A Road Map To Solid Modeling

Christoph M. Hoffmann and Jaroslaw R. Rossignac

Abstract—The objective of solid modeling is to represent, manipulate, and reason about; the three-dimensional shape of solid physical objects, by computer. Such representations should be unambiguous.

Solid modeling is an application-oriented field that began in earnest in the early 1970s. [46]. Major application areas include design, manufacturing, computer vision, graphics, and virtual reality. Technically, the field draws on diverse sources including numerical analysis, symbolic algebraic computation, approximation theory, applied mathematics, point set topology, algebraic geometry, computational geometry, and data bases. Monographs and major surveys of solid modeling include [13], [19], [27], [37], [44], [45], [46].

In this road map article, we begin with some mathematical foundations of the field. We review next the major representation schemata of solids. Then, major layers of abstraction in a typical solid modeling system are characterized: The lowest level of abstraction comprises a substratum of basic service algorithms. At an intermediate level of abstraction there are algorithms for larger, more conceptual operations. Finally, a yet higher level of abstraction presents to the user a functional view that is typically targeted towards solid design. Here, we will look at some applications and at user interaction concepts.

The classical design paradigms of Solid Modeling concentrated on obtaining one specific final shape. Those paradigms are becoming supplanted by feature-based, constraint-based design paradigms that are oriented more toward the design process and define classes of shape instances. These new paradigms venture into territory that has yet to be explored systematically. Concurrent with this paradigm shift, there is also a shift in the system architecture towards modularized confederations of plug-compatible functional components. We explore these trends lightly in the last section.

Index Terms—Solid modeling, solid representations, conversion between solid representations, feature-based design, constraint-based design.

1 Mathematical Foundations

An *algebraic halfspace* is defined as the point set $H = \{(x, y, z) \mid p(x, y, z) \leq 0\}$, where $p(x, y, z)$ is a polynomial in x, y, and z with real coefficients. A *semialgebraic set* is any point set obtained as the result of a finite number of set operations (union, intersection, difference) applied to algebraic halfspaces. Bounded, homogeneously three-dimensional, semialgebraic sets were proposed in the 1970s as mathematical models for solids [43] by Requicha, who called such sets *r-sets*. Requiring homogeneous three-dimensional sets conforms with the intuition of modeling physical solids. The requirement to build r-sets from finitely many set operations applied to algebraic halfspaces is intuitively a variational restriction on the solid boundary. It conforms to the intuition that any line should not intersect the solid boundary infinitely often.

In recent years, more general mathematical models of solids have been proposed that do not require homogeneously three-dimensional sets. The relaxation responds to applying the tools of solid modeling to more general problem domains. For example, in some approaches to robot motion planning, configuration spaces are modeled and analyzed, and those spaces usually are dimensionally inhomogeneous. In fact, they may require higher dimensional spaces with noneuclidean metrics.

Other applications require modeling an inhomogeneous solid interior, for instance in the investigation of composite structures, such as certain industrial parts or electronic components. When geological structures are modeled for the oil industry, dimensionally inhomogeneous structures arise as well. Such models require sets of semialgebraic sets to express internal boundaries and irregularities.

Tools from algebraic geometry have been applied systematically to problems in geometric and solid modeling. Such tools include elimination techniques that algorithmically reduce the number of variables in a system of nonlinear equations, or else transform such systems to equivalent ones that are in a form that is especially well-suited to solving the system. For an introduction and a sampling of some of the results in this subject see, e.g., [4], [7], [8], [14], [19], [53]. Such symbolic computation algorithms tend to require highly sophisticated adaptations to geometric problems, because applying them straightforwardly usually leads to unacceptable running times or to mathematical representations that are computationally fragile.

A difficult subject in need of foundational work is tolerancing and the general problem of drawing logical conclusions from computations that are inexact. Further discussed in Section 3.1, these subjects pose extremely demanding problems that remain wide open to future advances.

2 Solid Representations

The three dominant solid representations in use are constructive solid geometry (CSG), boundary representation (Brep), and spatial subdivision. There are other representation schemata as well, less widely used. Conversion between different representations is in some cases an open problem.

Any solid representation should admit the unambigu-

- *C. Hoffmann is with the Computer Science Department, at Purdue University, West Lafayette, IN 47907. E-mail: cmh@cs.purdue.edu..*
- *J.R. Rossignac is with IBM, T.J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: jarek@watson.ibm.com.*

For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number V96001.

ous, algorithmic determination of point membership: Given any point $p = (x, y, z)$, we have to be able to determine algorithmically whether the point is inside, outside, or on the surface of a solid. Moreover, restrictions on the topology of the solid and its embedding are desirable that exclude, for example, fractal solids.

These restrictions are intuitive. However, as indicated before, increasingly solid modeling departs from this strict notion of solid and permits representing a mixture of solids, surfaces, curves and points. Such generalizations require the development of new or extensively modified classical data structures to represent solids.

2.1 Constructive Solid Geometry

Classical *Constructive Solid Geometry*, CSG, represents a solid as a set-theoretic Boolean expression of *primitive* solid objects, of a simpler structure. Both the surface and the interior of the final solid are thereby implicitly defined. The CSG representation is valid if the primitives are valid. A solid's surface is closed and orientable and encloses a volume. The traditional CSG primitives are block, sphere, cylinder, cone, and torus.

A solid is represented as an algebraic expression that uses rigid motions and regularized set operations. The traditional operations are regularized union, intersection and difference. A regularized set operation requires taking the closure of the interior of the set-theoretic result. Regularization eliminates lower-dimensional components from the solid representation such as interior, or dangling exterior, faces, edges, and vertices.

Each solid has a default coordinate system. Using a rigid body transformation, the solid is positioned relative to a global coordinate system. A Boolean operation then combines the solids with respect to the common coordinate system. The result solid can be repositioned by another rigid-body transformation.

As example, consider Fig. 1. Using the coordinate system conventions for the primitives as shown, the CSG representation of the T-bracket is the expression

```
block(8, 3, 1) <* move(block(1, 3, 3), (0, 4, 1))
    - *  move(cylinder(0.5, 1), (1.8, 1.8, -0.5))    (1)
```

where the * indicates a regularized operation.

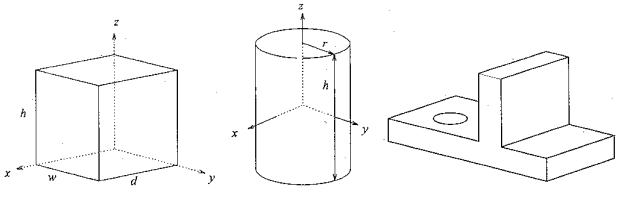

Fig. 1. Left and middle: CSG primitives `block`(w, d, h) and `cylinder`(r, h) with default coordinate systems. Right: T-bracket as union of two blocks minus a cylinder.

Basic operations one performs on CSG representations are classifying points, curves, and surfaces with respect to a solid; detecting redundancies in the representation; and approximating CSG objects systematically.

More general primitives are obtained by considering the volume covered by sweeping a solid along a space curve, or sweeping a planar contour bounding an area. Defining a sweep is delicate, requiring many parameters to be exactly defined, but simple cases are widely used. They are extrusion, i.e., sweep along a straight line; and revolution, i.e., a sweep about an axis. The evaluation of general sweeps can be done by a number of methods; e.g., [1], [27], [40]. Spatial deformations of solids are another way to obtain new solids, whether they be used as primitives or as final solids. A general set of primitives is the set of algebraic halfspaces; [4].

2.2 Boundary Representation

In *boundary representation*, Brep, the solid surface is represented as a quilt of faces, edges, and vertices. A distinction is drawn between the topological entities, vertex, edge, and face, related to each other by incidence and adjacency, and the geometric location and shape of these entities; Fig. 2. For example, when representing polyhedra, the faces are polygons described geometrically by a face equation plus a description of the polygon boundary. Geometrically, the entities in a Brep must not intersect anywhere except in edges and vertices that are explicitly represented in the topology data structure. In addition to the classification operations mentioned for CSG, Boolean union, intersection and difference operations are usually implemented for Brep systems. Both regularized and nonregularizing Boolean operations may occur.

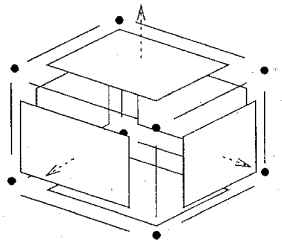

Fig. 2. Topological entities of a box. Adjacency and incidence are recorded in boundary representation (Brep). Dotted arrows indicate face orientation.

Different Brep schemata appear in the literature, divided into two major families. One family restricts the solid surfaces to oriented manifolds. Here, every edge is incident to two faces, and every vertex is the apex of a single cone of incident edges and faces. The second family of Brep schemata allows oriented nonmanifolds in which edges are adjacent to an even number of faces. When these faces are ordered radially around the common edge, consecutive face pairs alternatingly bound solid interior and exterior [62]. See Fig. 3 for examples.

More general nonmanifold Breps are used in systems that combine surface modeling with solids modeling. In such representation schemata, a solid may have interior (two-sided) faces, dangling edges, and so on [62]. Such systems are used, for example, by the oil industry for modeling geological structures, or to model civil engineering projects.

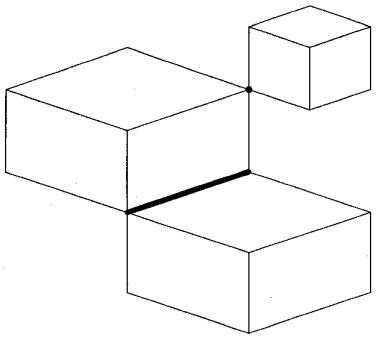

Fig. 3. A nonmanifold solid without dangling or interior faces, edges and vertices; the nonmanifold edges and vertices are drawn with a thicker pen.

The topology may be restricted in more technical ways. For instance, the interior of a face may be required to be homeomorphic to a disk, and edges to have two distinct vertices. In that case, the Brep of a cylinder would have four faces, two planar and two curved. Such restrictions often reflect algorithmic requirements of the implementation and are sometimes transparent to users of the system.

2.3 Spatial Subdivision Representations

Spatial subdivision decomposes a solid into cells, each with a simple topological structure and often also with a simple geometric structure. We categorize subdivision representations into *boundary conforming* and *boundary approximating*.

Boundary conforming subdivision schemata include *meshes* [60] and the *binary space partition tree* (BSP tree) [38]. Mesh representations are used in finite element analysis, a method for solving continuous physical problems. The mesh elements can be geometric tetrahedra, hexahedra, or other simple polyhedra, or they can be deformations of topological polyhedra so that curved boundaries can be approximated exactly.

Binary space partition trees are recursive subdivisions of 3-space. Each interior node of the tree separates space into two disjoint point sets. In the simplest case, the root denotes a separator plane. All points of $\mathbf{R}^3$ below or on the plane are represented by one subtree, all points above the plane are represented by the other subtree. The two point sets are recursively subdivided by half planes at the subtree nodes. The leaves of the tree represent cells that are labeled *in* or *out*. The (half) planes are usually face planes of a polyhedron, and the union of all cells labeled *in* is the polyhedron modeled.

A *selective geometric complex* (SGC) is a collection of semi-algebraic sets in n-space that are dimensionally homogeneous; [48]. The cells are disjoint, open, and connected. Since curved boundaries are modeled, SGCs are a boundary-conforming subdivision schema. Adjacency information can be added in a compact form. When allowing that cells be disconnected, not necessarily open, and dimensionally inhomogeneous, constructive nonregularized geometry can be defined. Algorithms for this more general representation are discussed in [49].

Boundary approximating representations are *grids* [60] and *octrees* [6], [51]. In grids, space is subdivided in conformance with a coordinate system. Typically, the division is into hexahedra whose sides are parallel to the Cartesian coordinate planes. When a different coordinate system is used, the division would differ accordingly. For instance, for cylindrical coordinates the division would be into concentric sectors, and so on. The grids may be regular or adaptive, and may be used to solve continuous physical problems by differencing schemes. Changes in the coordinate system may be used to simplify the numerical treatment of the differential equations to be solved with the grid; [60]. Rectilinear grids that are geometrically deformed can be boundary-conforming. Otherwise, they only approximate curved boundaries.

An octree divides a cube into eight subcubes. Each subcube may be further subdivided recursively. Cubes and their subdivision are labeled white, black or grey. A grey cube is one that has been subdivided and contains both white and black subcubes. A subcube is black if it is inside the solid to be represented, white if it is outside. Quadtrees, the two-dimensional analogue of octrees, are used in many geographical information systems [50].

2.4 Medial Surface Representations

Medial axis and medial surface can unambiguously represent two-dimensional domains and three-dimensional solids, respectively. The representations are not widely used for this purpose at this time. However, as explained later, some sophisticated meshing algorithms are based of the medial axis and the medial surface, and some commercial systems are implementing medial surface construction algorithms.

The medial axis of a two-dimensional domain is defined as the closure of the locus of centers of disks inscribed within the domain. A disk is maximal if no other disk properly contains it. An example is shown in Fig. 4 along with some maximal disks.

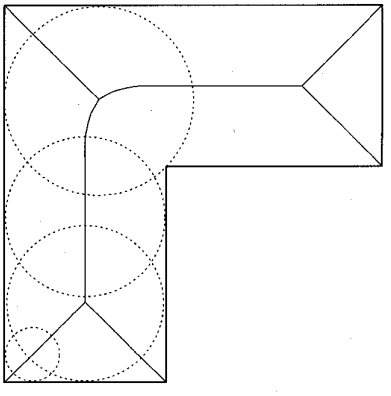

Fig. 4. L-shaped domain and associated medial axis. Some maximal inscribed circles contributing to the medial axis are also shown.

The medial surface of a solid is the closure of the locus of centers of maximal inscribed spheres. When we know the radius (the limit radius in case of closure points) of the corresponding sphere for each point on the medial surface, then an unambiguous solid representation is obtained that is sometimes called the medial axis transform (MAT). The

MAT has a number of intriguing properties that have been explored in mathematics and physics; [21]. For example, by enlarging the radius values by a constant, the MAT of a dilation of the solid is obtained.

2.5 Conversion Between Representations

Most solid modeling systems use Brep. Conversion from CSG to Brep is well understood, and is implemented as regularized Boolean operations on Brep solids. An extensive literature addresses these complex algorithms; [13], [19], [37].

The conversion from Brep to CSG is not fully understood. In the polyhedral case, the conversion is essentially the same as the conversion from Brep to BSP tree [23]. The major issue in the conversion is to find cells that partition the solid interior and are boundary conforming. Briefly, the controlling number of cells, as well as finding surfaces that make the cells boundary-conforming, constitute obstacles that are difficult to surmount. Conversion involving higher degree surfaces is largely open [53], [54]. However, some progress has been made recently by Naylor and Rogers [39] in the case of Bézier curves and B-splines. Roughly speaking, a coarse BSP tree is constructed that encloses sections of the curve in convex polygonal regions. On demand, the tree can be extended dynamically thereby refining the enclosing regions. In this way, points may be classified efficiently with respect to the curve to a required resolution.

There are several algorithms for converting from CSG or Brep to the MAT. Some are based on geometric principles, some on a Delaunay triangulation of an approximated boundary, some on a grid subdivision of ambient space [12], [57], [56], [58]. The conversion from MAT to Brep has been addressed in [61].

The conversion from CSG or Brep to mesh representations is a partially solved problem when the conversion is done for finite element analysis or other numerical treatment of continuum problems. In that context, the problem is not a geometric problem alone: The quality of the subdivision must also be judged by nongeometric criteria that come from the nature of the physical problem and the numerical algorithm used to solve it. Many approaches are based on octree subdivision, on Delaunay triangulation, and on MAT computations; [24].

2.6 Geometric Coverage

The range and geometric representation of solid surfaces is referred to as geometric coverage. Polyhedral modeling restricts to planes. Classical CSG allows only planes, cones, cylinders, spheres, and tori. Experimental modelers have been built allowing arbitrary algebraic halfspaces [4]. Most commercial and many research modelers use B-splines (uniform or nonuniform, nonrational or rational) or Bézier surfaces. The properties and algorithmic treatment of these surfaces is studied by computer-aided geometric design (CAGD) and has an extensive literature. See, for example, [16], [30], [31] for introductions into the subject, and [15] for a sampling of recent results.

2.7 Assemblies

Most mechanical products are assemblies of parts. To represent assemblies, we must represent the individual parts, as described before. In addition, the relationship between mating parts should be expressed with additional information. Such information tries to identify functional properties of the interacting parts and constraints they have to satisfy; e.g., [35].

3 Layers of a Solid Modeling System

A solid modeling system spans several layers:

1) On the lowest level, there is the substratum of arithmetic and symbolic computations.
2) Next, there is an intermediate level comprising the algorithmic infrastructure. This level implements larger conceptual operations.
3) Finally, system interfaces present a view of the functional capabilities of the system. Such interfaces include the graphical user interface (GUI) as well as application programming interfaces (API).

Ideally, the levels of abstraction should be kept logically apart. However, such a separation is fundamentally limited by problems that arise from the interaction of numeric and symbolic computation.

3.1 The Substratum

The substratum consists of many low-level computations and tests, for example vector computations, simple incidence tests, and computations for ordering points along a simple curve in space. Ideally, these operations create an abstract machine whose functionality simplifies the algorithms at the intermediate level of abstraction. But it turns out that this abstract machine is unreliable in a subtle way when implemented using floating-point arithmetic. Exact arithmetic would be desirable, but is widely held to be unacceptably inefficient when dealing with solids that have curved boundaries. We call this the *robustness problem* of solid modeling. To illustrate how inexact arithmetic at the substratum level can impact the geometric computation, consider modeling polyhedral solids, the simplest possible situation for solid modeling.

All computational decisions that arise in the course of a regularized Boolean operation on polyhedra can be reduced to determining the sign of 4×4 determinants [59]. Geometrically, this is a test of whether a point is above, on or below a plane. When the determinant's value is nearly zero, floating-point evaluation will decide based on a tolerance. But the decision is unreliable because logically equivalent tests may arise as different determinants in the course of the algorithm, and some of the determinants will have small, others large values (see [19], Chapter 4). This gives an opportunity for the algorithm to build inconsistent data structures and fail. The problems are magnified when dealing with curved solids.

Usually, robustness problems are approached with a mixture of heuristics and user work-arounds discovered by trial-and-error. For example, consider the CSG expression of the T-bracket shown in Fig. 1. The CSG expression (1) given before exactly matches the faces of the two blocks. While this may be tolerable in principal orientation, it could result in a small crack between the two blocks when the bracket is rotated.

The robustness problem has also arisen in computational geometry. It appears to be quite difficult to devise a solution that negotiates pragmatically conflicting demands of efficiency of computation and reliability of the decision making that would require in some cases extraordinary precision. Sug-

gested trade-offs include implementing a meta-level computation that estimates the required precision for any particular geometric computation. If normal floating-point arithmetic is sufficient, then no further action is taken. However, if the accuracy is not sufficient, then an extended precision computation is carried out on which to base a reliable decision; [18].

A problem that would appear to be related but arises from different roots is the tolerancing problem. Given a physical part and its electronic model, the physical part will deviate from the nominal dimensions and geometry of the model. Finding mathematical models that quantify such deviations, manipulating the models and reasoning about them is a subject that is of interest to solid modeling. It is plausible that effective tolerancing models and algorithms impact the robustness problem.

3.2 Algorithmic Infrastructure

Algorithmic infrastructure is a prominent research subject in solid modeling that addresses the development of efficient and robust algorithms for carrying out major geometric computations that arise in solid modeling. The problems include point/solid classification, computing the intersection of two solids, determining the intersection of two surfaces, interpolating smooth surfaces to eliminate sharp edges on solids, and many more. See [13], [19], [37], [27], [44], [45] and the literature in CAGD.

An important consideration when devising infrastructure is that the algorithms are often used by other programs or system components, as service routines. Therefore, they must be extremely reliable and completely autonomous, requiring no user intervention in exceptional situations. This places an extra burden on the development of geometric algorithms that is absent in many other areas of geometric computation.

The major geometric computations implemented at the infrastructure level have to balance the conflicting goals of efficiency, accuracy, and robustness. For this reason, many operations that have a sizable literature already continue to be researched in efforts to seek new perceived optima. Moreover, new mathematical surface representations continue to be devised that necessitate different approaches. Some of the major operations on which research continues include the following.

SURFACE INTERSECTION. Given two bounded areas of two surfaces, determine all intersection curve components. A sampling of different approaches includes [19], [29], [30], [31]. Many algorithms have two phases. In the first phase, all components of the intersection curve are identified and some representative points on them. In the second phase, each component is evaluated, for instance by a marching algorithm, and is then approximated in a suitable format. Both phases are difficult to make fully reliable.

OFFSETTING. Given a surface, its offset is the set of all points that have fixed minimum distance from the surface. Offsets can have self-intersections that must be eliminated. There is a technical relationship between offsetting and forming the MAT [21]. Note that offsetting is used to determine certain blending surfaces, and is also used in the solid operation of *shelling* that creates thin-walled solids [17].

BLENDING. Given two intersecting surfaces, a third surface is interpolated between them to smooth the intersection edge. A simple example is shown in Fig. 5. Several difficulties must be addressed; e.g., [5]. First, given two surfaces and contact curves on both, find a third surface touching the two given ones along the contact curves. Techniques exist both for implicit and for parametric surfaces; [2], [20], [25], [41]. A second, less well-understood issue is how to devise the contact curves when dealing with solids, so that the curves connect properly at adjacent faces, behave correctly at vertices, and so on. Again, the requirement of developing a fully automated solution exacerbates the second issue.

DEFORMATIONS. Given a solid body, deform it locally or globally. The deformation could be required to obey constraints such as preserving volume [42] or optimizing physical constraints. For example, [2] deforms a basic shape for a ship hull to minimize drag in fluids of various viscosities.

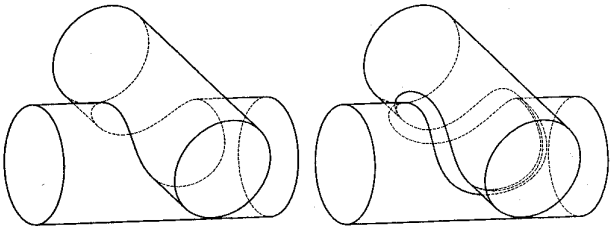

Fig. 5. Left: Two cylinders intersecting in a closed edge. Right: Edge blended with a constant-radius, rolling-ball blend; note that the bounding curves of the blend are shown.

3.3 User Interfaces

Ultimately, the functional capabilities of a solid modeling system have to be presented to a user, typically through a graphical user interface (GUI). The GUI should conceptualize the functionalities and application needs. As in programming language design, this conceptual view can be convenient or inconvenient for a particular application. Research on GUIs therefore is largely done with a particular application area in mind. Some of the technical aspects and design paradigms that are involved in current conceptualizations are explored next.

4 Features and Constraints

Two design paradigms are emerging for manufacturing applications: *feature-based design* and *constraint-based design*. Research on both has some history, but it has been only in the past five years that these paradigms are appearing in commercial practice. Along with this transition, specific technical problems are coming into focus that have received relatively light treatment by researchers.

The new design paradigms expose a need to reconsider solid representations at a different level of abstraction: The representations described before are for individual solids. However, in the new design methodologies we need to represent entire *classes* of solids, comprising a *generic design*. Roughly speaking, solids in a class are built structurally in the same way, from possibly complex shape primitives, and are instantiated subject to constraints and dimensions that interrelate specific shape elements. How such a class should

be defined precisely, how each generic design should be represented, and how designs should be edited are all important research issues of considerable depth.

4.1 Feature-Based Design

Feature-based design is usually understood in manufacturing applications to mean designing with shape elements such as slots, holes, pockets, etc., that have significance to manufacturing applications, relating to function, manufacturing process, performance, cost, etc.; e.g., [52], [63]. Focusing on shape primarily, we conceptualize solid design in terms of three classes of features: generative, modifying, and referencing features; [26]. A feature is added to an existing design using attachment attributes and placement constraints. Subsequent editing may change both attributes and constraints.

As example, consider the solid shown in Fig. 6 right. A hole was added to the design on the left. This new "feature" can be specified by giving the diameter of the hole, placing its cross section, a circle, on the side face, and requiring that the hole extend to the next face encountered. Should the slot at which the hole ends be moved or altered by subsequent editing, then the hole would automatically be adjusted to the required depth. See, e.g., [11], [10], [47].

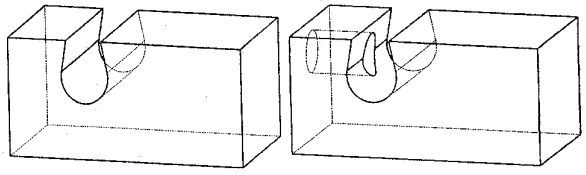

Fig. 6. Left: Solid block with a profiled slot. Right: After adding a hole with the attribute "through next face," an edited solid is obtained. If the slot is moved later, the hole will be adjusted automatically.

4.2 Constraint-Based Design

Constraint-based design refers to specifying shape with help of constraints, when placing features or when defining shape parameters. For instance, assume that we are to design a cross section for use in defining a solid of revolution. A rough topological sketch is prepared (Fig. 7 left) and is annotated with constraints by the user. Then, a computation instantiates the rough sketch to one that satisfies the constraints precisely (Fig. 7 right). Auxiliary geometric structures can be added, such as an axis of rotation.

There is an extensive literature on constraint solving, from a variety of perspectives. See, e.g., [14], [28], [32], [36]. Some authors concentrate on applications in solid modeling or assembly modeling of the nature similar to the example just given. Others investigate the use of constraints to reason about geometric properties, or as an application of symbolic algebraic computation.

Commercial solid modeling systems use both features and constraints in the user interface. Typically, the constraints on cross sections and other 2-dimensional structures are unordered. To evaluate such constraints, the solver has to first derive a solution strategy. The constraints on 3-dimensional geometry are usually defined and considered in a fixed sequence. Solving systems of unordered constraints is sometimes referred to as *variational constraint solving*. Mathematically, it is equivalent to solving a system of (nonlinear) simultaneous equations. Solving constraints in a fixed sequence is also known as *parametric constraint solving*. The latter is equivalent to solving a system of nonlinear equations that has a fixed, triangular structure.

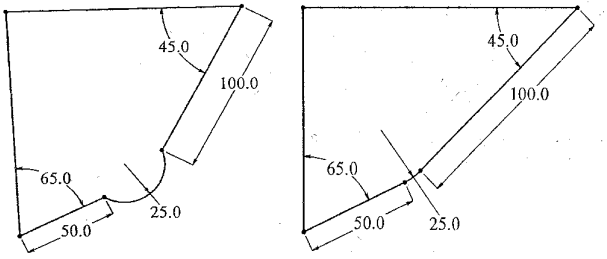

Fig. 7. Geometric constraint solving. Input to solver shown left. In addition to the constraints shown, the arc should be tangent to the adjacent segments, and the two other segments should be perpendicular. Output of the constraint solver shown right.

A well-constrained geometric constraint problem corresponds naturally to a system of nonlinear algebraic equations with a zero-dimensional set of solutions [8]. Thus, there are, in general, different solutions of the same, well-constrained geometric problem. An example is shown in Fig. 8.

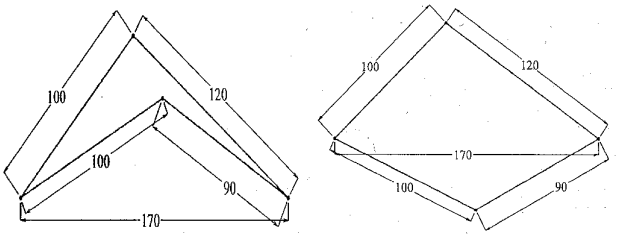

Fig. 8. The well-constrained geometric problem shown has two distinct solutions. Which one to select in a design application and how is an interesting research issue.

Therefore, the constraint solver must apply a strategy for efficiently selecting one of the potential solutions. Since the set of solutions of n nonlinear simultaneous equations can be proportional to 2^n, solvers do not, in general, explore the space of possible solutions but apply heuristics. Very few solvers have the ability to systematically explore the entire solution space [3].

Symbolic algebraic computation has developed algorithms that convert a nontriangular system of nonlinear equations into a triangular system; e.g., [7]. The distinction between parametric and variational constraint solving is therefore artificial in theory, except that the variable ordering is determined in a parametric constraint problem. However, full-scale triangularization of systems of nonlinear equations is not tractable in many cases, so the distinction is relevant in practice. Moreover, a predetermined sequential evaluation of constraints is simple to implement and can be interfaced easily with conditional constraint evaluation, thereby increasing the expressive power of the constraint system without raising fundamental implementation issues. For these reasons, many developers of solid modeling systems leverage core modeling capabilities by such extensions.

4.3 Semantic Problems

When constraints and parameters are used in solid design, a *generic design* is obtained. Generic designs are instantiated by constraint values, and may be edited by changing the constraint values, the constraint schema, and the feature attributes. A design so edited can then be automatically re-instantiated by the solid modeler. A central difficulty implementing this scenario, however, is that the generic design is usually defined visually on the basis of a particular instance, and when the design changes, the instance geometry is no longer present. Thus, visually identified instance structures must be suitably described generically, so that re-instantiation can be carried out correctly. This *persistent naming problem* has been characterized in [22], [55].

As example, consider the solid shown in Fig. 9 left. It was constructed as follows: First, a rectangle was drawn and extruded into a block. On the front face of the block, a circle was drawn as profile of a slot across the top of the block. Then, the left edge of the slot was visually identified for rounding. The result is shown in Fig. 9 left.

This design is now edited by altering the position of the circular slot profile. The correct result is shown in Fig. 9 middle, but some systems may construct instead the shape of Fig. 9 right, clearly an error. The problem is how to describe the edge to be rounded in the generic design. An identification such as "left" is not reasonable because it presupposes a coordinate system which we do not have. Moreover, reliance on a default coordinate system is questionable in view of the great variability of the shape under constraint-based editing. What is needed is an unambiguous semantics for editing generic design that is intuitive and complete.

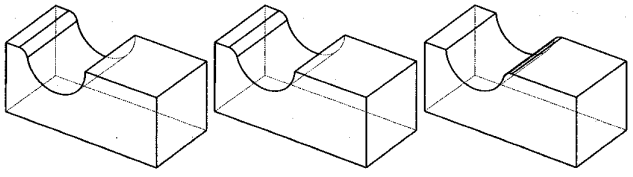

Fig. 9. A block with a slot and round on the left edge is shown left. After editing, in this case raising the depth of the slot, re-instantiation should produce the solid shown in the middle. However, some systems may re-instantiate as shown to the right, an error. The confusion relates to the *persistent naming* problem.

Closely related to defining a well-founded semantics, a design representation is also needed. It seems appropriate that the representation would be unevaluated and that it would be instantiated depending on parameter values, attributes, and constraints [11]. Such a design representation would formalize the behavior of solid modeling operations also under editing that involves simply changing constraint values and feature attributes.

There seems to be little published work on these topics although they are of intense commercial interest and intellectually challenging. Whether a formal, high-level representation is devised or not, the research issues characterized before come up in the implementation of constraint-based, feature-based CAD systems, and such systems are being developed and offered in the commercial sector; see [9], [10], [33], [34].

Since generic design representations must be responsive to the information content between the user interface, the core solid modeler, and a variety of other system components, devising a practical representation has implications on system architectures. That is, when formalizing the information flow between functional components, interface specifications are obtained. Whenever the formalization seeks independence from the specific implementation of the system components, system modularization is facilitated. Ultimately, this will accelerate the current trend to decompose solid modeling systems into standardized components that can function interchangeably and can be combined in a variety of ways.

ACKNOWLEDGMENTS

Dr. Hoffmann gratefully acknowledges the support of the Office of Naval Research under contract N00014-90-J-1599, of the National Science Foundation under grants CCR 95-05745 and CDA 92-23502, and the support of Purdue's Engineering Research Center under National Science Foundation grant ECD 88-03017.

REFERENCES

[1] D. Blackmore and M. Leu, "A Differential Equations Approach to Swept Volume," *Proc. Rensselaer Second Int'l Conf. on Computer-Integrated Manufacturing*, pp. 143–149, Troy, N.Y., 1990.
[2] M. Bloor and M. Wilson, "Generating Blending Surfaces with Partial Differential Equations," *Computer Aided Design*, vol. 21, pp. 165–171, 1989.
[3] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige, "A Geometric Constraint Solver," *Computer Aided Design*, 1995, to appear.
[4] A. Bowyer, J.H. Davenport, D.A. Lavender, P.S. Milne, and A.F. Wallis, "A Geometric Algebra System," D. Kapur, ed., *Integration of Symbolic and Numeric Methods*. MIT Press, 1991.
[5] I. Braid, "Nonlocal Blending of Boundary Models," *Computer-Aided Design*, 1996, to appear.
[6] P. Brunet and I. Navazo, "Solid Representation and Operation Using Extended Octrees," *ACM Trans. Graphics*, vol. 9, pp. 170–197, 1990.
[7] B. Buchberger, "Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory," N.K. Bose, ed., *Multidimensional Systems Theory*, pp. 184–232. D. Reidel Publishing Co., 1985.
[8] B. Buchberger, G. Collins, and B. Kutzler, "Algebraic Methods for Geometric Reasoning," *Ann. Reviews in Computer Science*, vol. 3, no. 85, p. 120, 1988.
[9] V. Capoyleas, X. Chen, and C.M. Hoffmann, "Generic Naming in Generative, Constraint-based Design," to appear in *Computer Aided Design*.
[10] X. Chen and C. Hoffmann, "Editing Feature Based Design," to appear in *Computer Aided Design*.
[11] X. Chen and C. Hoffmann, "Towards Feature Attachment," *Computer Aided Design*, 1995.
[12] C. Chiang, C. Hoffmann, and R. Lynch, "How to compute offsets without self-intersection," *Proc SPIE Conf. Curves and Surfaces in Computer Vision and Graphics*, vol. 1610, pp. 76–87. *Int'l Society for Optical Engineering*, 1991.
[13] H. Chiyokura, *Solid Modeling with Designbase*. Addison-Wesley, 1988.
[14] C.-S. Chou, *Mechanical Theorem Proving*. D. Reidel Publishing, 1987.
[15] M. Daehlen, T. Lyche, and L. Schumaker, *Math. Methods for Curves and Surfaces*. Vanderbilt University Press, 1995.
[16] G. Farin, *Curves and Surfaces for Computer-Aided Geometric Design*. Academic Press, 1988.
[17] M. Forsyth, "Shelling and Offsetting Bodies," *Proc. Third Symp. on Solid Modeling*. ACM Press, 1995.
[18] S. Fortune, "Polyhedral Modeling with Multiprecision Integer Arithmetic," *Proc. Third Symp. on Solid Modeling*. ACM Press, 1995.
[19] C. Hoffmann, *Geometric and Solid Modeling*. Morgan Kaufmann, 1989.
[20] C. Hoffmann, "Algebraic and Numerical Techniques for Offsets and Blends," S.M.M. Gasca, and W. Dahmen, eds., *Computations of Curves and Surfaces*, pp. 499–528. Kluwer Academic, 1990.
[21] C. Hoffmann, "Computer Vision, Descriptive Geometry, and Classical Mechanics, B. Falcidieno, and I. Herman, eds., *Computer Graphics and Math.*, Eurographics Series, pp. 229–244, Springer Verlag, 1992.

[22] C. Hoffmann, "On the Semantics of Generative Geometry Representations," *Proc. 19th ASME Design Automation Conf.*, vol. 2, pp. 411–420, 1993.
[23] C. Hoffmann, "On the Separability Problem of Real Functions and its Significance in Solid Modeling," *Computational Algebra*, pp. 191–204, Marcel Dekker, 1993. *Lecture Notes in Pure and Applied Math.*, pp. 151.
[24] C. Hoffmann, "Geometric Approaches to Mesh Generation," I. Babuska, J. Flaherty, W. Henshaw, J. Hopcroft, J. Oliger, and T. Tezduyar, eds., *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*. Springer Verlag, 1995.
[25] C. Hoffmann and J. Hopcroft, "The Potential Method for Blending Surfaces and Corners, G. Farin, ed., *Geometric Modeling*, pp. 347–365. SIAM, 1987.
[26] C. Hoffmann and R. Juan, "Erep, An Editable, High-level Representation for Geometric Design and Analysis," P. Wilson, M. Wozny, and M. Pratt, eds., *Geometric Modeling for Product Realization*, pp. 129–164. North Holland, 1992.
[27] C. Hoffmann and G. Vaněček, "Fundamental Techniques for Geometric and Solid Modeling," C.T. Leondes, ed., *Advances in Control and Dynamics*, vol. 48, pp. 101–165. Academic Press, 1991.
[28] C. Hoffmann and P. Vermeer, "Geometric Constraint Solving in R^2 and R^3," D.Z. Du and F. Hwang, eds., *Computing in Euclidean Geometry*. World Scientific Publishing, second edition, 1994.
[29] M. Hohmeyer, "Surface Intersection," PhD thesis, Univ. of California, Berkeley, 1992.
[30] M. Hosaka, *Modeling of Curves and Surfaces in CAD/CAM*, New York, Springer Verlag, 1992.
[31] J. Hoschek and D. Lasser, *Computer Aided Geometric Design*, A.K. Peters, 1993.
[32] G. Kramer, *Solving Geometric Constraint Systems*, MIT Press, 1992.
[33] J. Kripac, "Topological ID System–A Mechanism for Persistently Naming Topological Entities in History-based Parametric Solid Models," PhD thesis, Czech Technical Univ., Prague, 1993.
[34] J. Kripac, "A Mechanism for Persistently Naming Topological Entities in History-based Parametric Solid Models," *Proc. Third Symp. on Solid Modeling*. ACM Press, 1995.
[35] J.-C. Latombe and R. Wilson, "Assembly Sequencing with Toleranced Parts," *Proc. Third Symp. on Solid Modeling*. ACM Press, 1995.
[36] W. Leler, *Constraint Programming Languages*. Addison-Wesley, 1988.
[37] M. Mäntylä, *An Introduction to Solid Modeling*. Computer Science Press, 1988.
[38] B. Naylor, "Binary Space Partitioning Trees As an Alternative Representation of Polytopes," *Computer Aided Design*, vol. 22, 1990.
[39] B. Naylor and L. Rogers, "Constructing Binary Space Partitioning Trees From Piecewise Bézier curves," *Graphics Interface '95*, 1995.
[40] A. Pasko and V. Savchenko, "Function Representation for Sweeping By a Moving Solid," *IEEE Trans. Visualization and Computer Graphics*, 1996.
[41] J. Peters, "Joining Smooth Patches Around a Vertex to Form a C^1 Surface," *Computer Aided Geometric Design*, vol. 9, pp. 387–411, 1992.
[42] A. Rappoport, A. Sheffer, and M. Bercovier, "Volume-preserving Free-form Solids," *IEEE Trans. Visualization and Computer Graphics*, 1996.
[43] A. Requicha, "Mathematical Models of Rigid Solids," Tech. Report PAP Tech. Memo 28, Univ. of Rochester, 1977.
[44] A. Requicha, "Representations For Rigid Solids: Theory, Methods, and Systems," *ACM Computing Surveys*, vol. 12, pp. 437–464, 1980.
[45] A. Requicha, "Solid Modeling—a 1988 Update," B. Ravani, ed., *CAD Based Programming for Sensory Robots*, pp. 3–22, New York: Springer Verlag, 1988.
[46] A. Requicha and J. Rossignac, "Solid Modeling and Beyond," Tech. Report RC 17676, IBM, Yorktown Heights, 1992.
[47] J. Rossignac, "Issues in Feature-based Editing and Interrogation of Solid Models," *Computers and Graphics*, vol. 14, pp. 149–172, 1990.
[48] J. Rossignac and M. O'Conner, "SGC: a Dimension-independent Model for Pointsets with Internal Structure," M. Wozny, J. Turner, and K. Preiss, eds., *Geometric Modeling for Product Engineering*, pp. 145–180. North Holland, 1989.
[49] J. Rossignac and A. Requicha, "Constructive Nonregularized Geometry," *Computer-Aided Design*, vol. 23, pp. 21–32, 1991.
[50] H. J. Samet, *Applications of Spatial Data structures: Computer Graphics, Image Processing, and GIS*. Addison-Wesley, 1989.
[51] H. J. Samet, *Design and analysis of Spatial Data Structures: Quadtrees, Octrees, and other Hierarchical Methods*. Addison-Wesley, 1989.
[52] J. Shah, D. Hsiao, and J. Leonard, "A Systematic Approach for Design-manufacturing Feature Mapping," P. Wilson, M. Wozny, and M. Pratt, eds., *Geometric Modeling for Product Realization*, pp. 205–222, North Holland, 1992.
[53] V. Shapiro, "Representations of Semialgebraic Sets in Finite Algebras Generated by Space Decompositions," PhD thesis, Cornell University, Sibley School of Mechanical Engineering, 1991.
[54] V. Shapiro and D. Vossler, "Separation for Boundary to CSG Conversion, *ACM Trans. Graphics*, vol. 12, pp. 35–55, 1993.
[55] V. Shapiro and D. Vossler, "What Is a Parametric Family of Solids?," *Proc. Third ACM Symp. on Solid Modeling*, 1995.
[56] D. Sheehy, C. Armstrong, and D. Robinson, "Computing the Medial Surface of a Solid From a Domain Delaunay Triangulation," *IEEE Trans. Visualization and Computer Graphics*, 1996.
[57] E. Sherbrooke, N. Patrikalakis, and E. Brisson, "An Algorithm for the Medial Axis Transform of 3-d Polyhedral Solids," *IEEE Trans. Visualization and Computer Graphics*, 1996.
[58] V. Srinivasan and L. Nackman, "Voronoi Diagram of Multiply Connected Polygonal Domains," *IBM J. Research and Development*, vol. 31, pp. 373–381, 1987.
[59] K. Sugihara and M. Iri, "A Solid Modeling System Free From Topological Inconsistency," *J. Information Processing*, vol. 12, pp. 380–393, 1989.
[60] J. Thompson, Z. Warsi, and W. Mastin, *Numerical Grid Generation*. North Holland, 1985.
[61] P. Vermeer, "Medial Axis Transform to Boundary Representation Conversion," PhD thesis, Purdue Univ., 1994.
[62] K. Weiler, "Topological Structures for Geometric Modeling," PhD thesis, Rensselaer Polytechnic Inst., 1986.
[63] P. Wilson and M. Pratt, "A Taxonomy of Features for Solid Modeling," J.E.J. Wozny, and H.W. McLaughlin, eds., *Geometric Modeling for CAD Applications*, pp. 125–136. North Holland, 1988.

Christoph M. Hoffmann is a professor of computer science at Purdue University. His research interests include solid modeling, geometric reasoning and constraint solving, and applications in design for manufacturing. Dr. Hoffmann edits seven technical journals in computer graphics, solid modeling, geometric design, computational geometry, and symbolic computation.

Jaroslaw R. Rossignac joined IBM Research in 1985 with a PhD in electrical engineering on solid modeling from the University of Rochester, New York. He is now a senior manager of Visualization, Interaction, and Graphics, covering 50 people in six departments with a wide range of research activities in scientific visualization, 3D modeling, interactive design, software and hardware graphics, robot-assisted surgery, collaborative environments, and virtual reality. He is also managing the development, support, and marketing of two products: the IBM Visualization Data Explorer and the IBM 3D Interaction Accelerator. Dr. Rossignac co-chaired the ACM Symposium on Solid Modeling '91-95, the Eurographics '94 Workshop on Graphics Hardware, and the Program Committees for CAD/Graphics '95, CSG '96, and Eurographics '96. He served as associate editor of ACM TOG, CAD, TVC, and CGF; and as guest editor for seven special issues in ACM TOG, IEEE CG&A, CAD, and IJCG. He published over 40 papers (winning four Best Paper awards), authored 13 invention disclosures, and received numerous IBM awards. He has been program director for SIAM's Activity Group on Geometric Design, and is a member of the Eurographics Board.