grouping, or groupings are often enhanced with additional geometric representations specifying precisely what geometry is being approximated [18, 61]. Creating groupings may be expensive and operating on them may be difficult; restrictions on groupings (for example, shape of cells or orientation) may imply that the represented solids are not closed under common transformations. Even the usual operations of rigid body motions or set operations may require substantial processing and reconstruction.

[12] We prefer the term *k-grouping* (for k-dimensional grouping) introduced recently by [25] in favor or 'spatial enumerations,' or 'sampled' representations, because we do not wish to imply an approximation or to specify the source of the represented data.

Another serious drawback of groupings has to do with lack of explicit representation for the incidence or adjacency between the cells in the grouping: no conditions are usually imposed on the neighborhoods of points with respect to neighboring cells. While the corresponding topological polyhedron can be computed in principle, it may not be a grouping itself, which implies the need for additional data structures and algorithms. For example, it is not immediately clear how to define, compute, and represent the (well-defined but lower-dimensional) boundary of a solid represented by a d-grouping. Last but not least, because groupings are inherently homogeneous combinatorial structures, they are not suitable for representing mixed-dimensional point sets.

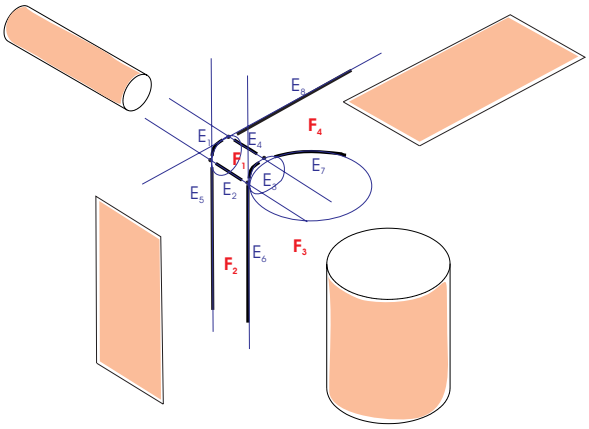
3.2.3. Cell complexes

The key difference between a grouping and a cell complex is that the latter requires neighborhoods for points in a cell to be constant not only with respect to E^3, but also with respect to the other cells. In other words, a cell complex is a representation of a stratification of a solid. Cell complex representations implement the combinatorial model of solidity directly, requiring that every solid is a topological polyhedron and may therefore be decomposed into a finite cell complex assembled from solid cells of different dimension. This effectively turns all geometric and topological queries on a given solid into simple algebraic (syntactic) computations. A number of data structures have been proposed for representing general (heterogeneous) cell complexes [29, 64, 129]. Further generalizations appear to be taking place, both in geometry and topology of the sets representable by solid modeling systems. A cellular representation of n-dimensional manifolds and sets has been proposed by [17, 50], and a functional programming language whose semantics is defined by operations on n-dimensional polyhedral cell complexes has recently emerged [70].

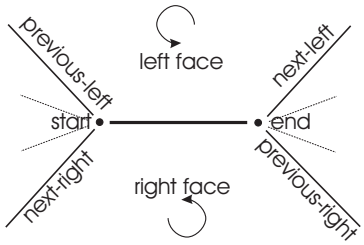
Any such representation must contain enough information to determine the geometry of every cell *and* the incidence between the cells – and therein lies the main difficulty with all such representations: geometric and incidence information are not independent. In principle, defining geometry of every cell in the cell complex is sufficient to *also* represent the incidence information through geometric tests. But in practice, geometric tests are imprecise, searching for adjacent cells is inefficient, and specifying geometry of adjacent cells independently is redundant and wasteful since incidence constrains the corresponding geometries to 'match'. Specifically, but without loss of generality, let us assume that every cell σ is a subset of a larger set that must include at least the closure of σ and is called the *carrier*[13] of σ; then by the definition of cell complex: (1) every k-cell belongs to the intersection of the carriers of all incident $(k+1)$-cells (cofaces); and (2) the carrier of every k-cell is an interpolation of all incident $(k-1)$ cells (faces). Therefore, the incidence relations themselves are a convenient method for defining the geometry of the incident carriers. The two conditions apply simultaneously and independently, even though only one of them suffices to define the complex mathematically. For example, in a simplicial complex, only coordinates of the 0-simplices (points) are represented geometrically; carriers of higher-dimensional simplices are defined as interpolations (usually linear): edges interpolate points, triangles interpolate edges, and so on. By contrast, geometry of many solid cells is defined by set operations, for example curves of intersection between second degree surfaces are commonly found in many mechanical parts. In such situations it may be more

[13] In other words, carrier is a generic dimension-independent term for possibly unbounded curve, surface, volume, etc.

convenient to define the carriers of the 2-cells (surfaces) and to define the carriers of the 1-cells (edges) implicitly as subsets of the corresponding intersection curves. Thus, it should be clear



Geometric carrier of every cell in a cell complex interpolates all incident lower dimensional cells, and every cell lies in the intersection of all carriers of the incident higher-dimensional cells. Shown above are carriers for a partial cell complex representing the boundary of the solid in Figure 3.



Winged-edge data structure represents incidence and local orientation of cells in a manifold boundary representation in terms of edge records. Each record points to ordered lists of two vertices, two faces, and two edges on each of the faces. Vertices and faces are represented by loops of edges. Geometric carriers can be associated with each cell, or implied by the interpolation and/or intersection conditions.

Figure 7. Geometric compatibility (intersection, interpolation) conditions and incidence information in cellular representations are not independent and must be maintained at all times.

that many cellular representations may be devised that would differ in the choice of which geometric carriers are specified explicitly and which are implied by the incidence relations between adjacent cells [132]. This determines in part which incidence information can be determined in constant time and which requires searching. Representing *all* incidence relationships in a cell complex substantially increases the size of the resulting data structure and is therefore deemed impractical.

The main advantage of the cell complexes over groupings is that they represent exactly and explicitly all topological information about the solid, including its interior, boundary, dimensional skeletons, and connectivity. No additional numerical computations are required to answer such topological queries, and all solids may be represented exactly – at least in principle. On the other hand, the graph representation of the incidence between the various cells and the high degree of geometric redundancy turns cell complex representations into inherently serial linear size structures and complicates development of efficient algorithms. Maintaining the validity of cellular representations is a non-trivial task requiring a guarantee that all redundant information remains consistent at all times: all represented geometric carriers must satisfy all intersection and interpolation conditions, while the cells themselves must remain disjoint as required by the definition of the cell complex [100].

3.3. Boundary representation: a compromise

Historically, boundary representation was one of the first computer representations to be used for description of polyhedral three-dimensional objects [8, 15], but both its strengths and weaknesses as a representation scheme can be appreciated better when examined in terms of properties of implicit and combinatorial representations. Recall that every solid X has the well-defined boundary ∂X, *and* the boundary of every three-dimensional solid in E^3 uniquely determines it.[14] We can use this fact to represent the solid *implicitly* by its boundary, without enumerating points in the solid's interior. In other words, we represent the solid X by the predicate:

$$X = \{p \,|\, p \in \text{set bounded by } \partial X\} \tag{8}$$

and rely on the Jordan-Brouwer theorem (a generalization of the planar Jordan curve theorem) guaranteeing that ∂X separates the Euclidean space E^3 into exactly two subsets, one of which is the bounded interior of X and the other is unbounded exterior space. Before we can declare this apparently implicit representation of solids a *bona fide* representation scheme, we need to give a computable point membership query on such a representation of X.

Suppose we have some representation for ∂X and we *know* that the represented set is the boundary of some unique solid X. If we pick two arbitrary points: point a in the interior of X and point b outside of X, then *any* path connecting a and b must intersect the boundary ∂X an odd number of times. Suppose we pick the point b to be always outside the solid X, then we can test if point a is inside or outside the solid by simply counting the number of times the path from a to b intersects ∂X: odd means a is *in* X, even means *out*. By far the easiest way to implement this is to choose any linear path from a to the point b at infinity, which reduces the point membership test to intersecting a linear ray with the set ∂X.

But how do we represent ∂X? The answer is: any way we like, as long as we can guarantee that it is indeed an unambiguous representation of some solid's boundary and that the intersection with an arbitrary line segment can be computed. One could choose to represent the boundary ∂X using any of the implicit, constructive, parametric, or combinatorial methods we already described above. Most of the methods and various combinations have been tried, but the main challenge for any boundary representation is to assure that the represented set is indeed a boundary of a valid solid. The validity conditions follow clearly from the combinatorial model of solidity, making the combinatorial representation a natural choice. Specifically, since the boundary of every solid is a 2-cycle in E^3, it must: (1) be a valid cell complex (disjoint cells satisfying the frontier); (2) be homogeneously two-dimensional (every lower-dimensional cell is in the closure of some 2-dimensional cell); (3) have every edge shared incident on an even number of faces; (4) be orientable, which means that the material side can be defined on every face in a globally consistent manner.

Furthermore, *every* such structure is guaranteed to represent a boundary of some valid solid.[15] Many cellular representations have been proposed for boundary representations, but the oldest and the most popular representations enforced the above conditions only for manifold solids whose boundary points have neighborhoods homeomorphic to two-dimensional disks. In such

[14] This statement is not as trivial as it may appear: the boundary of a semi-infinite set or the boundary of a curved face (for example lying on a sphere or a torus) in general does not uniquely determine the set it bounds, because there is usually another set with the same boundary.

[15] In fact, every 2-cycle in E^3 is orientable, but explicit representation of orientation allows to check for this condition locally and to extend applicability of boundary representations to more general sets.

boundary representations, every edge is shared by *exactly* two faces, as typified by the popular *winged-edge* data structure [8] illustrated in Figure 7. The incidence information in a winged-edge boundary representation consists of linked lists of edge records, with each record using eight pointers (pointing to two vertices, two faces, and four other edges) to enforce locally the orientability and manifoldness conditions. In this scheme, vertex and face records are defined by the ordered lists of the incident edges. Geometric carriers may be associated with any of the cells in the data structure parametrically or constructively, in a manner that satisfies all implied interpolation and intersection conditions, but the global non-interference conditions on cells are not enforced automatically and must be checked. A variety of other, similar in spirit, manifold boundary representations have been proposed in efforts to optimize the space, ease of manipulation, or handling of specific geometric carriers[23, 28, 132].

Boundary representations for non-manifold solids can be designed directly as representations of 2-cycles in Euclidean space or by treating the boundary of a non-manifold solid as a union of manifold shells.[16] Both approaches also apply to boundary representations of arbitrary heterogeneous cell complexes, that can be viewed as a union of boundary-represented manifolds of various dimensions [29, 47, 64, 129, 134].

As cellular structures, boundary representations have a number of attractive properties. Because the boundary of a solid is unique, it is possible to invoke additional conditions (e.g., smoothness, connectedness, sign-invariance, orientation, and so on) in order to define and construct the unique canonical decomposition of the boundary [111]. Due to the dimensional reduction, boundary representations can be much smaller than most three-dimensional cellular representations of the same solid. Boundary representations also inherit the disadvantages of the cellular structures mentioned above, and in particular they are non-trivial to construct and maintain. Regularized set operations and other constructions that guarantee solidity can be implemented directly on boundary representations, but this approach requires support for non-manifold boundary representations because manifold boundary representations are not closed under such operations. Another alternative is to devise a set of direct operations on boundary representations that must preserve the validity of boundary representations at all times. Both approaches have been employed and we will briefly discuss them below in section 4.

3.4. Unification of representation schemes

Based on the assumed mathematical properties, we know that all of the above representation schemes are different methods for capturing complete geometric information about the *same* class of objects: semi-analytic subsets of Euclidean space. Therefore, it must be possible to convert such representations into each other, as may be required for different applications. This in turn suggests that all representation schemes are simply different ways to organize the same geometric and topological data. What is this data?

All representation schemes are organized in terms of a finite number of operations on a given set of *primitives*. The primitives are halfspaces in constructive representations, geometric carriers in cellular structures, and geometry of the cells in groupings. In all cases, the primitives are defined by analytic equalities and inequalities. The operations on primitives either produce new primitives (via interpolation, motion, deformation, etc.) or produce semi-analytic sets using set operations ($\cap$, $\cup$, $-$), closure $\mathbf{k}$, and selecting a connected component. All other operations and

[16] Strictly speaking, the resulting structure is not a proper cell complex, because it contains topologically distinct but geometrically coincident cells.

queries are simply compositions of these basic operations. In other words, a *fixed* finite set of primitives H gives rise to a representational space $M(H, O)$ consisting of the transitive closure of the primitives under the selected operations O [78]. It turns out that this representational space is *finite* (because only finitely many semi-analytic sets may be constructed using a finite set of primitives) and can be completely characterized by a particular stratification of space determined by the primitives in H [102, 104].

Given a finite collection of primitives, H, consider the stratification of the whole Euclidean space E^3 such that every every stratum is a maximal connected k-manifold that is also sign-invariant[17] with respect to all primitives in H. Such a stratification exists for every collection of semi-analytic primitives, is unique, and is sometimes called *Whitney regular*. It has a reasonable low-degree polynomial size, and its strata satisfy the frontier conditions. Figure 3 shows some examples. The practical significance of this stratification lies in the fact that *every* representation (constructive and/or combinatorial) in the modeling space $M(H, O)$ is essentially an optimized union of some strata. For example, boundary representations consists of merged 0-, 1-, and 2-dimensional strata (vertices, edges, and faces respectively), CSG representations are regularized Boolean representations of unions of three-dimensional strata, and so on. Thus, the Whitney regular sign-invariant stratification of space serves as the lowest common denominator for all representation schemes and allows systematic development of algorithms and queries. See [102] for additional details.

4. Algorithms

4.1. Fundamental Computations

The unified view of solid representations allows to identify a small set of 'fundamental' operations: *primitive selection, stratification, point generation, point membership classification, set comparison, and ordering.* These operations are fundamental in the sense that most other algorithms can be defined through their composition. As with our classification of mathematical models, these operations are not entirely independent: for example, point membership may require stratification, and almost all operations require some form of comparison. Specific representation schemes are often optimized for several (but rarely all) of these fundamental operations. In practice, all fundamental operations except ordering can be implemented only approximately; see section 6.3 on standards and section 7.1 on unsolved problems for discussion of the ensuing difficulties.

Selection of geometric primitives

Before attempting any solid modeling computations, we must make sure that the set of known primitives and/or carriers is sufficient to represent the result of computation. Oversimplifying, the results of geometric computations are made up from portions of the candidate sets. In some cases, this is a trivial step: it is clear that in order to compute the intersection of a line with a solid's boundary, we must have a representation of the line and representations of the surfaces bounding the solid. It may be less obvious (but true) that the boundary representation of a solid may be constructed from boundaries of the corresponding CSG primitives. But in many other situations, the candidate primitives are not obvious, must be determined as part of the computation, and are usually not unique. For example, it is well known that carriers found

[17] In other words, every primitive function in H has the same sign on all points of the stratum.

in a typical boundary representation are insufficient for the CSG representation of the same solid [108], and it is far from obvious which geometric primitives are needed to describe the boundary of a blend or a sweep [128].

Stratification

Perhaps the most difficult of solid modeling computations is the process of identifying all cells in the Whitney regular sign-invariant stratification for a given set of primitives. The theoretical stratification process decomposes any semi-analytic set into a collection of connected smooth k-submanifolds by recursively extracting k-dimensional solid portions of the given set [64, 130, 131] (see Figure 3). But consider the simplest case of two primitives h_i and h_j. There are at most nine non-empty sign-invariant sets corresponding to the pairwise *intersections* of the sets defined by the signs of each primitive. Recall that intersection is also required for computing and representing the geometric carriers of lower-dimensional cells in a cell complex where only the geometry of the higher-dimensional cells is specified directly. In particular, boundary representations routinely require computation of the intersection curves between incident faces. Point classification against a boundary representation requires computing intersections of bounding surfaces with a line, and one of the most popular methods for visualizing implicit sets relies on ray-casting (intersecting the set with a grid of parallel lines). Furthermore, such intersection sets may be also disconnected, heterogeneous in dimension, or contain singularities and self-intersections. Thus, at the very least, stratification requires computing connected smooth submanifolds of the intersecting primitives satisfying the frontier condition, and it is not surprising that one of the chapters in this handbook (see the chapter on intersection problems by N. Patrikalakis and T. Maekawa) is devoted to the study of the intersection problems. While in theory any semi-analytic set may be stratified *exactly*, it appears that general and practical algorithms almost always resort to numerical methods.

Point Membership Classification (PMC)

We already discussed point membership queries in the context of individual representations schemes (constructive, combinatorial, and boundary). A more general PMC operates on an arbitrary point p and a representation of a set S, and returns *in*, *on*, or *out* depending on whether p is respectively in the interior, boundary, or outside of the solid S [118]. Since solidity, interior, and boundary are all topological concepts defined relative to some universal set X (for example E^3, surface, curve), it should not come as a surprise that PMC of a point p usually requires computing the neighborhood of p relative to X. Neighborhoods of smooth and regular points on ∂S are adequately represented by the tangent (or normal) information at p. Other points may require non-trivial analysis and computations depending on the representation of S.

Point Generation

Point generation is required to produce a single representative point from a set which may or may not have some special properties (such as center of mass), or to generate many such samples throughout the represented set with some (regular or irregular) intervals. Point generation is straightforward for sets represented parametrically or enumeratively, but may be difficult for other representations. For combinatorial representation, point generation is performed cell-by-cell serially or hierarchically, depending on how the cells are organized. For implicitly defined sets, points may be generated by sampling (and classifying usng PMC) with desired resolution; for constructive representations, points may be generated for each of the primitives and then

filtered using PMC test as described above.

Comparison

Comparison of two points represented by their coordinates is relatively straightforward. But if the numbers are not exactly the same – and they rarely are – one must compute the *distance* between the two points using some appropriately defined metric. The most common type of distance is the usual Euclidean distance, but other metrics, such L_p and Hausdorff, are often useful and necessary. Comparing sets of points is a substantially more difficult task that depends on how the sets are represented, and what kind of metric is used. Notice that distance computation is a perfectly valid method for answering PMC queries, and comparison is usually required in order to decide whether two sets are incident on each other. Because neither implicit nor parametric representations for a set of points are unique in general, set comparison usually requires point generation, classification, and point comparison. Comparisons of different combinatorial representations may be formulated in terms of comparisons of individual cells. Finally, comparison of two solids may require development of metrics that take into account not only geometric distance, but also their topological form [14].

Ordering

A process of combining the results of the primitive computations into the representation of the result is difficult to describe generically, because the ordering process itself depends on the way the representation is organized. For constructive representations, ordering produces a tree of constructions; ordering of groupings produces either serial or hierarchical structures; and ordering of cell complexes involves creating and maintaining incidence and adjacency information.

A particularly elegant approach to ordering 2-manifold boundary representations takes advantage of the familiar Euler characteristic χ which is defined as an alternating sum:

$$\chi = V - E + F, \tag{9}$$

where V, E, and F are the numbers of vertices, edges, and faces respectively in the boundary representation. If boundary representation is *known* to be a connected manifold with every k-cell homeomorphic to a k-dimensional disk, then an even number χ provides complete classification of all such surfaces up to a homeomorphism. For a fixed χ, equation (9) can be viewed as a two-dimensional linear subspace of the three dimensional space defined by coordinates V, E, and F; valid operations on the boundary representation may change the numbers of vertices, edges, and faces, but all resulting boundary representations must be confined to the same plane. Such valid operations are termed *Euler operators* (because they preserve the Euler characteristics) and can be viewed as vectors in the two dimensional linear subspace [55]. Because a two-dimensional linear space can be spanned by two linear independent vectors, only two (independent but not unique) Euler operators are required to build a boundary representation for any polyhedron homeomorphic to the sphere.

This counting principle can be extended to more complex boundary representations with more general cells and multiple connected components (shells) [16]. For example, simple counting arguments show that $\chi = R + 2(S - n)$ where R is the number of interior face rings (counted by internal loops on the faces), S is the number of connected shells, and n is the genus. Eliminating Euler characteristic, χ, from equation (9), we obtain a linear constraint

$$V - E + F - R - 2(S - n) = 0, \tag{10}$$

representing a five-dimensional hyperplane in the six-dimensional space and implying that at least five distinct Euler operators are necessary and sufficient to span the space of all such cell complexes. Euler operators conveniently enforce the required combinatorial conditions on such cell complexes,[18] but they are not unique, and other approaches to constructing and maintaining ordering are possible [28].

4.2. Enabling algorithms

All other geometric queries and algorithms in solid modeling may be constructed using sequences of the above fundamental computations. It would be impractical to describe here all algorithms important in solid modeling, but it is instructive to consider how some of the more common and critical computations can be cast in terms of the fundamental operations. Specifically, we focus on those algorithms that can be broadly classified as *representation conversions* because they tend to re-represent the solid in a manner that makes desired computations simpler; these algorithms shaped the solid modeling technology, enabled specific applications, and defined the architecture of the commercial systems. The following description is optimized for clarity; efficient algorithms involve essentially the same steps but are structured to take advantage of locality, proximity, coherence, symmetry, and hierarchy, as well as specific assumptions and known properties of particular geometric representations.

Ray casting

Ray-casting and ray-tracing are popular techniques for rendering solids and their boundaries [97], for representing a solid as a grouping of line segments, for performing PMC against a boundary representation of a solid, as well as for performing other types of analysis [27]. Such algorithms require computing the intersection of a given (possibly unbounded) line segment with a representation of the solid. For all representations, the ray-casting algorithm reduces to computing the intersection of the unbounded candidate line with each of the geometric primitives (leaves of a CSG tree or carriers of the highest dimensional cells) yielding an unsorted list of points along the line. Those points that classify *in* or *out* the solid are discarded, as well as those points that classify *out* with respect to the line segment. The remaining points are *on* the solid and bound one or more linear segments – the result of the intersection. These points are sorted along the line in order to induce in/out classification for the segments of the line they bound. The result must be regularized in the topology of the line (because we only want solid line segments), which may require constructing one dimensional neighborhoods of points and/or additional PMC tests. Ray casting is particularly popular with the CSG representations due to the elegant divide-and-conquer algorithm (similar to the merge sort) that merges ordered lists of intersection points for every interior node of the tree [118].

Sampling and polygonization

Many of the application algorithms require sampling and/or approximating the represented solid. These include rendering, path generation (for motion planning or machining), computation of integral properties, and finite element meshing. Broadly, all such algorithms produce a k-dimensional grouping or a simplified cell complex from a given solid's representation.

To generate points (0-dimensional grouping) on the solid's boundary, one generates the points on the boundaries of candidate primitives (geometric carriers in boundary representation or

[18] Note Euler operators do *not* guarantee the validity of the results unless additional geometric conditions are satisfied as well.

boundary of primitives in the CSG tree), and classifies these candidate points against the given representation of the solid. Similar processes may be used for generating points in the interior of the solid, except that the candidate points must be generated throughout the three-dimensional space, for example on the regularly-spaced grid throughout the space containing the solid. Ray-casting (see above) can be used to generate 1-dimensional groupings.

Two-dimensional groupings are usually made up from triangles and polygons constructed from points generated on the solid's boundary. The construction may require that the vertices of every edge and/or polygons must have the same classification with respect to solid's faces or edges, if such information is available. The constructed edges and polygons may be further ordered into a valid cell complex satisfying the usual combinatorial conditions.

Popular three-dimensional sampling include octrees and tetrahedralization. Octrees are constructed by recursively classifying orthogonal boxes with respect to the solid as *in*, *out*, and those intersecting the boundary of the solid. The latter are subdivided further until the desired level of resolution is reached. The box/solid classification is in turn reduced to classifying the vertices of the box, intersecting edges or faces of the box with the solid's boundary, and ordering of the result. The octree cells may be further subdivided into tetrahedra; tetrahedra may be also constructed directly from a 0-dimensional grouping of points sampled or generated in the interior of the solid, for example using the Delaunay constraint [101].

Set membership classification

This is a more general computation that subsumes PMC, ray-casting, cell sampling, and many other computations in the following sense [118]. Given representations of two sets: X is a *candidate* set, S is a *reference* set; both are solids but do not have to be of the same dimension. The SMC procedure $M(X, S)$ partitions the candidate set with respect to the reference set

$$M(X, S) =< X\mathrm{in}S,\ X\mathrm{on}S,\ X\mathrm{out}S > \tag{11}$$

into the three solid portions of X. In this context, solidity is defined with respect to X. When X is a single point, SMC reduces to PMC described under the fundamental computations; when X is a curve segment and S is a solid, SMC is the curved ray casting procedure outlined above. More generally, SMC subsumes many other geometric computations in solid modeling. For example, when both X and S are solids, $X\mathrm{in}S$ is their regularized intersection; when S is a two-dimensional face and X is a curve lying in the same surface, $X\mathrm{in}S$ is the portion of the curve contained in the region bounded by the face; and so on. SMC could be considered itself as one of the fundamental computations, except that it is not implemented directly but must be reduced to some sequence of the other fundamental computations, as illustrated by the examples above.

Boundary evaluation and merging

The queen of all representation conversion procedures – both in complexity and its importance in solid modeling – is the so called *boundary evaluation* procedure that produces a valid boundary representation of a solid given its constructive representation [89]. For CSG representations, the procedure conceptually is straightforward. If the solid S is represented by a sequence of regularized set operations on collection of primitives $\{h_1, h_2, \ldots, h_n\}$, then it is not difficult to show that

$$\partial S \subset (\partial h_1 \cup \partial h_2 \cup \ldots \cup \partial h_n) \tag{12}$$

In other words, the boundary representation of the same solid may be stitched together from the boundary pieces of the CSG primitives. Which pieces? Those pieces that classify *on* with respect to the given CSG representation; they are also called *faces* in the boundary representation [111]. In generic terms, the boundary evaluation reduces to performing $SMC(\partial h_i, S)$ for every primitive, and representing the union of the result. This in turn requires partitioning ∂h_i into the candidate pieces that may (or may not) lie on the solid's boundary. For efficiency, we want the pieces to be as big as possible, but they need to be small enough so that we do not miss any portion of ∂S. It can be shown that a sufficient (but not necessary) partition is obtained by intersecting ∂h_i with the boundaries of all other primitives in the given CSG representation. Each portion of ∂h_i bounded (or *trimmed*) by the intersection curves becomes a potential candidate face, and a single PMC test for any point in the interior of the candidate face is sufficient to determine if the face belongs to the boundary representation or not. Each face that passes the *on* test must be represented in the resulting boundary representation, typically (but not necessarily) by *its* boundary which consists of the *segments* (i.e., connected 1-dimensional manifold strata) of the intersection curves, called *edges*. A sufficient set of candidate edges is obtained by intersecting each intersection curve with all other surfaces, but only those with points classifying *on* with respect to the face belong to the boundary representation.

Thus, a typical boundary evaluation algorithm involves computing intersection curves between the primitive surfaces, computing intersection between the curves and the surfaces, generating points in the tentative curves and faces, followed by PMC testing these points with respect to solids and faces, and ordering the passing edges and faces into a valid cell complex. Consider now a very special case when there are only two solids h_1 and h_2 – both given by their boundary representations – and combined using either regulalized union $\cup^*$ or regularized intersection $\cap^*$ operation. Following the steps in the above procedure, we would have to compute intersection between the two sets $\partial h_1 \cap \partial h_2$, which involves trimming the faces and edges in the two boundary representations against each other, and merge the resulting pieces into a new boundary representation. This special but important case of boundary evaluation is often called boundary *merging*.

The conceptual structure of the CSG-to-boundary evaluation procedure provides a recipe for all other types of boundary evaluations. For example, suppose one wants to perform boundary evaluation for a constructive representation containing the sweep operation defined by (7). The general procedure involves exactly the same steps as before: generate a sufficient set of candidate surfaces, trim the surfaces against each other to produce a set of candidate faces and edges, generate a point in each candidate cell, perform PMC against the sweep representation, and order the results into a cell complex. It is intuitive and can be shown formally that

$$\partial(sweep(S, M)) \subset sweep(\partial S, M), \tag{13}$$

which means that the candidate surfaces must be obtained by sweeping the edges and faces in the boundary of the given solid. This could be challenging for general boundary representations and motions, and various approximations for such sweep surfaces have been proposed using sampling and polygonization techniques. Boundary evaluation for other constructive representations requires similar sequences of fundamental computations.[19] Practical implementations are usually optimized to generate the smallest possible number of candidate faces and edges,

[19]This assumes that a construction is properly defined and supports an unambiguous point membership test; unfortunately, such definitions may not always be available even for very common constructions, such as blending

never repeat the same computations, and employ coarse spatial tests to localize computations whenever possible.

Other representation conversions

It can be shown that all other computations in this category can be designed systematically using sequences of the same fundamental steps [104]. For example, CSG representations may be computed from a given boundary representation roughly as follows: select the sufficient set of primitives; intersect all primitives to produce the usual stratification of the whole space; generate at least one point in every sign-invariant three-dimensional component in the stratification; classify the generated points (and therefore the corresponding components) against the given boundary representation; the regularized union of the components classifying *in* correspond to the disjunctive canonical 'sum-of-products' CSG representation which can be further optimized using Boolean optimization techniques. The last step relies on repeated point membership tests and comparison. The most difficult step in this procedure is the selection of a sufficient set of primitives, which has been solved for restricted but common types of boundary representations. More details on boundary to CSG conversion may be found in [107, 108].

Other representation maintenance utilities – from comparing and optimizing constructive representations and approximating solids by groupings, to computing wireframes, silhouettes, and meshes – all may be systematically designed following similar procedures composed from the same fundamental computations [104].

5. Applications

An inexhaustible variety of applications may be developed with the help of fundamental computations and enabling algorithms. Below we briefly survey those popular engineering applications that helped to shape solid modeling as we know it today.

5.1. Geometric design

Historically, the activity of geometric design is largely devoid of physical analysis, but seeks efficient methods for creating, modifying, visualizing, and annotating geometric representations of solid shapes. By definition, constructive representations are well suited to constructing and modifying the models using high-level engineering parameters: distances, angles, radii, coordinate systems, etc – all can be encoded as parameters of the constructions. The resulting solid shape (although implicit) may be controlled by modifying these parameters. At the same time, constructive representations provide *no* explicit information about the boundaries, which makes them difficult to visualize, modify locally, or annotate. By contrast, combinatorial representations, and boundary representation in particular, are ideal for visualization and annotation tasks that require direct access and traversal of a solid's boundary. But because combinatorial representations are difficult to construct and manipulate directly, they appear to be ill-suited for design activities.

To facilitate subsequent editing of solid models (a critical issue, since the vast majority of designs are in fact modifications of earlier designs), both constructive and combinatorial representations are often supplemented by constraints, expressed as equations and inequalities on the parameters in the constructive representations and/or on the coordinates of the carriers on the combinatorial representations [51, 91, 113]. Typical constraints include tangency, incidence, perpendicularity, distances, angles, and so on. More recent approaches apply such constraints

directly on the cells in a stratification underlying the constructive parametric model [9]. To effect a desired modification, the user typically changes the values of constraints and/or some parameters; these changes are reflected in the updated systems of equations that are solved for the new values of all other parameters and coordinates, leading to a new instance of the solid model [67]. See the chapter on parametric modeling by Christoph Hoffmann and Robert Joan-Arinyo for more details.

5.2. Analysis and simulation

Most of the solid modeling computations in this category may be abstracted by single-valued functions of one or more solids (and possibly other variables) and have recognizably correct answers. Some of the more popular applications include rendering, computation of integral properties [48], assembly modeling, interference detection, simulation of mechanisms, and NC (numerical control) machining simulation.

Rendering and computation of *mass properties* are in fact quite similar in the sense that both require some form of finite enumeration of the points in – or rather the cells in a grouping associated with – the solid's interior or its boundary. In the case of integral property computations (which include volume, surface, inertial properties, and other integrals of functions defined over the solid's interior or boundary), the contribution of each individual cell is added to the result. In the case of rendering, the contribution of every cell is displayed on the screen. Two main principles used for such computations are *sampling* and *dimensional reduction.* Dimensional reduction relies on the generalized Stokes theorem to reformulate the computation over solid S in terms of another computation over the boundary ∂S [54]. Thus, the volume integral over S may be computed directly or reformulated as a surface integral over ∂S, and the integrals over individual faces may sometimes be reformulated in terms of the path integrals over the edges bounding the face. Similarly, rendering may be performed by visualizing solid cells drawn on the screen in the depth-first order, boundary cells using the surface normal information, or by drawing the edges and silhouette curves generated from the boundary representation. Sooner or later, the computation reduces to evaluating some function over a relatively simple constituent cell: a line or curve segment, a triangle, a polygon, a tetrahedron, a cube, etc. Evaluation of polynomial functions over linear polyhedral cells may be performed exactly (within the machine precision) [49], but for more general functions and/or cells evaluation is performed only approximately based on the function's values at carefully chosen (e.g. Gauss) points in the cell.

Assembly is a collection of solids that are positioned and oriented by some some rigid motions in space, subject to mating and non-interference conditions [46]. At the very least, the mating conditions identify pairs of contacting surfaces and thus require explicit boundary information. Non-interference between two solids A and B requires that their regularized intersection $A \cap^* B$ is empty (the solids interiors are disjoint); empty non-regularized set intersection $A \cap B$ implies that there is no contact between the boundaries ∂A and ∂B. Non-interference conditions may be defined and computed with any unambiguous representation, but recall that deciding 'emptiness' of constructive representation is a non-trivial matter.

A *moving solid* is easily represented by applying the rigid motion to the coordinate system in which the solid is designed. A *mechanism* with two solids A and B is an assembly where A moves *relative* to (the coordinate system of) B. Thus the *static* mating and non-interference conditions must be enforced at all times, resulting in more complex *dynamic* conditions. Dynamic non-interference between two moving solids A and B may be formulated as a static

non-interference between stationary B and $sweep(A, M)$, where M is the motion of A relative to B. The dual operation of $unsweep(A, M)$ can be used to formulate and compute queries about largest/smallest non-interfering objects [39]. The continuous motion may be also simulated discretely by evaluating the static solid configuration in small time increments. See [52] for a recent survey. Maintaining proper assembly conditions at all time steps may require substantial computing resources.

The simple pairs of solids may be chained together into graph structures to model *mechanism linkages,* such as robot arms. The mechanism motion is instantiated by combining the individual relative motions according to the graph by a procedure called *forward kinematics,* which involves multiplication of the matrices representing the individual relative motions. The inverse kinematics algorithms require solving the systems of non-linear equations to determine the individual relative motions, given the motion of some point or coordinate system on the mechanism. Mechanism modeling may be viewed as a natural extension of the constructive representation that allows using continuous motions (as opposed to instances of motions used to position a solid in space) [121].

Simulating *numerically controlled (NC) machining* is an application similar to mechanism modeling: solid cutter A is moving relative to solid stock B which is fixed in some solid fixture C. Motion of the cutter A is determined by the NC program, and the purpose of the simulation is to determine whether the moving cutter A removes the desired amount of material from B without interference with the fixture C. The material removal operation is modeled by the regularized difference operation: at every time step, the material removed by the solid cutter A is subtracted from the stock B. A more accurate approximation of the NC machining process requires computing $sweep(A, M)$, estimating volume removal rates, and other integral properties. See [60] for discussion of these and related issues.

5.3. Dynamic analysis and lumped-parameter systems

Simulation of a rigid body motion under the externally applied forces and moments requires computation of a solid's mass properties (mass, moments of inertia, center of mass), solving for the acceleration of the solid at that time instant, and integrating it through time to modify the solid's velocity. This in effect approximates the solution of the ordinary differential equation of solid's motion. Repeating this computation at small discrete time intervals produces realistic simulation of motion. A more sophisticated simulation applies the same procedure to the *system* of rigid bodies interconnected at joints, solving the constrained system of ordinary differential equations [22]. In this case, the physical object under consideration is better represented combinatorially as a *graph* of solids, whose links specify the types of the motions allowed at the joints, and no interference is allowed between the individual solids at any time during motion. The latter task requires dynamically tracking the distances and identifying the collisions between all moving solids, which can be handled, albeit at a high computational cost, by the standard solid modeling methods [52]. However, the contact geometry of joints in such a mechanism structure is usually *assumed;* computing it would require full power of solid modeling discussed above plus adequate models of contact and friction mechanics [7].

The next logical step is to enhance the simulation model with a proper model of impact mechanics, and predict the motion after an impact takes place. In addition to the need to identify precisely the time and type of all possible contacts, it is apparent that the contacting solids usually constitute a non-manifold geometric model, and can be also considered as a mechanism

with a "contact" joint at the time of impact. Commercial implementations of such simulations are already available, but they are all limited by the lack of good models of impact mechanics – that must include mechanical deformations and/or experimentally-determined coefficients of restitution.

A one-dimensional graph structure of interacting solids can be employed with more general systems of ordinary differential equations arising in other branches of physics. Such graph structures have been known for a long time [71] and are now used by commercial software systems. The graphs are composed from nodes that represent 'lumped'[20] constitutive properties, such as masses, inertia, dampers, resistances, inductances, spring constants, and so on, plus a finite number of transducers that convert and couple different physical models in a single structure. In this context, spatial information serves three purposes: computation of the lumped properties associated with each solid, visualizing the response of the system in terms of the solids, and managing the spatial incidence and adjacency of the lumped elements based on interaction between the solids.

5.4. Planning and Generation

In contrast to analysis and simulation applications, planning and generation tasks do not typically have a unique answer but must produce one or more acceptable solutions from some usually large spaces of feasible answers.[21] An application in this category is usually constructed as a heuristic search procedure based on repeated querying of known solid models – typically using the fundamental operations described in section 4.1.

Motion planning requires finding a collision-free path for a moving object with respect to one or more solids [45]. In addition to the numerous popular robot motion planning tasks, other applications in this category include generation of tool paths for NC machining and automatic inspection plans by coordinate measurement machines. *Feature recognition* is a process of matching the portions of a represented shape to previously known parameterized forms, shapes, or processes, e.g., for the purpose of constructing a manufacturing process plan or identifying parts for inspection purposes [41]. The vast majority of such features appear to be defined by relationships between portions of a solid's boundary: faces, edges, special points, incidence, convexity, symmetry, specific sizes, and so on. Another important example of an application in this category is *mesh generation,* which is a process of constructing a cell complex representation for a given solid (called a mesh), satisfying specific requirements on the size, shape, number, and topology of cells. These requirements are defined by some analysis application that seeks to approximate the answer to a boundary-value problem using a particular type of spatial discretization of the domain (finite element, finite differences, hexahedral, simplicial, etc.) All meshing procedures require generating points in the solid's interior and boundary and ordering them to form a complex that has the same topology as, and conforms geometrically (as closely as possible) to the solid's boundary [68].

It has been clear from the outset that the guaranteed validity of solid models also holds a great promise for *automatic design* of engineering artifacts [125]. The constructive representations appear to be particularly well suited for the task, if a way can be found to relate the construction

[20] 'Integrated' may be a more proper way to described 'lumped' [106].

[21] The two types of problems are not entirely independent of course; since many of the analysis and simulation computations are approximate, the 'correct' answers are not unique, and in fact may depend on the more difficult planning and generation tasks.

parameters to the desired design and/or functional characteristics. The constructive representation may then be viewed as a procedural definition – a computer program that implements the design algorithm and outputs a valid solid-represented design. Indeed a number of languages and grammars for generating such solid-represented designs have been proposed, but all faced challenges because evaluation of the generated solids required not only explicit geometric and combinatorial information, but also physical and functional information that is normally not present in a geometric modeling system [106].

5.5. Manufacturing

Effective application of solid modeling techniques is possible for many traditional manufacturing processes that are idealized as *unit processes* [123] with well-characterized input and output geometry. Simulation, planning, and verification of unit processes, particularly machining and assembly, were some of the main catalysts for the development of solid modeling [86]. More recently, the range of supported applications has been greatly expanded to include sheet metal manufacturing, injection molding, stamping, pipe routing, and so on. Such applications are supported within the solid modeling framework through specialized user interfaces that force designers to create models in terms of a limited application-specific lexicon. For example, machined parts would be designed in terms of pockets, holes, and other machining features; sheet metal parts would be designed by sequences of bending and stamping operations; draft angles are automatically added to injection-molded parts; and so on. Usually, such operations are translated into application-specific constructive representations that are then evaluated into combinatorial (usually boundary) representations. In addition, heuristic knowledge-based systems are emerging that are capable of identifying common geometric features that may (or may not) be supported by a particular manufacturing process.

Beyond traditional manufacturing, if every solid can be approximately represented on a computer by a grouping of some k-dimensional cells, why not manufacture it by a computer-controlled process that will grow the solid by depositing the individual k-cells? This basic idea underlies a host of new manufacturing techniques that are often referred to as layered-manufacturing (because they build the solid layer by layer), solid imaging or printing (because they employ deposition processes that are reminiscent of the paper printing methods), or solid free-form fabrication (because the cell-by-cell deposition process removes many of the manufacturability constraints on the solid's shape [56]. For a layered process to work correctly, it must work with valid solid representations accepted from different sources. A *defacto* standard representation for this purpose is called STL: a boundary representation constructed as a collection (a 2-grouping) of triangular oriented 2-cells. The main virtue of this representation is its simplicity; but as with most groupings, STL representation is only approximate, lacks information on incidence between the cells, is sensitive to round-off and precision errors, and as such, is quite unreliable.

The ability to rapidly manufacture solids of arbitrary shape without specialized tooling naturally leads to an attractive idea of object copying. This involves sampling points on an existing object and constructing its solid representation (a process often called *reverse engineering*), followed by a layered reproduction process [124].[22] Detailed discussion of reverse engineering can be found in the chapter by Martin of this handbook. Briefly, reverse engineering is a process

[22] A more general notion of reverse engineering should include to reproducing the functionality, as well as the shape, of the artifacts.

of constructing a solid's representation from an unorganized cloud of points that are sampled on the boundary of an existing physical object with some accuracy. In terms of fundamental solid modeling operations, the process involves comparing and ordering the points into strata of dimensions 0 (points), 1 (edges), and 2(faces) whose union represents the boundary of the object being reconstructed.

6. Systems

6.1. Classical Systems

We saw that each constructive and combinatorial approach to solid modeling offers specific practical advantages when it comes to implementing particular engineering applications. Theoretically, the two approaches are equivalent, because both are informationally complete with respect to the postulated mathematical models, and therefore are capable of supporting all of the fundamental and enabling computations discussed above. But the constructive representations are parameterized, concise, and robust representations that come with a built-in point-membership test and are a natural choice for creating, editing, and programming solid models; they are a poor choice for applications requiring point generation, boundary traversal, and persistent spatial addressing. In contrast, the combinatorial representations provide explicit, persistent, and spatially addressable enumeration of the solid's boundary (and possibly interior) that is perfectly suited for applications requiring point generation and sampling; they also appear to have a greater geometric coverage, but combinatorial representations are verbose, more sensitive to errors and inconsistencies, and are difficult to create and manipulate directly.

The early solid modeling systems were designed around single carefully chosen representation schemes, usually either the CSG or the boundary representation, but by the mid-1980's virtually all single representation systems were enhanced and extended with auxiliary representation and features, making them in fact hybrid *dual-representation* systems [87, 88]. The emerging architecture for such a dual-representation solid modeling systems is shown in Figure 8(a). The constructive representation facilitates model creation and editing, often in terms of application specific constructions that reflect individual user's concepts and mentality, while the associated combinatorial representation supports other applications requiring generation, traversal, and spatial referencing. Computations may be performed against either representation. The combinatorial representation usually takes the form of a boundary representation, an approximate polygonal model, or an octree that is produced automatically from the constructive representation. Note that the representation conversion is one way: always from the constructive representation to the combinatorial one – partly because the inverse conversion problem is more difficult, but mostly because the conversion process does not usually capture the constructive semantics of particular applications. This implies that the combinatorial representations should not be allowed to be edited or modified directly, because this could lead to inconsistency between the two representations.

6.2. Parametric Interaction

But by the late 80's, parametric modeling acquired one new and decisively critical ingredient: the constructions and constraints were applied interactively and incrementally by making direct references to the *previously constructed* cellular representation of geometry [21, 109]. The resulting construction method was strongly reminiscent of the datum-based dimensioning system used by engineers, and the resulting graphical user interface resonated well with design-

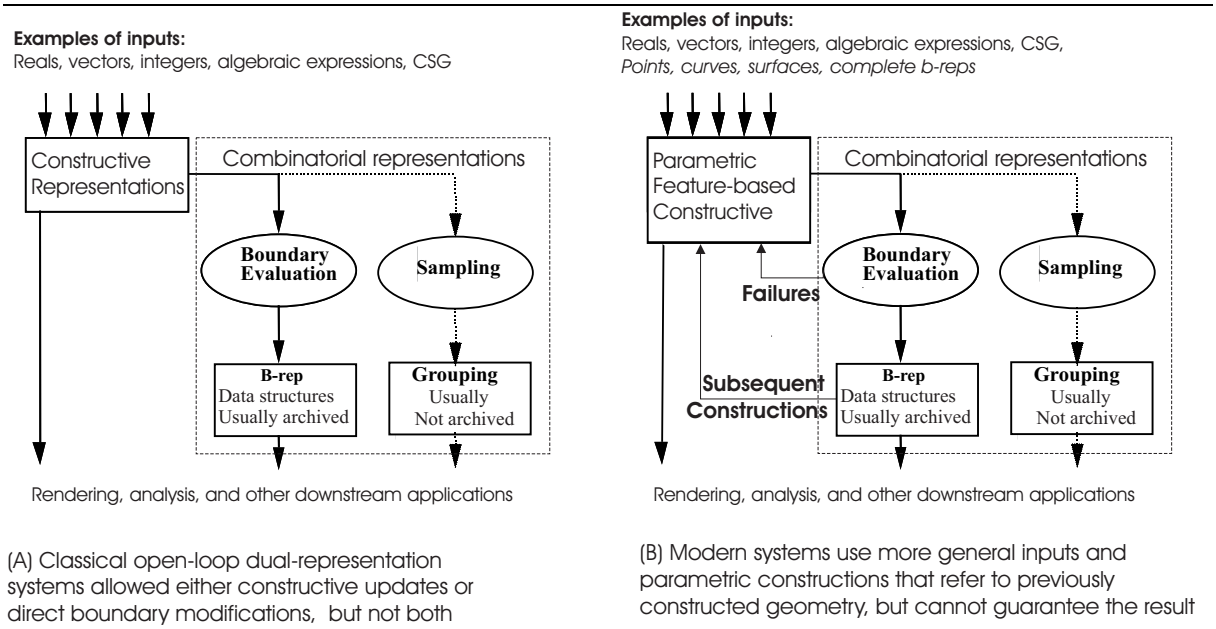


Figure 8. The architecture of the classical dual-representation solid modeling systems is defined and limited by the representation conversion technology; the architecture of the new parametric systems is shown side-by-side.

ers whose primary job was to produce engineering drawings fast and without mistakes.

The new parametric solid modeling systems have a complex multi-layered architecture (shown schematically in Figure 8(b)) that combines constructive and combinatorial representations with constraint-solving and heuristic algorithms [32, 109]. As in the earlier systems, direct modifications of cells in the boundary representation is not allowed in this architecture, because this may lead to a loss of consistency with the corresponding construction; however, evaluation of every parametric edit does modify the cells in the boundary representation thereby affecting all future constructions that refer to this cell. Recall also that all combinatorial representations (including the boundary representations) support persistent spatial addressing, which means that every cell in the representation has a unique name that identifies the set of points associated with this cell. By contrast, the constructive representations define sets of points that may be indistinguishable from each other, as is the case with the connected components of a disconnected set defined by intersection of simpler sets.[23] When a solid modeling construction refers to a particular cell in the boundary representation (for example, a new hole is positioned with respect to an existing reference face), it assumes that the cell is persistent. But the cell itself was evaluated from an earlier construction, and may have a different name should the boundary model be regenerated at a different time with modified parameters or conditions. If the name of the reference cell changes, all future constructions change their semantics, resulting in a drastically different and

[23]The underlying mathematical problem is much more difficult that it may appear: ordering connected components of an implicitly semi-analytic set without computing its boundary or combinatorics is as difficult as enumerating the multiple roots for a system of equations before it is solved[104].

unpredictable behavior(see example in Figure 9). The problem of assigning unique names to the cells in the boundary representation as they are generated from a parametric definition came to be known as *persistent naming*. It has been characterized formally and solved under certain conditions in [76]; several heuristic approaches have also been proposed [19, 42], but no general solution is known at this time. Thus, it should come as no surprise that the architecture of parametric modeling system as illustrated in Figure 8(b) allows for the failed operation feedback loop. The parametric solid modeling systems do produce solid models, but not all constructions have well-defined semantics or are guaranteed to succeed, and however solid the results may be, they are often not predictable or repeatable.

But let us suppose for a moment that we completely solve the two difficult problems of persistent naming and of the representation conversion. Would that eliminate the apparent difficulties in the parametric modeling? Hardly. Consider the solid model generated in response to the parametric change shown in Figure 6.2. Is the solid produced by the system correct? Maybe or maybe not, depending on your *definition* of correctness. In retrospect, it should not be surprising that parametric modeling does not guarantee the properties of the results beyond solidity. The classical mathematical models assumed in the scenario of Figure 1 correspond to instances, and do not reflect the parametric nature of the modeling systems.

6.3. Standards and Interfaces

Different users and applications often rely on different systems and representation schemes. The common approaches to the critical problem of data exchange and transfer include native translation, neutral standard file format, and standardized programming interfaces. The most general and ubiquitous approach is the neutral format because it does not require knowing *a priori* which systems are involved and does not lead to proliferation of special-purpose translators. Such formats were designed for the common CSG and boundary representations in early 80's and were later adopted as part of the international standards for product data description [66]. Work is currently underway to extend such formats for representation of parametric representations by including specification of common parametric constraints and constructions. This should allow systems to exchange representation of a nominal solid and the syntax for an associated parametric family, but cannot guarantee that this family is valid or is the same in all systems (see discussion above) [74].

Most commercial solid modeling systems provide an application programming interface (API) that allows to access and compute with models in a given system. Standardizing on such an API is another way to alleviate the problem of data exchange. A number of limited proposals for such standard APIs have been made recently as surveyed in [74]. Since the stratification of a solid model into cells provides a common theoretical framework for unifying all representations, it also provides a basis for designing a representation-free API that is both formal and general. The initial effort towards that goal is described in [6].

All existing approaches to data exchange and standardization are further undermined by the accuracy and robustness problems in solid modeling. Recall that the fundamental operations, in particular stratification and point membership classification, cannot be computed exactly and must be approximated in *all* systems. Different systems make widely varying and often incompatible assumptions about accuracy and precision of these and other approximations, at times making data exchange and standardization difficult or even impossible.

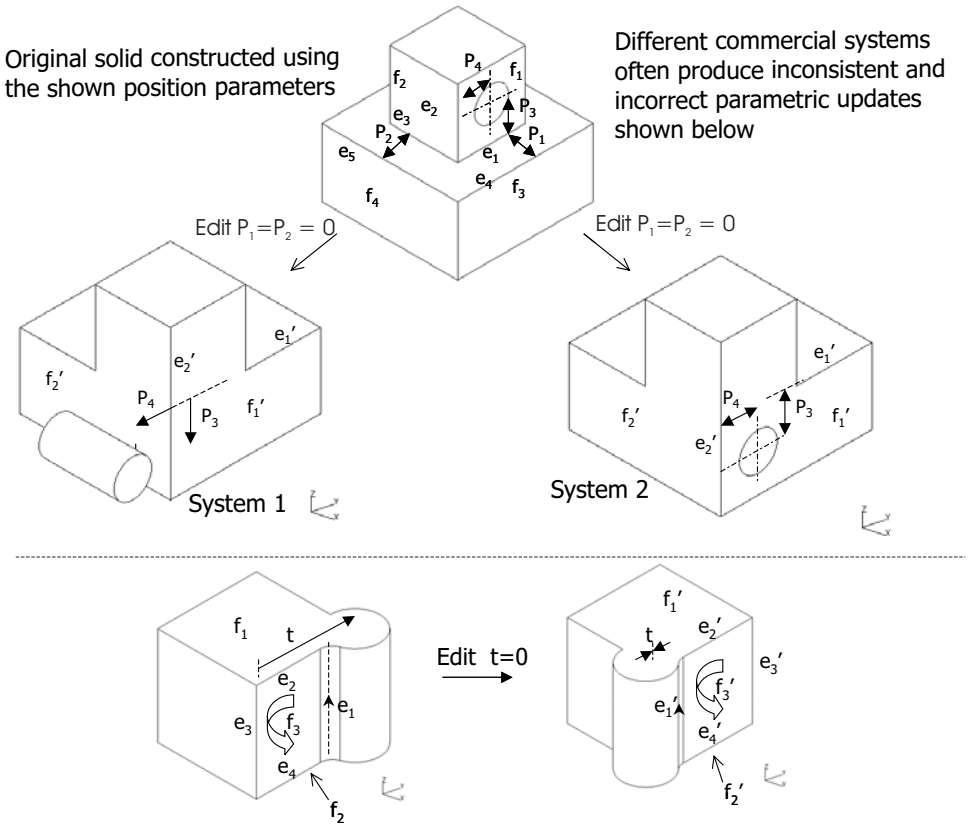


Figure 9. In the absence of standard mathematical models, parametric solid modeling systems often produce incorrect or inconsistent results.

7. Conclusions

7.1. Unsolved Problems and Promising Directions

Continuing improvements in computer representations, algorithms, and general computing technology have led to mature industrial strength implementation of the modeling scenario shown in Figure 1. It is safe to predict that the improving data structures for increasingly complex geometric shapes, relatively efficient and tested algorithms for fundamental and enabling computations taking advantage of the latest techniques in hardware, and the increasing computing power will result in further evolutionary progress in specialized situations and the ever-growing array of engineering and consumer applications. But these advances alone are not likely to solve the major outstanding problems in solid modeling, because they require substantial revision of the fundamental premises and the assumed mathematical models. A number of such issues and promising directions are discussed below in a logical order that does not necessarily correspond to their importance or priority.

Robustness of geometric modeling computations and systems has remained a challenge, despite numerous advances in accuracy and consistency (see [24] for a sample of recent work). The fundamental issue is that the theory of geometric and solid modeling is based on the classical model of exact geometry, but engineering data and computations are almost always approximate. Strictly speaking, this means that no theorem of exact geometry can be assumed to hold in the approximate model, and it must be proved again in the presence of errors. It is not enough to compute the results very accurately and consider all special cases, because introduction of *any* errors may lead to inconsistent results and contradictions. An ultimate solution of the robustness problem requires recognizing the imprecise nature of the mathematical models and reformulation of the key concepts and computations. For example, what is the meaning of the approximate stratification (or intersection) of imprecise primitives and how is it related to the approximate point membership test?

Persistent naming of cells in a combinatorial representation in terms of the primitives and operations in the corresponding constructive representation is required in order to support regeneration, editing, and exchange of parametric models. Mathematically, the problem reduces to the difficult problem of indexing the connected components of an implicitly represented set. The most promising approach is to devise a new class of constructive representations, where every such component is named persistently by construction, for example using interactive datums and references.

Parametric families of solids is a widely accepted, but poorly understood, notion. Informally, since we are dealing with families of physical objects, it is reasonable to expect that small changes in the values of parameters of a given solid should result in another solid that is similar or is "near" the original solid. This should in principle eliminate the jumps, sudden changes, and other unpredictable behaviors observable in current systems. Formally, this assumption corresponds to the notion of *continuity* of the mapping from the parameter space into the space of subsets of E^3 and/or their representations [76]. And therein lies the major difficulty of parametric modeling: for a given representation scheme, there is more than one way to define the notion of continuity, and furthermore, different solid representations schemes imply different parametric families. Specifically, there are several distinct methods for defining continuous parametric families based on a boundary representation [76], and they vary widely in their computational properties.[24] As expected, CSG and boundary representations of the same solid normally imply very different families [109] that may be related using tools from category theory [75]. Since most parametric systems rely simultaneously on the constructive and the boundary representations, generating and classifying solids in the combined family, as well as maintaining consistency between the distinct parametric families has emerged as a major technical challenge [35, 77] that must be met if there is any hope for computer interpretation of parametric models, such as those required in shape optimization or for standardization of parametric definitions.

Tolerancing and Metrology includes issues of accuracy, variability, measurement, and reconstruction, that are important in all manufacturing applications. One of the pillars of modern mass manufacturing, the doctrine of *interchangeability* of mechanical parts in an assembly,

[24]The main technical task is that of deciding whether or not two boundary representations belong to the same continuous family requires constructing explicit maps that may be difficult or impossible with the existing data structures, because they were designed to model solid instances and do not always properly record changes in orientation and dimension [76].

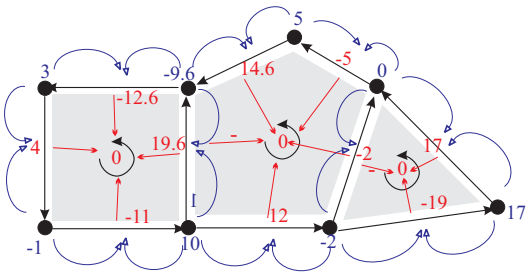
calls for precise geometric specification of when a part may replace another part in an assembly without affecting the functionality [36]. This suggests strongly that proper models for mechanical parts must include the notion of mechanical variation or *tolerance* and the procedure for deciding when two such parts are interchangeable. A simple minded approach to mechanical tolerancing is to endow some or all parameters of solid's representation with variations in $\pm\epsilon$ range. While it is easy to implement, this approach suffers from several severe limitations, including restriction to perfect-form and ambiguous semantics. A *geometric dimensioning and tolerancing* (GD&T) approach that is more consistent with modern mechanical engineering practices is based on the notions of datums and tolerance zones and gives the basis for various national and international standards. Several researchers proposed mathematical extensions of solid modeling theory to accommodate GD&T [14, 82], it is now clear that the formal semantics of the national standards are not completely defined. Despite massive efforts to mathematize the standards [115], it will be some time before a standard approach to computational modeling of tolerances can be defined [126], because this may require fundamentally reformulating the notions of variational control [116]. In the absence of a common mathematical standard, tolerances are being treated syntactically as attributes that are attached as labels to solid's features and surfaces. Nominal solid models can be used to plan sampling of the points on the surface of the corresponding actual parts, and numerous commercial packages will numerically fit the sampled data to curves and surfaces (typically using proprietary algorithms), but no such fitting algorithms are standard today.

Interoperability refers to the ability of solid modeling systems to exchange and compute on the models constructed by other systems or applications[40]. It plays the role of virtual interchangeability for computer models of mechanical parts. As such, full interoperability subsumes the issues of standardization, representation conversions, robustness, and well-defined semantics for parametric models and tolerances. Furthermore, it is clear that the very notion of interoperability depends on the application and the type of queries. Two solid models may be considered equivalent for the purpose of space packaging studies and point membership testing, but may not be interchangeable when it comes to performing engineering analysis, parametric studies, or manufacturing process planning. Systematic formulation and solution of such problems is a critical and fruitful area of research.

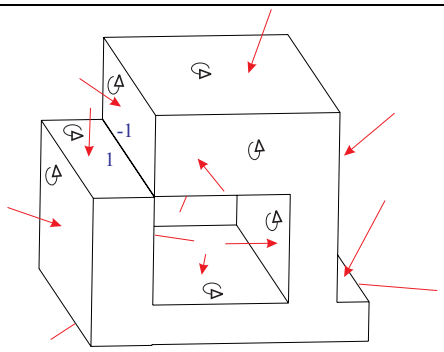
Physical field modeling is a natural generalization of solid modeling aimed at modeling spatially and temporally distributed physical properties. In a nutshell, the problem amounts to developing methods for representing, constructing, and manipulating discrete and continuous variations of physical quantities defined over a given geometric domain, subject to the postulated physical laws. For example, classical solid models presume material homogeneity, but new modeling techniques are needed to support design, analysis, and manufacturing of objects with materials that vary heterogeneously and anisotropically, resulting in products with superior structural properties, compliant mechanisms, and embedded sensors [72]. Similarly, engineering analysis requires representing and computing assumed physical properties (displacement, energy, temperature, stress, flux, etc.) over the solid's interior and boundary. The predominant approach to such problems requires conforming approximation of a given solid by a particular type of spatial discretization that supports numerical computations for the problem at hand. Finite-element and finite-difference meshing have emerged as a major research area and a substantial bottleneck to advances in engineering analysis [68]. From a practical point of view, such representation conversions are computationally intensive and numerically sensitive procedures

that are difficult to automate; proliferation of different and often incompatible techniques further undermines the standardization, robustness, integration, and interoperability efforts described above.

More fundamentally, modeling of physical fields should generalize and subsume continuum and combinatorial solid modeling, as well as constructive and combinatorial representations. A suitable continuum generalization of an r-set is based on the concept of a *fiber bundle* [43, 135] with the solid model playing the role of the base space and the fiber space corresponding to the field defined over the solid. The corresponding generalization of the combinatorial model may be defined using algebraic topological *cochains* over finite cell complexes [69, 122]. Just as the notions of boundary and cycles are needed to define valid boundary representations, the dual notions of coboundary and cocycles formally capture the combinatorial physical balance laws that must be satisfied by all valid field models (see Figure 10). It is reasonable to expect that the two models of the field problems – the continuum using fiber bundles and the combinatorial using cochains, provide different characterizations of the field that will lead to computationally complementary representations. For example, the fiber bundle model naturally leads to new methods for field description and analysis that are are not limited by the traditional mesh-based methods and difficulties [99, 105, 135], while the cochain models of physical distributions have been instrumental in developing new geometric languages for describing physical phenomena [20] and improved numerical techniques that preserve the indicated physical laws [57].



Coboundary operation on k-cochain transfers the coefficients from every k-cell to all incident (k+1)-cells with +/- sign depending on relative orientation. Addition of coefficients yields the total quantities that enter the (k+1) cells through their boundary. Repeating the coboundary operation produces a (k+2)-chain with all 0-coefficients.



A k-cochain associates a coefficient with every k-cell to represent a distribution of a physical quantity over a cell complex – in this case, a 2-cochain for a vector quantity distributed over the boundary of the solid. The cochain is a cocycle if its coboundary is 0, corresponding to the model of physical equilibrium with respect to the physical quantity.

Figure 10. Cochains, coboundaries, and cocycles as combinatorial models for physical fields. Compare to Figure 4.

Applications and Systems will continue to get faster, more robust, and sophisticated, but the breakthroughs in solid modeling technology require solving many of the above problems and introducing new mathematical models and paradigms. Thus, application of solid modeling to engineering problems involving complex physics, deformation, and phase changes (for

example, those arising in etching, micromachining, compression molding, metal casting, and so on) requires either complete geometric characterization of the controlled transformations, or merging solid and physical field modeling techniques. By contrast, the current trend towards function- and criterion-driven design requires rethinking the role of geometric models in synthesis of mechanical artifacts, which over centuries has been reduced to catalogue search and *posteriori* numerical analysis. Each engineering function usually influences only *some* of the design shape, while the same geometry usually serves multiple functions [38]. Somewhat paradoxically these observations suggest that synthesis tools may be more effective with *partial* geometric information at different stages; this in turn demands updating the classical notions of informational completeness, validity, and membership classification in solid modeling to include more general physical and process characteristics.

7.2. Summary

Solid modeling was conceived as a universal informationally complete geometric language for describing physical artifacts in support of industrial automation. It has had a dramatic effect on those areas where the classical notions are sufficient. The evolutionary progress in solid modeling during the last decade led to dramatic improvements in speed, reliability, domain coverage, and widespread use of commercial solid modeling systems.

At the same time, solid modeling grew and expanded to the point where it is no longer adequately supported by the original theoretical foundations. Many emerging applications cannot guarantee the correctness of results, because they are often based on heuristic and/or idealized geometric analysis of the physical problems. Since the guarantee is not always possible or is too expensive to compute, heuristic and sometimes wrong answers seem to be tolerated, as long as they can be generated quickly and checked by a human user or another system for correctness and then used for the downstream applications, such as producing annotated engineering drawings, manufacturing process planning, or engineering analysis. Validity and guarantee appear to have been replaced by an effective and iterative paradigm that demands speed and interactivity, while supporting and encouraging incremental improvements in the solid modeling technology.

Taking a longer term view of the field, further progress in solid modeling requires development of new mathematical models to support computer representations of increasingly complex physical artifacts. A pragmatic approach is to assume that such models may be application dependent. In particular, design, analysis and manufacture of engineering systems is driven by manufacturing processes, physical criteria, variability and incomplete information. This suggests that the corresponding notion of the “informational completeness” of such solid models must be modified to recognize that these and other attributes are at least as important as the associated nominal geometry.

Acknowledgments

This work was supported in part by the National Science Foundation grants DMI-9502728, DMI-9900171, DMI-0115133, and CCR-0112758. I am grateful to Alberto Paoluzzi for numerous stimulating discussions over the course of writing this survey, to Jeff Chard, Horea Ilies, Earlin Lutz, Malcolm Sabin, Nick Sapidis and Herb Voelcker for careful reading and criticism of the preliminary versions of this paper, and to Myung-Soo Kim for his patience and encouragement.

REFERENCES

1. *ACM Symposium on Solid Modeling and Applications*. ACM Press, 1991, 1993, 1995, 1997, 1999, 2001.
2. K. Abdel-Malek, D. Blackmore, and K. Joy. Swept volumes: Foundations, perspectives, and applications. *International Journal of Shape Modeling*, 2001. submitted.
3. Max K. Agoston. *Algebraic Topology*. Marcel Dekker Inc, New York, 1976.
4. P. S Aleksandrov. *Combinatorial Topology, Volume 1*. Graylock Press, Rochester, New York, 1956.
5. F. Arbab. Set models and Boolean operations for solids and assemblies. *IEEE Computer Graphics and Applications*, pages 76–86, November 1990.
6. C. Armstrong, Bowyer A.A., S. Cameron, J. Corney, G. Jared, R. Martin, A. Middleditch, M. Sabin M, and J. Salmon. *Djinn: A geometric interface for solid modeling*. Winchester, UK, 1998.
7. David Baraff. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10(2-4):292–352, 1993.
8. B. G. Baumgart. *Geometric modeling for computer vision*. Computer science, Stanford University, 1974.
9. R. Bidarra, K. J. de Kraker, and W. Bronsvoort. Representation and management of feature information in a cellular model. *Computer-Aided Design*, 30(4):301–313, 1998.
10. J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109–116, March 1990.
11. Jules Bloomenthal and Ken Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, 1991.
12. P.J. Booker. *A history of engineering drawing*. Northgate Publishing, London, 1979.
13. A. Bowyer. *SVLIS – Introduction and User Manual*. Information Geometers Ltd, Winchester, UK, 1994.
14. M. Boyer and N. Stewart. Modelling spaces for toleranced objects. *International Journal of Robotics Research*, 10(5):570–582, 1991.
15. I. Braid. The Synthesis of Solids Bounded by Many Faces. *Communications of the ACM*, pages 209–216, 1975.
16. I. C. Braid, R. C. Hillyard, and I. A. Stroud. Stepwise construction of polyhedra in geometric modeling. In *Mathematical Methods in Computer Graphics and Design, K. W. Brodlie ed.,*. Academic Press, 1980.
17. E. Brisson. Representing geometric structures in d dimensions: Topology and order. *Discrete and Computational Geometry*, 9:387–426, 1993.
18. P. Brunet and I. Navazo. Solid representation and operation using extended octrees, 1990.
19. V. Capoyleas, X. Chen, and C. M. Hoffmann. Generic naming in generative, constraint-based design. *Computer-Aided Design*, 28(1):17–26, 1996.
20. Jeffrey A. Chard and V. Shapiro. A multivector data structure for differential forms and equations. *IMACS Transactions Journal, Mathematics and Computers in Simulation*, 54:33–64, 2000.
21. X. Chen and C. M. Hoffmann. Towards feature attachment. *Computer-Aided Design*, 27:675–702, 1995.
22. J. Cremer and A. Steward. The architecture of Newton, a general–purpose dynamics sim-

ulator. In *IEEE International Conference on Robotics and Automation*, pages 1806–1811. IEEE Press, 1989.
23. D. P. Dobkin and M. J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Proc. 3rd Symp. on Computational Geometry*, pages 86–99, 1987.
24. S. Fortune (ed). Special issue on implementation of geometric algorithms. *Algorithmica*, 27(1), 2000.
25. R. Egli and N. F. Stewart. A framework for system specification using chains on cell complexes. *Computer-Aided Design*, 32:447–459, 2000.
26. G. Elber and M. Kim. Special issue on sweeps and Minkowski sums. *Computer Aided Design*, 31, 1999.
27. J. Ellis, G. Kedem, T. Lyerly, D. Thielman, R. Marisa, J. Menon, and H. Voelcker. The ray-casting engine and ray representations. In *Proceedings of Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pages 255–267. ACM Press, 1991.
28. L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4:74–123, 1985.
29. E. L. Gursoz, Y. Choi, and F. B. Prinz. Vertex-based representation of non-manifold boundaries. In M. Wozny, J. Turner, and K. Preiss, editors, *Geometric Modeling for Product Engineering*, pages 107–130. North Holland, 1990.
30. H. Hironaka. Triangulation of algebraic sets. In *Proceedings of Symposia in Pure Mathematics, Algebraic Geometry, ARCATA 1974, Vol. 29*, pages 165–185, 1975.
31. J. G. Hocking and G. S. Young. *Topology*. Dover Publications, New York, 1961.
32. C. H. Hoffmann and J. R. Rossignac. A road map to solid modeling. *IEEE Transactons on Visualization and Computer Graphics*, 2(1):3–10, March 1996.
33. C. M. Hoffmann. *Geometric and Solid modeling*. Morgan Kaufman, USA, 1989.
34. C. M. Hoffmann. *CRC Handbook of Discrete and Computational Geometry*, chapter Solid Modeling, pages 863–880. CRC Press LLC, Boca Raton, FL, 1997.
35. C.M. Hoffmann and K.-J. Kim. Towards valid parametric CAD models. *Computer-Aided Design*, 33(1):81–90, 2001.
36. D. Hounshell. *From the American System to Mass Production, 1800-1932*. The Johns Hopkins University Press, 1984.
37. K.C. Hui. Solid modelling with sweep-CSG representation. *Proceedings of CSG 94 Set Theoretic Solid Modelling; Techniques and Applications*, pages 119–131, 1994.
38. H. Ilieş and V. Shapiro. An approach to systematic part design. In *Proceedings of the 5th IFIP WG5.2 Workshop on Geometric Modeling in CAD*, pages 383–392, May 1996.
39. H. Ilieş and V. Shapiro. The dual of sweep. *Computer Aided Design*, 31(3):185–201, March 1999.
40. Research Triangle Institute. Interoperability cost analysis of the U. S. automotive supply chain. Technical Report 99-1, NIST, 1999.
41. Qiang Ji and Michael M. Marefat. Machine interpretation of CAD data for manufacturing applications. *ACM Computing Surveys*, 29(3):264–311, 1997.
42. J. Kripac. A mechanism for persistently naming topological entities in history-based parametric solid models. *Computer-Aided Design*, 29(2):113–122, 1997.
43. V. Kumar, D. Burns, D. Dutta, and C. Hoffmann. A framework for object modeling. *Computer-Aided Design*, 31:541–556, 1999.
44. K. Kuratowski and A. Mostowski. *Set Theory*. North-Holland Publishing Co., USA, 1976.

45. Jean-Claude Latombe. *Robot Motion Planning*. Boston: Kluwer Academic Publishers, Boston, 1991.
46. K. W. Lee and D. C. Gossard. A hierarchical data structure for representing assemblies: Part 1. *Computer Aided Design*, **17**, 1, 15 – 19, 1985.
47. S. H. Lee and Kunwoo Lee. Partial entity structure: A compact non-manifold boundary representation based on partial topological entities. In *6th ACM Symposium on Solid Modeling and Applications, Ann Arbor, MI*, 2001.
48. Y. Lee and A. Requicha. Algorithms for computing the volume and other integral properties of solids II: a family of algorithms based on representation conversion and cellular approximation, 1982.
49. S. Lien and James T. Kajiya. Symbolic method for calculating the integral properties of arbitrary nonconvex polyhedra. *IEEE Computer Graphics and Applications*, 4(10):35–41, 1984.
50. P. Lienhardt. Subdivisions of n-dimensional spaces and n-dimensional generalized maps. In *Proc. Fifth ACM Annual Symposium on Computational Geometry*, pages 218–227. 1989.
51. R. Light and D. C. Gossard. Modification of geometric models through variational geometry. *Computer-Aided Design*, 14(4):209–214, 1982.
52. M. Lin and S. Gottschalk. Collision detection between geometric models: a survey. In *IMA Conference on Mathematics of Surfaces*, pages 602–608. San Diego, CA, 1998.
53. S. Lojasiewicz. *Triangulations of semi-algebraic sets*, volume 18 of *Series 3*. Annali della Scuola Normale Superiore di Pisa, 1964.
54. E. D. Lutz. *Numerical Methods for Hypersingular and Near-Singular Boundary Integrals in Fracture Mechanics*. PhD thesis, Cornell University, Computer Science Department, May 1991.
55. M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, Maryland, USA, 1988.
56. A. Marsan and D. Dutta. A survey of process planning techniques for layered manufacturing. In *Proc. 1997 ASME Design Technical Conferences*. Sacramento, CA, 1997.
57. C. Mattiussi. An analysis of finite volume, finite element, and finite difference methods using some concepts from algebraic topology. *J. Comput. Phys.*, 133:289–309, 1997.
58. J. C. C. Mckinsey and A. Tarski. On closed elements in closure algebras. *Annals of Mathematics*, 47(1):122–162, 1946.
59. D. Meagher. Geometric modelling using octree encoding. *Computer Graphics and Image Processing*, 19, 1982.
60. J. Menon and H. Voelcker. Toward a comprehensive formulation of NC verification as a mathematical and computational problem. In *Proceedings of the 1992 Winter Annual Meeting of ASME*, volume 59, pages 147–164. Anaheim, CA, 1992.
61. J. Menon and H. Voelcker. The completeness and conversion of ray representations of arbitrary solids. In *Proc. of ACM Symposium on Solid Modeling and Applications*, pages 175–186. May 1995.
62. A. E. Middleditch. Applications of vector sum operator. Computer-Aided Design, 1988.
63. James R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley, Reading, Massachusetts, 1984.
64. M. A. O'Connor and J. R. Rossignac. SGC: A dimension independent model for pointsets with internal structures and incomplete boundaries. In *IFIP/NSF Workshop on Geometric*

Modeling, Rensselaerville, NY, 1988. North-Holland, September 1990.
65. N. Okino, Y. Kakazu, and H. Kubo. TIPS-1: technical information processing system for computer-aided design, drawing, and manufacturing. In J. Hatvany, editor, *Computer Languages for Numerical Control*, pages 141–150. North-Holland, Amsterdam, 1973.
66. J. Owen. *STEP: An Introduction*. Information Geometers Ltd, Winchester, UK, 1993.
67. J. C. Owen. Algebraic solution for geometry from dimensional constraints. In Jaroslaw Rossignac and Joshua Turner, editors, *Proc of the 1st ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pages 397–408, 1991.
68. S. Owen. A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*. Dearborn, Michigan, Oct. 26–28 1998.
69. R. S. Palmer and V. Shapiro. Chain models of physical behavior for engineering analysis and design. *Research in Engineering Design*, 5:161–184, 1993.
70. A. Paoluzzi, F. Bernardini, C. Cattani, and Ferrucci V. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1):56–102, 1993.
71. Henry M. Paynter. *Analysis and Design of Engineering Systems*. The M.I.T. Press, Cambridge, Massachusetts, 1961.
72. M. J. Pratt. Modelling of material property variation for layered manufacturing. In *Mathematics of Surfaces IX*, Cambridge, UK, 2000.
73. M. J. Pratt. Solid modeling. In *Encyclopedia of Computer Science and Technology*, volume 42 (Supplement 27), pages 295–333. Marcel Dekker, New York, NY, 2000.
74. M.J. Pratt and Bill D. Anderson. A shape modelling application programming interface for the step standard. *Computer-Aided Design*, 33:531 – 543, 2001.
75. S. Raghothama. *Models and Representations for Parametric Family of Parts*. PhD thesis, Department of Mechanical Engineering, Univeristy of Wisconsin-Madison, September 2000. Spatial Automation Laboratory Technical Report, SAL-2000-5.
76. S. Raghothama and V. Shapiro. Boundary representation deformation in parametric solid modeling. *ACM Transactions on Graphics*, 17(4):259–286, October 1998.
77. S. Raghothama and V.Shapiro. Consistent updates in dual representation systems. *Computer-Aided Design*, 32(8–9):463–477, 2000.
78. Ari Rappoport. Geometric modeling: a new fundamental framework and its practical implications. In *SMA '95: Proceedings of the Third Symposium on Solid Modeling and Applications*, pages 31–41, 1995.
79. A. Requicha. Solid modeling: a 1988 update. In B. Ravani, editor, *CAD Based Programming for Sensory Robots*, pages 3–22. Springer Verlag, 1988.
80. A. A. G. Requicha. Mathematical models of rigid solid objects. Technical report, TM-28, PAP, University of Rochester, Rochester, NY, November 1977.
81. A. A. G. Requicha. Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, 12(4):437–464, December 1980.
82. A. A. G. Requicha. Towards a theory of geometric tolerancing. *International Journal of Robotics Research*, 2(4):45–60, 1983.
83. A. A. G. Requicha and J. R. Rossignac. Constructive non-regularized geometry. *Computer-Aided Design*, 23(1), January 1991.
84. A. A. G. Requicha and R. B. Tilove. Mathematical foundations of constructive solid geometry: General topology of closed regular sets. Technical report, TM-27a, PAP, University of Rochester, Rochester, NY, June 1978.

85. A. A. G. Requicha and H. B. Voelcker. Constructive Solid Geometry. Technical report, TM-25, PAP, University of Rochester, Rochester, NY, November 1977.
86. A. A. G. Requicha and H. B. Voelcker. An introduction to geometric modeling and its applications in mechanical design and production. In J. T. Tou, editor, *Advances in Information Systems Science, Vol. 8*. Plenum Publishing, 1981.
87. A. A. G. Requicha and H. B. Voelcker. Solid modeling: A historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, pages 9–24, March 1982.
88. A. A. G. Requicha and H. B. Voelcker. Solid modeling: Current status and research directions. *IEEE Computer Graphics and Applications*, October 1983.
89. A. A. G. Requicha and H. B. Voelcker. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. *Proceedings of the IEEE*, 73(2):30–44, January 1985.
90. A. Ricci. A constructive geometry for computer graphics. *Computer Journal*, 16(3):157–160, May 1973.
91. J. R. Rossignac. Constraints in constructive solid geometry. In *Proc. of 1986 Workshop on Interactive 3D Graphics, F. Crow and S. M. Pizer (eds)*, pages 93–110. ACM Press, 1986.
92. J. R. Rossignac and A. A. G. Requicha. Offsetting operations in solid modeling. *Computer Aided Geometric Design*, 3(2):129–148, 1986.
93. Jarek R. Rossignac. Through the cracks of the solid modeling. In S. Coquillart, W. Strasser, and P. Stucki (eds), editors, *From Object Modelling to Advanced Visualization*. Springer Verlag, 1994.
94. Jarek R. Rossignac and Aristides A. G. Requicha. Solid modeling and beyond. *IEEE Computer Graphics and Applications*, pages 31–44, September 1992.
95. Jarek R. Rossignac and Aristides A. G. Requicha. Solid modeling. In J. Webster, editor, *Encyclopedia of Electrical and Electronics Engineering*. Webster, John Wiley & Sons, 1999.
96. Jarek R. Rossignac and Herbert B. Voelcker. Active zones in CSG for accelerating boundary evaluation, redundancy elimination, interference detection, and shading algorithms. *ACM Transactions on Graphics*, 8(1):51–87, 1989.
97. S. Roth. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18(2), 1982.
98. V. L. Rvachev. *Theory of R-functions and some applications*. Naukova Dumka, Kiev, 1982. in Russian.
99. V. L. Rvachev, T. I. Sheiko, V. Shapiro, and I. Tsukanov. Transfinite interpolation over implicitly defined sets. *Computer Aided Geometric Design*, 18:195–220, 2001.
100. M. Sabin and V. Shapiro. Modifications in cellular models. Technical report, NA-09, DAMTP, University of Cambridge, UK, September 1998.
101. N. Sapidis and R. Perucchio. Advanced techniques for automatic finite element meshing from solid models. *Computer-Aided Design*, 21:248–253, 1989.
102. V. Shapiro. *Representations of Semi-Algebraic Sets in Finite Algebras Generated by Space Decompositions*. PhD thesis, Cornell University, Cornell Programmable Automation, Ithaca, NY, 14853, February 1991.
103. V. Shapiro. Theory of R-functions and applications: A primer. Tech. Report TR91-1219, Computer Science Department, Cornell University, Ithaca, NY, 1991.
104. V. Shapiro. Maintenance of geometric representations through space decompositions. *In-*

ternational Journal of Computational Geometry and Applications, 7(4):383–418, 1997.
105. V. Shapiro and I. Tsukanov. Meshfree simulation of deforming domains. *Computer-Aided Design*, 31(7):459–471, 1999.
106. V. Shapiro and H. Voelcker. The role of geometry in mechanical design. *Research in Engineering Design*, 1(1), 1989.
107. V. Shapiro and D. L. Vossler. Construction and optimization of CSG representations. *Computer-Aided Design*, 23(1):4–20, January 1991.
108. V. Shapiro and D. L. Vossler. Separation for boundary to CSG conversion. *ACM Transactions on Graphics*, 12(1):35–55, January 1993.
109. V. Shapiro and D. L. Vossler. What is a parametric family of solids? In *3rd ACM Symposium on Solid Modeling and Applications, Salt Lake City, Utah, May*, 1995.
110. N. Shareef and R. Yagel. Rapid previewing via volume-based solid modeling. In *Proceedings of the 3rd Symposium on Solid Modeling and Applications*, pages 281–292. 1995.
111. C. E. Silva. Alternative definitions of faces in boundary representations of solid objects. Technical report, TM-36, PAP, University of Rochester, Rochester, NY, November 1981.
112. John M. Snyder and James T. Kajiya. Generative modeling: A symbolic system for geometric modeling. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 369–378, 1992.
113. L. Solano and P. Brunet. A system for construtive constraint-based modelling. In *B. Falcidieno and T. L. Kunii, editors, Modeling in Computer Graphics*, pages 61–83. Springer-Verlag, New York, 1993.
114. Alexi I. Sourin and Alexander A. Pasko. Function Representation for Sweeping by a Moving Solid. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):11–18, 1996.
115. V. Srinivasan. Recent efforts in mathematization of ASME/ANSI Y14.5M standard. In *Proc. 3rd CIRP Seminars on Computer Aided Tolerancing*, pages 223–232. Editions Eyrolles, Paris, 1993.
116. V. Srinivasan. A geometric product specification language based on a classification of symmetry groups. *Computer-Aided Design*, 31(11):659–668, 1999.
117. William C. Thibault and Bruce F. Naylor. Set operations on polyhedra using binary space partitioning trees. *Computer Graphics*, 21(4):153–162, 1987.
118. R. B. Tilove. Set membership classification: A unified approach to geometric intersection problems. *IEEE Transactions on Computer*, C-29(10), October 1980.
119. R. B. Tilove. Exploiting spatial and structural locality in geometric modeling. Technical report, TM-38, PAP, University of Rochester, Rochester, NY, October 1981.
120. R. B. Tilove and A. A. G. Requicha. Closure of Boolean operations on geometric entities. *Computer-Aided Design*, September 1980.
121. Robert B. Tilove. Extending solid modeling systems for mechanism design and kinematic simulation. *IEEE Computer Graphics and Applications*, 3(3):9–19, 1983.
122. Enzo Tonti. *On the Formal Structure of Physical Theories*. Istituto Di Matematica Del Politecnico Di Milano, Milan, 1975.
123. National Research Council Unit Manufacturing Process Research Committee. *Unit Manufacturing Processes: Issues and Opportunities in Research*. National Academy Press, 1995.
124. T. Varady, R. Martin, and J. Cox. Reverse engineering of geometric models — an intro-

duction. *Computer-Aided Design*, 29:255–268, 1997.
125. H. Voelcker. Modeling in the design process. In W. Dale Compton, editor, *Design and Analysis of Integrated Manufacturing Systems*. National Academy Press, Washington, DC, 1988.
126. H. B. Voelcker. The current state of affairs in dimensional tolerancing: 1997. *Integrated Manufacturing Systems*, 9(4):205–217, 1998.
127. H. B. Voelcker and A. A. G. Requicha. Research in solid modeling at the University of Rochester: 1972-1987. In L. Piegl, editor, *Fundamental Developments in Computer-Aided Modeling*. Academic Press, London, 1993.
128. W. Wang and K. Wang. Geometric modeling for swept volume of moving solids. *IEEE Computer Graphics and Applications*, pages 8–17, 1986.
129. K. J. Weiler. *Topological Structures For Geometric Modeling*. PhD thesis, Rensselaer Polytechnic Institute, 1986.
130. H. Whitney. Elementary structure of real algebraic varieties. *Annals of Mathematics*, 66(3):545–556, November 1957.
131. H. Whitney. Local properties of analytic varieties. In S. S. Cairns, editor, *Differential and Combinatorial Topology, A Symposium in Honor of Marston Morse*, pages 205–244. Princeton University Press, 1965.
132. T. C. Woo. A Combinatorial Analysis of Boundary Data Structure Schemata. *IEEE Computer Graphics and Applications*, 5(3):21–40, 1985.
133. J. Woodwark. Blends in geometric modeling. In R. Martin, editor, *The Mathematics of Surfaces II*, pages 255–297. 1987.
134. Y. Yamaguchi and F. Kimura. Nonmanifold topology based on coupling entities. *IEEE Computer Graphics and Applications*, 15(1):42–50, January 1995.
135. J. Zagajac. *Engineering Analysis Over Subsets*. PhD thesis, Cornell University, Ithaca, NY, May 1997.

Index