Representations for Rigid Solids: Theory, Methods, and Systems

ARISTIDES A. G. REQUICHA

Production Automation Project, College of Engineering and Applied Science, The University of Rochester, Rochester, New York 14627

Computer-based systems for modeling the geometry of rigid solid objects are becoming increasingly important in mechanical and civil engineering, architecture, computer graphics, computer vision, and other fields that deal with spatial phenomena. At the heart of such systems are symbol structures (representations) designating "abstract solids" (subsets of Euclidean space) that model physical solids. Representations are the sources of data for procedures which compute useful properties of objects.

The variety and uses of systems embodying representations of solids are growing rapidly, but so are the difficulties in assessing current designs, specifying the characteristics that future systems should exhibit, and designing systems to meet such specifications. This paper resolves many of these difficulties by providing a coherent view, based on sound theoretical principles, of what is presently known about the representation of solids.

The paper is divided into three parts. The first introduces a simple mathematical framework for characterizing certain important aspects of representations, for example, their semantic (geometric) integrity. The second part uses the framework to describe and compare all of the major known schemes for representing solids. The third part briefly surveys extant geometric modeling systems and then applies the concepts developed in the paper to the high-level design of a multiple-representation geometric modeling system which exhibits a level of reliability and versatility superior to that of systems currently used in industrial computer-aided design and manufacturing.

Keywords and Phrases: CAD/CAM, computational geometry, computer graphics, design and manufacturing automation, geometric modeling, representation of solids

CR Categories: 3.20, 3.26, 8.2

INTRODUCTION

Three-dimensional solid geometry plays an important role in many scientific and engineering fields, and is vital to the industries which produce discrete goods. The need for powerful computational means for dealing with geometry is widely recognized, and a variety of geometry programs and systems have been or are being developed by research laboratories, industrial users, and commercial vendors.

Useful geometry systems have four primary components (see Figure 1): (1) symbol structures which represent solid objects; (2) processes which use such representations for answering geometric questions about the objects (such as "What is the volume?"); (3) input facilities, that is, means for creating and editing object representations and for evoking processes; and (4) output facilities and representations of results. The subsystem which provides facilities for entering, storing, and modifying object representations is called a *geometric modeling system* (GMS—see the dashed box in Figure 1). Because the geometric questions that a system may be required to accommodate are strongly application-dependent and the range of potential appli-

CONTENTS

cations is broad (graphics, design analysis, manufacturing, inspection, assembly, and so on), it is useful to study applications subsystems and GMSs separately, and to identify the constraints that they impose on each other.

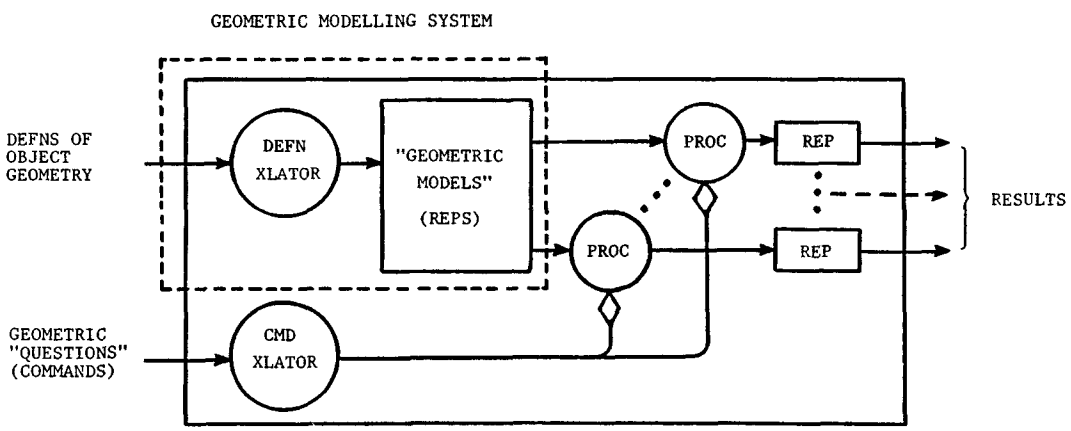
Most of the geometry systems currently used in industrial computer-aided design and manufacturing (CAD/CAM) possess relatively sophisticated human-engineered input facilities but suffer from major representational deficiencies. Specifically, representations are collections of curves and surfaces which need not correspond to single, well-defined, solid objects. This implies that such systems must rely on human assistance to supply missing information and resolve inconsistencies, and therefore cannot support *reliably* and *automatically* such applications as the calculation of moments of inertia or the generation of displays with hidden surfaces suppressed.

This paper focuses on the *representational* issues that arise in the design of GMSs. (Input techniques for effective man–machine communication are important but warrant a separate discussion.) The goals of the paper are

- to establish a framework for characterizing representation schemes for rigid solid objects;
- to describe the known schemes in terms of the framework;
- to discuss briefly how characteristics of the schemes influence the design of input and output (applications) subsystems;
- to demonstrate that the concepts developed in the paper are useful in the design of advanced modeling systems.

A thorough discussion of contemporary GMSs is beyond the scope of the paper, but

FIGURE 1. A useful geometry system.



a brief survey and references for further reading are provided.

This paper resulted from research in design and manufacturing automation in the mechanical industries [VOEL77, VOEL 74a], but it should be relevant to a variety of other scientific and engineering fields. Specifically, the discussion of representation schemes in Section 1 applies to any field that admits a well-defined mathematical modeling space, and the study of representations of solids in Section 2 is relevant to any field in which the geometry of solids is important. The paper's emphasis, however, is on the representation of the detailed geometry of objects specified or designed by humans or automata [SIMO69], and, more specifically, on "functional" rather than "aesthetic" objects.

An important topic not addressed in the paper is the representation of *classes* of "equivalent" solids. Two quite distinct examples of equivalence classes are (1) the class of mechanical parts which satisfy a particular tolerance specification and therefore are functionally equivalent and interchangeable in assembly, and (2) the class of all objects that humans recognize as (say) chairs. The methodology discussed in this paper is applicable to quantitatively defined classes, such as the former [REQU77a], but seems of limited usefulness for dealing with qualitatively defined classes, such as the latter.

1. CHARACTERIZATION OF REPRESENTATION SCHEMES

1.1 Representational Issues

Geometric algorithms do not manipulate physical solids; rather, they manipulate *data* (*symbol structures*) which *represent* solids. How are such data to be selected? Are some choices better than others?

Let us consider a specific example. Suppose we wish to represent flat-faced solid polyhedra. We take an edge-based approach because the edges of such solids are the most perceptually obvious features, and lend themselves well to line drawings. Each edge may be defined by its endpoints, and the solid may be represented by a collection of six-tuples of real numbers:

$$(X_i, Y_i, Z_i, X_j, Y_j, Z_j).$$

In addition to defining the *syntax* (notation) of representations, we must also define their *semantics* (geometric meaning) by specifying a rule which associates geometry with representations. In our polyhedral example we interpret (X_i, Y_i, Z_i) and (X_j, Y_j, Z_j) as the coordinates of two points and associate a six-tuple with the line segment whose endpoints have those coordinates. It is clear that six-tuples represent edges unambiguously, but we want to represent *solids*, not edges. The following questions come to mind immediately.

- Do edges supply enough information about a solid polyhedron to enable us to compute (fully automatically) the volume, appearance, and any other geometric property of the polyhedron?
- Does an arbitrary collection of six-tuples represent a solid polyhedron? If not, how can one ensure that geometric algorithms and systems operate on valid data?
- Are there other ways of representing a solid? (Yes.)
- How does one determine if two ostensibly different representations correspond to the same solid?
- Can some geometric properties be computed from one representation but not from another?
- Which representation is best?

These are fundamental questions that any theory of geometric modeling should address. In the sequel we shall seek mathematical and algorithmic answers to similar but more general questions. To do this, however, we must agree on what we mean *mathematically* by a solid, and we must define, formally, "representational completeness," "representation equivalence," and related concepts.

1.2 The Role of Mathematical Models

A simple example will elucidate the role of mathematical models. Consider the strings '125' and 'CXXV'. Everybody knows that such strings represent the "same thing," but what is it that they represent? We could say that they represent physical entities such as collections of pebbles, but it is more reasonable to say simply that they represent natural numbers, which are abstract

mathematical entities that model those aspects of reality relevant to counting. It is the existence of underlying abstract models—natural numbers—that allows us to study mathematically, without recourse to physical experiments, the properties of decimal and Roman representations.

We take a similar approach in our studies of geometric modeling by postulating abstract geometric entities—subsets of three-dimensional Euclidean spaces (E^3)—which model physical solids. The capabilities and limitations of Euclidean-geometry models are well understood: they have been established through centuries of experimentation.

Very few subsets of E^3 are adequate models of physical solids. The notion of "abstract solid" should capture mathematically the following properties to be useful, especially in the context of automation studies.

(1) Rigidity: An abstract solid must have an invariant configuration or shape which is independent of the solid's location and orientation.
(2) Homogeneous three dimensionality: A solid must have an interior, and a solid's boundary cannot have isolated or "dangling" portions.
(3) Finiteness: A solid must occupy a finite portion of space.
(4) Closure under rigid motions and certain Boolean operations: Rigid motions (translations and/or rotations) or operations that add or remove material (welding, machining) must, when applied to solids, produce other solids.
(5) Finite describability: There must be some finite aspect of E^3 models of solids (e.g., a finite number of "faces") to ensure that they are representable in computers.
(6) Boundary determinism: The boundary of a solid must determine unambiguously what is "inside," and hence comprises, the solid.

The mathematical implications of properties (1)–(6) are discussed in REQU77b, where it is argued that suitable models for solids are (congruence classes of) subsets of E^3 that are bounded, closed, regular, and

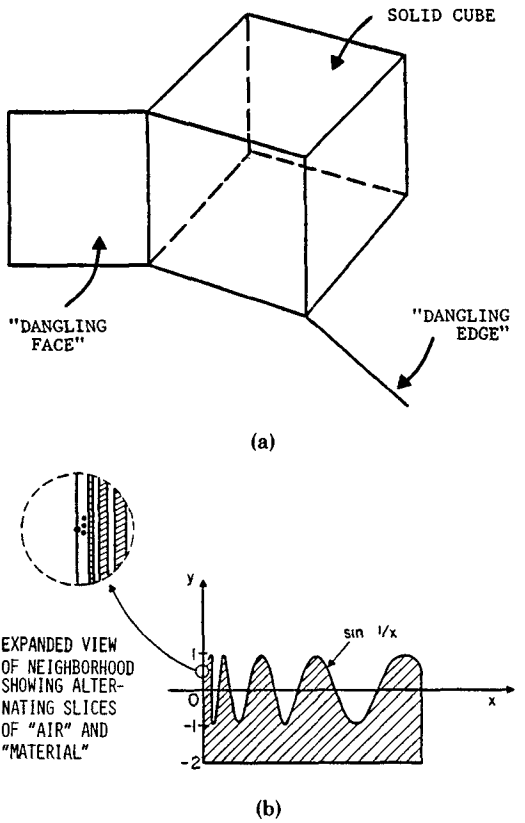


FIGURE 2. Examples of subsets of E^3 that are not r-sets. (a) Nonregular set. (b) Nonsemianalytic set.

semianalytic; we call such sets *r-sets*.[1,2] A congruence class is a collection of sets that can be obtained from one another by sequences of translations and rotations. A bounded set is one that occupies a finite portion of space. The mathematical notions of "closed regular set" and "semianalytic set" may be explained intuitively by referring to the counterexamples shown in Figure 2. The set in Figure 2a is closed because it contains its boundary, but is not a closed regular set because its boundary has dangling portions that are not adjacent to the set's interior. The set in Figure 2b is not semianalytic because its top face is ill-behaved; it oscillates infinitely fast as it approaches the left face.

R-sets are topological polyhedra

[1] The appendix contains formal definitions of mathematical terms that may be unfamiliar to readers.
[2] Because each r-set determines uniquely a congruence class, a solid can be modeled by a single r-set.

FIGURE 3. The regularized intersection of two *r*-sets is an *r*-set, but their conventional intersection need not be regular (sets shown in orthographic projection).

[AGOS76, REQU77b] and therefore may be viewed intuitively as curved polyhedra with well-behaved boundaries. Note, however, that *r*-sets need not be connected—that is, one-piece—and may have "through holes." (Some readers may prefer to think of *r*-sets as models for groups of solids in a fixed spatial relationship rather than for single physical solids.) Under the conventional (Boolean) set operations (see Figure 3) *r*-sets are *not* algebraically closed, but they are closed under the so-called *regularized* set intersection, union, and difference [REQU77c, REQU78, TILO80b], denoted $\cap^*$, $\cup^*$, $-^*$, which are modified versions of their conventional counterparts (see footnote 1).

1.3 Definitions

The following simple definitions, together with the sharp characterization of abstract solids provided by *r*-sets, allow us to use a large body of mathematical knowledge to study representations of solids and to answer questions such as those posed in Section 1.1.

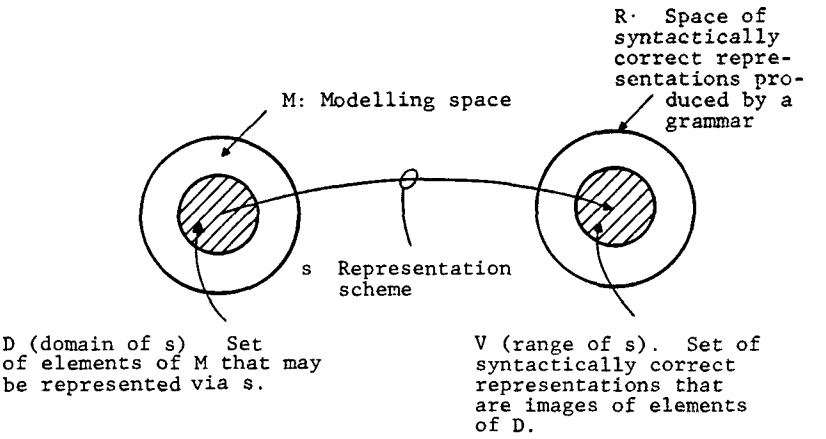
Syntactically correct representations are finite *symbol structures* constructed with symbols from an *alphabet* according to syntactical rules. The collection of all syntactically correct representations is called a *representation space R*. A representation space may be viewed as a language generated by some grammar. Note, however, that we admit representations that are not strings (they may be, for example, graphs), and place no restrictions on the generating grammars.

We define the semantics of representations by associating geometric entities with representations. Thus we postulate a *mathematical modeling space M* whose elements are abstract solids (*r*-sets), and establish a correspondence between the elements of *M* and the elements of *R* via a representation scheme; see Figure 4.

A *representation scheme* is defined formally as a relation s: $M \rightarrow R$. We denote the domain of s by D, and the range or image of D under s by V. Any representation in the range V is said to be *valid* since it is both syntactically and semantically correct (i.e., it belongs to R and has corresponding elements in the domain D).

Observe in Figure 4 that we neither assume that all objects are representable (D need not equal M) nor that all syntactically correct representations are valid (V need not equal R). We adopt this formulation largely as a matter of convenience because

FIGURE 4. Domain and range of a representation scheme.



the ranges of practical representation schemes seldom may be defined wholly syntactically by reasonable (e.g., context-free) grammars.

A representation r in V is *unambiguous* or *complete* if it corresponds to a single object (if the set $s^{-1}(r)$ is a single-element subset $\{m\}$ of D). It is *unique* if its corresponding objects do not admit representations other than r in the scheme (if $s(s^{-1}(r)) = \{r\}$).

A *representation scheme* s is unambiguous or complete if all of its valid representations are unambiguous (if the inverse relation s^{-1} is a function). It is *unique* if all of its valid representations are unique (if s is a function). Completeness neither implies nor is implied by uniqueness. An unambiguous and unique representation scheme establishes a one-to-one correspondence between its domain and range.[3]

The foregoing definitions may be summarized intuitively as follows. A representation scheme is a relation between (abstract) solids and representations. A representation is invalid if it does not correspond to any solid. A valid representation is ambiguous if it corresponds to several solids. A solid has nonunique representations if it can be represented in several ways in the scheme.

1.4 Formal Properties of Representation Schemes

This section discusses the practical implications of some important formal properties of representation schemes that follow from the earlier definitions.

1.4.1 Domain

The domain of a representation scheme characterizes the *descriptive power* of the scheme: the domain is the set of entities representable in the scheme.

1.4.2 Validity

The range of a representation scheme is the set of representations which are *valid*. Representational validity is of obvious importance in ensuring the integrity of databases, in that databases should not contain symbol structures which correspond to nonsense objects.[4] Invoking a geometric algorithm on an invalid representation may produce a system crash, obviously suspect results, or, in the worst case, results which appear to be credible but are in fact meaningless.

In the past the responsibility for ensuring the validity of representations in GMSs has rested with humans. Typically, a human would create an object representation and then check graphic displays to determine whether the representation "made sense." Not only is this procedure error-prone, but visual checking is usually impossible if representations of solids are created, in the solution of a problem, by automata (programs) rather than by humans. Program-created object representations already are used in the PADL-1 system [VOEL78] and will proliferate in the future as increasingly automated design and manufacturing systems are built.

Thus future GMSs will not be able to rely on human validation and must ensure representational validity by automatic means. This may be done by providing algorithms to check the validity of representations after they have been constructed, or by embedding validity constraints in the procedures which construct the representations. The cleanest approach is to design representation schemes in which essentially all syntactically correct representations are valid (see Section 2.5 for an example), thereby reducing validity checking to a parsing problem.

1.4.3 Completeness

The importance of completeness (as defined in Section 1.3) is best understood in the context of applications, where one must

[3] In the mathematical literature the term "representation" often is used to denote exclusively one-to-one mappings, and representations may be arbitrary mathematical entities rather than finite-length symbol structures.

[4] M. C. Escher's well-known prints provide examples of (graphical) representations of nonsense 3-D objects. Conditions under which collections of 2-D lines correspond to nonsense polyhedra have been studied in the literature on scene analysis [CLOW71, HUFF71, WALT75].

compute properties of represented entities. By definition, each unambiguous representation contains enough information to distinguish a single entity from all other entities in the modeling domain, and therefore is a sufficient source of data for evaluating any mathematically defined function of the entity. (We ignore here properties defined by functions that are not computable at all.)

Readers should not infer, however, that ambiguous representations are useless; they may be entirely adequate for specific applications. For example, the edge-based scheme for representing polyhedra introduced in Section 1.1 is ambiguous, as we see later, but is adequate for computing the convex hulls of polyhedra. Completeness is important when there is a *wide range* of applications to be supported by a practical modeling system, and is especially important when the range of applications is not known in advance.

1.4.4 Uniqueness

Representational uniqueness is important for assessing the equality of objects. The following examples illustrate the need for equality-assessment algorithms.

- Repeated representations may be culled from a database only if one can determine that two representations (in the same scheme) correspond to the same object.
- Automatic planning algorithms which search a space of alternatives may loop indefinitely if they cannot recognize previously encountered situations.
- Correct programs for numerically controlled (NC) machine tools must produce objects that equal those specified by the product designers.

Representation schemes which are *both* unambiguous and unique are highly desirable because they are one-to-one mappings. This implies that distinct representations in such schemes correspond to distinct objects, and therefore object equality may be determined by algorithms which compare object representations "syntactically." For example, it is obvious that the strings '125' and '54' represent distinct natural numbers because decimal representations are unambiguous and unique. Equality assessment in schemes which are unambiguous but not unique requires more elaborate techniques. For example, two point sets may be tested for equality by determining whether their symmetric difference is the null set.

Most representation schemes for geometric entities are nonunique for at least two reasons:

(1) Substructures in a representation may be permuted.
(2) Distinct representations may correspond to differently positioned but congruent copies of a single geometric entity.

Both permutational (case 1) and positional (case 2) nonuniqueness are conceptually trivial. Nevertheless, determining whether two structures contain the same elements may be computationally expensive when the structures are large. Positional nonuniqueness is even more pernicious because generally it is not easy to design algorithms for deciding whether two geometric entities are congruent.

1.5 Informal Properties of Representation Schemes

This section discusses several properties of representation schemes which are practically important but which cannot be formalized readily in a useful way.

1.5.1 Conciseness

Conciseness refers to the "size" of representations in a scheme. Concise representations are convenient to store and to transmit over data links, and contain relatively few redundant data. The validity of such representations is usually easy to ensure because nonredundancy implies that the entities comprising the representation are largely independent and therefore need satisfy few constraining relations.

It is important to realize, however, that selectively imposed redundancy may have practical advantages. Specifically, it may be used to detect and correct syntactic errors

in representations and to improve (often dramatically) computational efficiency by storing, rather then computing, often needed data derivable from a concise representation.

1.5.2 Ease of Creation

The ease with which (valid) representations may be created by users of modeling systems is of obvious importance, especially if the users are human. Concise representations generally are easier to create than verbose ones, because conciseness implies that fewer data need be specified and that the individual data items are largely independent.

Modeling systems based on verbose representations usually must contain powerful input subsystems to help users with the creation of representations. Such modeling systems also should possess automatic validity-ensuring mechanisms to relieve users of the burden of ensuring manually the validity of representations.

1.5.3 Efficacy in the Context of Applications

Representations of objects should be viewed for practical purposes as sources of data for algorithms. An unambiguous representation is guaranteed to be a sufficient source of data, but it may not be convenient or efficient for *all* algorithms that compute the value of some specific function. A nongeometric example: Roman representations of natural numbers are not convenient for algorithms which perform arithmetic operations.

The design of representation schemes that permit the use of "good" algorithms for evaluating useful functions is a central issue in the design of modeling systems. Good algorithms should be *correct*, *efficient*, and *robust* in the presence of numerical errors, and should exhibit other less quantifiable properties such as *extensibility*.

The correctness of an algorithm can be assessed only if precise algorithm *specifications* are available. In the realm of geometry an algorithm specification consists of the definition of a mathematical function (including its domain and range) to be evaluated by the algorithm, together with definitions of representation schemes for the domain and range. The proper specification of geometric algorithms has not received the attention it deserves.

The classical methods for analyzing and comparing algorithms (see AHO74 and recent work on geometric algorithms [SHAM78, LEE76]) ignore the issues of robustness and extensibility, and apply mainly to algorithms which operate on a *fixed* pair of domain and range representation spaces. The larger problem of comparing alternative "function-evaluation triples" consisting of algorithms *and* representation schemes is not well understood.

In summary, little is known in an abstract sense about representational efficacy. We lack not only formal means for characterizing the class of functions which may be evaluated efficiently by algorithms operating on representations in a particular scheme, but also insight into the interplay between representations and algorithms.

The experience accumulated to date in geometric modeling indicates that no single object representation scheme is uniformly "best" when many applications must be accommodated, and that (as cited earlier) redundant data often play a pivotal role in achieving efficiency. We conclude that general-purpose GMSs are likely to contain *multiple* representations of objects, in different schemes, with each representation tailored to a specific class of applications [BROW78].

1.6 Multiple Representations: Consistency and Equivalence

The presence of multiple representations introduces a strong form of redundancy in a modeling system and raises the issue of *consistency*. In essence, one must ensure that the various symbol structures which allegedly represent the same object (or objects) in different schemes do not carry contradictory information.

Figure 5 illustrates the formal notion of consistency. Representations r in R and r' in R' are consistent if there is (at least) an object m of M having representations r and r'. The dashed lines in Figure 5 indicate that m may have several representations in R and R' (when the schemes are not unique) and that other elements of M may

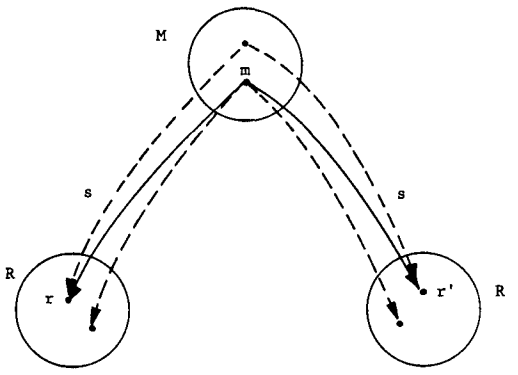


FIGURE 5. Representational consistency.

correspond to r and r' (when the schemes are ambiguous).

Consistency generally is not an equivalence relation between valid representations in schemes s and s'. We define representational equivalence as follows. Two representations r and r' are *equivalent* if they represent the same set of objects (if $s^{-1}(r) = s'^{-1}(r')$). The definition of representational equivalence may be extended as follows to cater for entire representation schemes. Two representation schemes s and s' are *equivalent* if each representation r of s possesses an equivalent r' of s', and vice versa. It follows from the definition that equivalent representation schemes have the same domain, and that two *unambiguous* representation schemes are equivalent if and only if they have the same domain.

Consistency, not equivalence, is the important concept when dealing with ambiguous representations. In geometric modeling, for example, it is important to require that (representations of) the top and front views of an object be consistent (both are ambiguous representations of solids). It would be unreasonable to require that the top and front views be equivalent: that all solids exhibiting a common top view also exhibit a common front view.

The need to enforce consistency has important implications for the design of multiple-algorithm GMSs, as we see later.

2. SCHEMES FOR REPRESENTING RIGID SOLIDS

This section focuses mainly on the known families of schemes that yield unambiguous representations of solids. It will become apparent that they are based on standard mathematical techniques for constructing sets from other sets, and therefore (1) analogous schemes may be designed for representing nonsolid entities such as surfaces, and (2) hybrid schemes may be designed by combining the various techniques. The schemes are characterized by the properties introduced in Section 1; the schemes' efficacies are discussed in the context of applications that are important in design and manufacturing automation.

2.1 Ambiguous Schemes

We begin with a discussion of a few widely used ambiguous representation schemes.

The traditional means for specifying solids—engineering drawings—are best viewed as informal means of communication among humans. Humans make liberal use of common sense to interpret drawings; sometimes they make errors of interpretation, and occasionally they correctly interpret erroneous drawings without noticing that the errors exist.

We know of no formal definition of drafting as a representation scheme. Drafting textbooks do not define in a precise manner the entities that appear in "views," and they state that the number of views, section views, details, and notes should be sufficient to avoid ambiguity. Thus the textbooks' definitions are too informal for a precise study of drafting as a representation scheme. Nevertheless, drawings are an effective means for communicating among humans and have been the major medium for geometric specification used by industry until recently. The first computer-aided drafting systems represented drawings (2-D entities) rather than 3-D solids. In such systems changes made to a view of an object are not propagated automatically to other views because the system does not know that the various views are projections of a single object.

One may move beyond drafting toward more formal schemes in the following directions.

(1) Represent solids by collections of planar projections. To construct unambiguous representation schemes

based on this notion, one must exhibit a mapping between valid drawings and solids in a domain. Defining such a mapping is a difficult and largely unresolved problem, and is related closely to scene-analysis studies which attempt to infer 3-D shape information from 2-D projections.

(2) Abandon the use of projections as representations and seek suitable collections of 3-D entities. Drafting practice suggests that "edges" be used as the primary representational entities. This yields so-called wireframe representations, which can be shown [VOEL77, MARK80] to be ambiguous even for the domain of plane-faced solids.[5] The majority of the current commercial 3-D systems are based on (curved) wireframe representations, and cannot support, for example, reliable hidden line elimination or automatic computation of section views. (If one goes further and associates "edges" with "faces," one can obtain unambiguous representations, as we see later.)

Another line of development has led to a distinctly different class of ambiguous representations. The development was motivated by the following practical problem: Given a *physical* solid—for example, a human head or a clay model of an automobile body—construct a reasonably accurate computer representation of the solid. The usual approach consists of measuring the coordinates of a large number of points lying on the boundary of the object or in the object's interior and using the set of coordinates as a representation of the object.

It is clear that such finite collections of points generally are ambiguous representations of *r*-sets. In practice one may construct an unambiguous representation of an *r*-set such that the measured points belong to the set, but the construction is not unique. An exemplary algorithm for constructing boundary representations from finite sets of boundary points is described in FUCH77. (Boundary representations are discussed in Section 2.7.)

The remainder of this section is devoted to a study of unambiguous schemes for representing solids.

2.2 Pure Primitive Instancing Schemes

An independent approach to solid-object representation has been used in the manufacturing world, mainly in the context of so-called Group Technology [GALL73]. It is based on the notion of families of objects, each member of a family being distinguishable by a few parameters. Each object family is called a *generic primitive*, and individual objects within a family are called *primitive instances*. Primitive instances are represented by fixed-length tuples of values. For example, instances of a family of prisms can be represented by tuples of the form ('PRISM', N, R, H) where 'PRISM' is a character string (the name of the family), N is the number of sides, and R and H are the radius and height of a circumscribing cylinder.

The distinguishing characteristic of pure primitive instancing schemes is the lack of means for combining instances to create structures which represent new and more complex objects. Such schemes are akin to languages defined by grammars in which it is not possible to combine words to form sentences.

In principle, pure primitive instancing schemes are unambiguous, unique, easy to validate, concise, and easy to use; they also promote standardization of components. In practice, however, such advantages obtain only for domains small enough to be covered by a small catalog of families, each having a small number of parameters.

The other main drawback of pure primitive instancing schemes is the difficulty of writing algorithms for computing properties of represented solids. A considerable amount of family-specific knowledge must be built into the algorithms, and therefore each object family must be treated as a special case, allowing no uniform overall treatment.

[5] We use "rectilinear polyhedron" to denote a plane-faced solid and "topological polyhedron" to denote any solid obtainable by an elastic deformation of a rectilinear polyhedron "Polyhedron," without a qualifier, is used when the intended meaning is clear from the context. (See AGOS76 or REQU77b for mathematical definitions.)

All of the other schemes discussed below represent solids by *structures* containing a variable number of (representations of) primitive instances (which need not be solids). The main value of pure primitive instancing schemes—their "part family" capability and associated conveniences—is disappearing rapidly because such a capability can be provided in structured schemes through generic-object representations (i.e., structured representations with uninstantiated parameters).

2.3 Spatial Occupancy Enumeration

A representation of a solid in a spatial occupancy enumeration scheme is essentially a list of spatial cells occupied by the solid. The cells, sometimes called *voxels* (an abbreviation of "volume elements"), are cubes of a fixed size and lie in a fixed spatial grid.[6] Each cell may be represented by the coordinates of a single point, such as the cell's centroid. Usually a specific spatial scanning order is imposed; the corresponding ordered sets of three-tuples are called *spatial arrays*. (An ordering is convenient for a variety of practical reasons.)

Spatial arrays are unambiguous, unique (except for positional nonuniqueness), and easy to validate, but they are potentially quite verbose. The degree of verbosity depends largely on how well the class of physical objects matches the domain representable by spatial occupancy enumeration. Thus spatial arrays may be reasonable representations in certain architectural applications where buildings are sufficiently modular [MARC74] or in tomography where irregular biological objects are modeled approximately by polyhedra. (Spatial occupancy enumeration is one of the methods for constructing an unambiguous representation from a finite-set-of-points ambiguous representation—see Section 2.1.) However, for such man-made objects as mechanical parts, which must be defined precisely but are neither boxlike nor extremely irregular, spatial arrays are too verbose for practical use as "master" (definitional) representations. (They can represent *coarse approximations* of such parts, and often can be used to improve the performance of geometric algorithms.) Some of the verbosity of pure spatial occupancy enumeration may be avoided by using hybrid schemes which combine a restricted form of constructive solid geometry (CSG), described in Section 2.5, with spatial arrays [REDD78].

Spatial occupancy enumeration shares certain interesting properties with other schemes that are based on the decomposition of solids into components with nonintersecting interiors. These properties are discussed in the context of cell decomposition schemes.

2.4 Cell Decompositions

A solid triangulation of a rectilinear polyhedron is a decomposition of the polyhedron into tetrahedra which must either be disjoint or meet precisely at a common face, edge, or vertex. Curved polyhedra may be triangulated by decomposing them into curved tetrahedra which satisfy a condition analogous to that above. Cell decompositions are a generalization of triangulations. The tetrahedra and triangles in a triangulation are replaced by 3-cells and 2-cells, respectively, in a cell decomposition. Cells may have an arbitrary number of sides.

A solid may be represented by decomposing it into cells and representing each cell in the decomposition. Spatial occupancy enumeration schemes are a particular case in which all of the cells in the scheme must be cubical and lie in a fixed grid.

Cell decompositions are unambiguous but nonunique. Validity is computationally expensive to establish because a 3-D version of the boundary validity problem discussed in Section 2.7 must be solved. Cell decompositions (of human- or machine-made objects) generally are neither concise nor easy to create. Valid cell decompositions of *curved* solids are hard to construct by humans, and algorithms for constructing them from others are a topic of current research [ARNO79].

Cell-decomposition representations provide convenient means for computing certain topological properties of objects. There are known algorithms, for example, to de-

[6] Other decompositions (e.g., tetrahedral) of E^3 may be used, but for simplicity we discuss only the cubical grid because the type of grid does not affect the basic properties of the schemes.

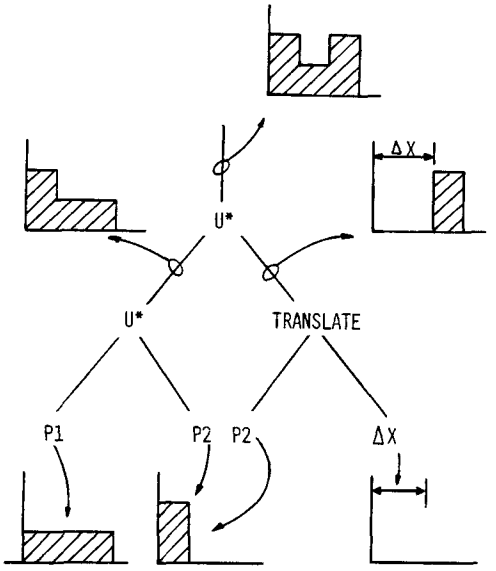


FIGURE 6. A CSG tree and the solids represented by its subtrees (solids are shown in orthographic projection).

termine whether a solid is "one-piece" (i.e., connected), and whether it has "voids" or "holes" [REQU77b, AGOS76].

Cell decompositions, spatial arrays (Section 2.3), and certain restricted forms of CSG (Section 2.5) represent solids as unions of components whose interiors are pairwise disjoint. Disjointness may be exploited in useful ways. For example:

(1) An object is null if and only if all of its components are empty.
(2) Various integral properties of solids—volume and moments of inertia, for example—may be computed as sums of the separately evaluated components' contributions [COHE79, LEE80].

Finally, it is important to remark that cell decompositions are the representations used in 3-D (solid) finite element methods for the numerical solution of differential equations.

2.5 Constructive Solid Geometry

Constructive solid geometry (CSG) connotes a family of schemes for representing rigid solids as Boolean constructions or combinations of solid components via the regularized set operators defined in the Appendix [REQU77c]. CSG and boundary representations (described in Section 2.7) are the best understood and currently most important representation schemes for solids, and therefore are treated in this survey more thoroughly than the other schemes.

2.5.1 CSG Trees

CSG representations are (ordered) binary trees. Nonterminal nodes represent operators, which may be either rigid motions or regularized union, intersection, or difference; terminal nodes are either primitive leaves which represent subsets of E^3, or transformation leaves which contain the defining arguments of rigid motions.[7] CSG trees may be defined by the following

⟨CSG tree⟩ ::= ⟨primitive leaf⟩ | ⟨CSG tree⟩ ⟨set-operator node⟩ ⟨CSG tree⟩ | ⟨CSG tree⟩ ⟨motion node⟩ ⟨motion arguments⟩.

The semantics of CSG-tree representations is clear (see Figure 6): Each subtree that is not a transformation leaf represents a set resulting from applying the indicated motional/combinational operators to the sets represented by the primitive leaves.

We consider only CSG schemes whose primitives either are r-sets or fail to be r-sets only because they are unbounded. Schemes whose primitives are bounded (i.e., are r-sets) are called "CSG based on bounded primitives," or simply "CSG" when no confusion is likely to arise, while schemes possessing unbounded primitives are called "CSG based on general half-spaces." (The distinction is important, as we see below.)

2.5.2 Properties of CSG Schemes

CSG schemes are unambiguous but not unique. The domain of a CSG scheme depends on the half-spaces which underlie its set of primitive solids (loosely, on the available surfaces), and on the available motional and combinational operators. Thus the two schemes whose primitives are de-

[7] In many CSG schemes subtrees may be shared, and therefore the representations are graphs rather than trees. Data sharing is unimportant for the purposes of this paper and will be ignored

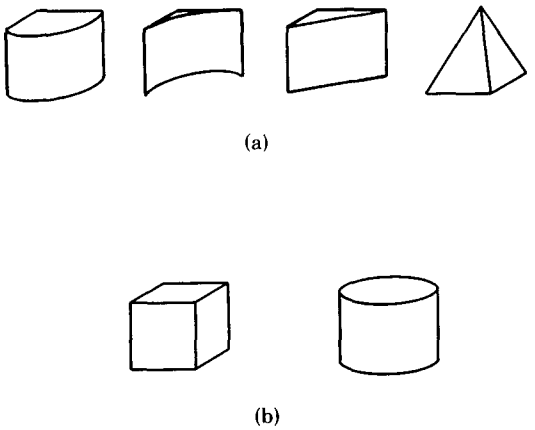


FIGURE 7. Two CSG schemes having different primitives but the same domain

pictured in Figures 7a and 7b have the same domain if both schemes have general motional and combinational operators.

When the primitive solids of a CSG scheme are bounded and hence are *r*-sets, the algebraic properties of *r*-sets guarantee that *any* CSG tree is a *valid* representation of an *r*-set if the primitive leaves are valid. (This would not be true if CSG trees were allowed to contain the regularized complement because it destroys boundedness.) Because primitive leaf validity usually may be assessed easily, the validity of CSG based on bounded primitives may be ensured essentially at the syntactical level. That is, in a CSG language (e.g., PADL [VOEL78]) every syntactically correct object description with correctly instanced primitives represents unambiguously some solid. CSG trees with uninstantiated parameters can be used to represent *generic objects* (also called procedure-, macro-, or parametric-objects, or object schemata). The validity of such generic-object representations also may be ensured easily.

CSG trees in schemes based on unbounded primitives may represent unbounded sets and therefore be invalid. Boundedness of a composition generally is difficult to verify. The only viable approach known to us involves re-representing the set via its boundary and testing for face boundedness. This is computationally expensive and the algorithms required are nontrivial.

It is important to realize that the "guaranteed validity" of CSG schemes based on solid primitives applies only to schemes in which the combinational operators are *general* regularized set operators which may be applied to *any* objects in the domain of the representation scheme. CSG-like schemes in which the operators are not general have validity properties akin to those of cell decompositions (Section 2.4).

CSG schemes whose primitives are well matched to the domain of objects to be represented are very concise. Schemes based on bounded primitives usually are more concise than those based on general half-spaces because the latter usually require that more (and lower level) primitives be used. For example, the object represented via three primitive "blocks" in Figure 6 would require at least nine instances of a planar half-space primitive.

The limited experience accrued with experimental CSG-based systems indicates that humans can easily create CSG representations of certain classes of objects, such as unsculptured (functional) mechanical parts.

CSG representations are not efficient sources of geometric data for producing line drawings of objects, and certain types of graphic interactions (e.g., "pick an edge") are difficult to support directly from CSG representations. However, simple and extensible algorithms are known for other interactions (e.g., "pick a face"), for generating shaded displays, and for computing integral properties (e.g., volume) of objects [GOLD71, GOLD79, LEE80]. Some of these algorithms are slow in sequential machines but are simple and inherently parallel, and therefore are promising candidates for hardware implementation through VLSI. It appears that CSG representations will be of considerable importance for manufacturing automation, such as in the study of rough machining operations.

2.5.3 Restricted Forms of CSG

A variety of restricted forms of CSG have been devised. Spatial arrays (Section 2.3), cell decompositions (Section 2.4), and certain schemes used in architecture [MITC78] may be viewed as particular cases of CSG possessing a single "glue" operator. ("Glue" is a restricted form of union that applies only to sets with disjoint interiors.)

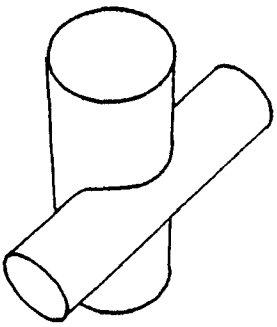


FIGURE 8. A set that cannot be represented easily in CSG without a general union operator

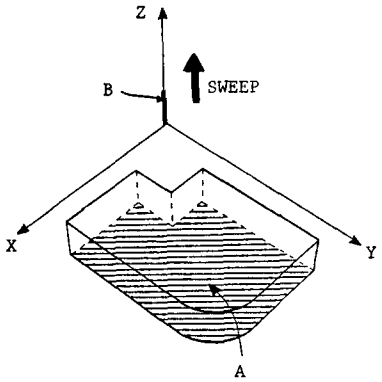


FIGURE 9. Translational sweeping.

CSG schemes possessing a single but general union operator and a single primitive have been used in computer graphics and computer vision [HERB78, BADL79]. Schemes possessing a single associative operator do not need to represent the operator explicitly, and may collapse the CSG trees into lists or sets of primitive leaves.

CSG-like schemes whose operators are not applicable to *all* pairs of objects in the domain (e.g., a glue rather than a general union operator) have been used more extensively than CSG proper. In practice, restrictions are placed on operators to facilitate boundary evaluation—the computation of the objects' boundaries. However such restrictions have the following unpleasant consequences.

(1) User inconvenience: The object shown in Figure 8 is the union of two cylinders but is difficult to represent in a scheme possessing only glue and intersection operators.
(2) Lack of closure: The results of applying operators to representable objects generally may not be used as inputs for further operations [TILO80b].
(3) Difficulty of ensuring validity: Testing the validity of representations involves procedures whose complexity is comparable to those of general boundary evaluation procedures.
(4) Inadequacy for certain applications: For example, the analysis of spatial interferences requires a general intersection operator.

2.6 Sweep Representations

The basic notion embodied in sweeping schemes is very simple: A set moving through space may sweep a "volume" (a solid) that may be represented by the "moving object" plus the "trajectory." Readers are warned, however, that sweeping is the least understood, in a theoretical sense, of all the schemes discussed in this paper. (For example, general validity conditions for sweep representations are unknown.)

2.6.1 Translational and Rotational Sweeping

Consider a 2-D set A lying in a plane and a line segment B perpendicular to the plane of A and having an endpoint on the plane. Let S be the solid swept by A when it is translated, parallel to its plane, along the trajectory B (see Figure 9).[8] Obviously S may be represented by representing A and B. Because the representation of B is trivial, the problem of representing a 3-D solid S is reduced to that of representing a 2-D set A.

The scheme just described is called *translational sweeping. Rotational* sweeping schemes (Figure 10) may be defined in an analogous manner. The properties of translational and rotational sweeping schemes are determined largely by the specific schemes used to represent 2-D sets. Most schemes described to date represent 2-D sets via their bounding edges, which

[8] Mathematically S is the Cartesian product $A \times B$. General sweeping schemes may be designed by selecting suitable mappings of set products into E^3; that is, a set S may be represented by a pair of sets (A, B) such that $S = f(A \times B)$, where f is a mapping $f : A \times B \rightarrow E^3$.

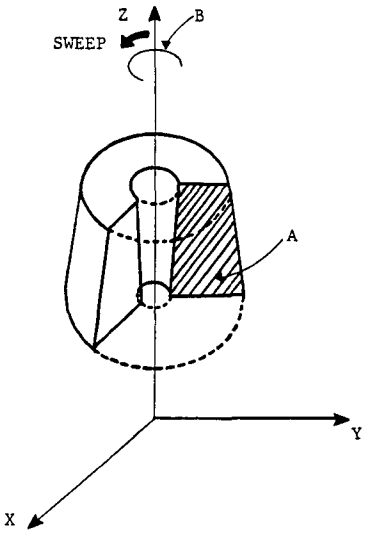


FIGURE 10. Rotational sweeping.

leads to convenient interfaces for traditionally trained draftsmen.

Translational and rotational sweeping schemes are unambiguous but not unique, and their domains are limited to objects with translational or rotational symmetry. Thus rotational sweeping is used by many GMSs that deal exclusively with turned (lathe) parts, and translational sweeping is used in systems for describing flat sheet metal parts. Both types of sweeping are used as a means of entering object (or subobject) descriptions in GMSs based on boundary representations.

2 6.2 *Volumes Swept by Motions of a Solid*

Sweep representations provide natural means for describing the solid volume swept by a rigid solid moving through space. Such swept volumes are important in the context of manufacturing automation, as the following examples indicate.

(1) Material removal: The effect of a machining operation is to remove from a workpiece the solid volume swept by the cutter when it moves along a specified trajectory [VOEL74a, VOEL77].
(2) Dynamic interference: A solid S_1 moving through space collides with another solid S_2 if the volume swept by S_1 intersects S_2 [BOYS79a].

The completeness of solid sweeping schemes implies that a cutter trajectory specified by a part program written in a language such as APT [IITR67], together with a representation of the cutter or its "envelope" and a representation of the initial workpiece, is an unambiguous representation of the workpiece which results from a specified machining operation. Such a representation might be termed "operational" or "procedural" since it defines a solid by describing a method for its manufacture. A major disadvantage of such a representation is the lack of known algorithms for computing properties of represented solids.

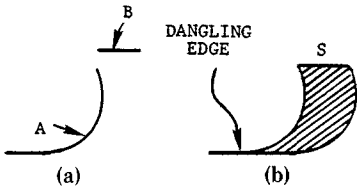


FIGURE 11. General sweeping may produce nonregular sets.

2.6.3 *General Sweeping*

Very little is known about general sweep representations. Figure 11 shows that sweeping a curve along another curve may produce a set which is not homogeneously 2-D. (Note the dangling edge in Figure 11b; analogous examples may be constructed in 3-D.) Mathematical conditions for ensuring that the resulting set is an r-set are unknown.

The examples of sweeping discussed in Sections 2.6.1 and 2.6.2 and the example depicted in Figure 11 amount to moving a rigid object (not necessarily a solid) through space. General sweeping may also involve the motion of a nonrigid object which undergoes deformations as it travels through space, as in the proposals by Binford and others for modeling systems based on so-called generalized cylinders or generalized cones [AGIN76, BROO79]. Generalized cylinders appear to be quite useful for describing objects such as airplanes, animals, and dolls which may be decomposed naturally into a few tubelike components.

General sweep representations also are used extensively in a recent proposal for an

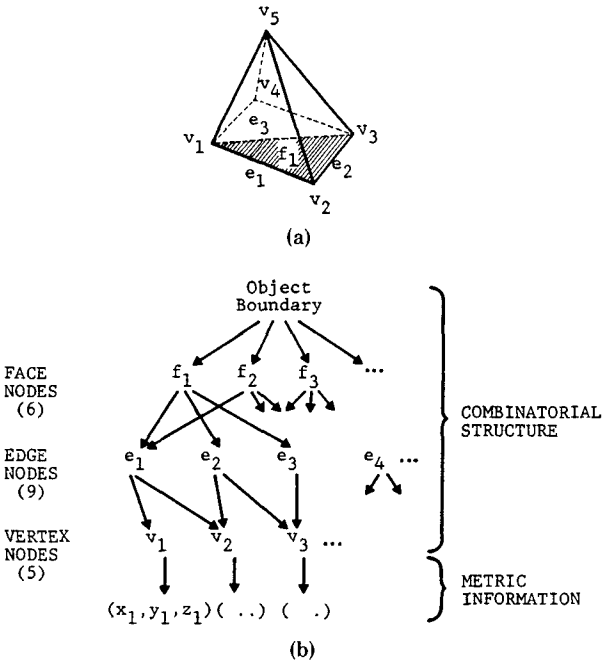


FIGURE 12. A boundary representation for a rectangular pyramid.

ANSI standard representation of solid objects [ANSI78].

2.7 Boundary Representations

Boundary representations of solids are familiar to most computer scientists because of their use in computer graphics. A solid is represented by segmenting its boundary into a finite number of bounded subsets usually called "faces" or "patches," and representing each face by (for example) its bounding edges and vertices.

Figure 12 provides an example of a boundary representation for a rectangular pyramid. The representation is a directed graph containing object, face, edge, and vertex nodes. The scheme illustrated in the figure is based on boundary *triangulation*, that is, the segmentation of an object's boundary into nonoverlapping triangles. It is clear that any rectilinear polyhedron may be represented in such a scheme. (Boundary triangulation schemes are used extensively in finite element analysis.)

The remainder of this section discusses issues that arise in the design of boundary representation schemes, and the properties of such schemes.

2.7.1 Design Issues

Most people have the same intuitive notion of "face" for rectilinear polyhedra. Observe, however, that the faces in the representation of Figure 12 do not agree with the intuitive notion because the pyramid's rectangular bottom has been segmented into two triangles. Furthermore, it is not intuitively clear what the faces of *curved* objects are; see Figure 13. The examples of Figures 12 and 13 show that faces are representational artifacts which may or may not correspond to one's intuitive notion of face; in any event the entity "face" must be defined by the designers of boundary representation schemes. We propose certain characteristics which a reasonable definition should possess.

First we assume that each boundary representation scheme has a finite number of generic *primitive surfaces* S_i. (Here we ignore certain mathematical details that are discussed in REQU80.) Faces should satisfy the following conditions.

(1) A face of an object is a subset of the object's boundary.
(2) The union of all of the faces of an object equals the object's boundary.

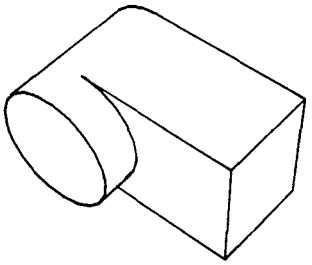


FIGURE 13 What are the faces?

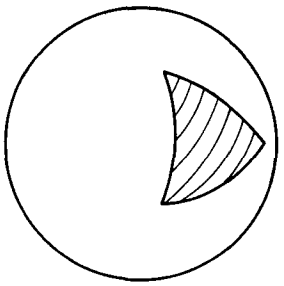


FIGURE 14. A face lying in a spherical surface.

(3) Each face is a subset of some primitive surface S_i.
(4) A face must have area and must not have dangling edges or isolated points. (Technically it must be a homogeneously 2-D topological polyhedron.)

In particular schemes, faces may be required not to overlap one another, to be connected, or to have no holes; none of these requirements is essential.

Faces must be represented unambiguously if a boundary representation scheme is to be unambiguous. Planar faces may be represented by their bounding edges, as in Figure 12, but nonplanar faces require that their "host" primitive surface also be represented, typically by primitive instancing. Figure 14 shows that additional information may be needed to represent unambiguously faces which lie in certain surfaces. (To distinguish between the shaded spherical triangle of Figure 14 and the remainder of the sphere, typically one orients the bounding edges of the face according to some convention.)

Specialized techniques exist for representing the doubly curved ("sculptured") faces usually called "patches"; see BARN74 or NOWA80 for entries into the extensive literature on the subject.

Face representation via edges is the 2-D analog of solid representation via faces, and therefore the entity "edge" must be defined mathematically and appropriate edge-representation schemes must be designed.

The various extant boundary representation schemes differ in the redundant data they provide to speed up specific algorithms. A simple example: Although surface normals are not required to define unambiguously a solid's boundary, most schemes store with each face the direction of an outward normal to it because such information is easy to store and is useful for generating graphic displays.

Other redundant information takes the form of adjacency relations between faces, edges, and vertices. Such information sometimes is stored to avoid expensive searches that certain algorithms require. One of the most elaborate schemes containing explicit adjacency information is that described in BAUM74. The trade-off between the complexity of the algorithms which construct and maintain elaborate boundary representation structures and the efficiency of algorithms which use such representations (e.g., to produce displays with hidden lines removed) is not well understood.

In summary, a designer of a boundary representation scheme must address the following questions:

(1) What is a face?
(2) How are faces represented?
(3) What is an edge?
(4) How are edges represented?
(5) What redundant data are maintained?

Each question has important implications, but a thorough discussion of such matters would take us too far afield.

2.7.2 *Formal Properties*

Boundary representation schemes are potentially capable of covering domains as rich as those of cell-decomposition or CSG schemes. Indeed, given a CSG scheme with primitive half-spaces $\{H_i\}$, it is always possible to design a boundary representation scheme with the same domain by using as primitive surfaces the boundaries of the H_i.

Boundary representation schemes are

unambiguous if faces are represented unambiguously, but generally they are not unique. Completeness follows from deep mathematical theorems which ensure that an r-set is defined unambiguously by its boundary [REQU77b]. It is worth remarking that general (non-r-set) subsets of E^3 are not defined unambiguously by their boundaries.

The validity of boundary representations raises interesting issues which we discuss in the remainder of this section. To study validity, one proceeds "bottom-up" by investigating conditions for vertex nodes to represent points of E^3, for edge subgraphs (i.e., edge nodes plus their corresponding vertex nodes) to represent line segments, and so on.

We present the validity conditions for the triangulation scheme illustrated in Figure 12. (Such conditions follow directly from elementary algebraic topology [REQU77b, AGOS76].) We distinguish two types of conditions required for validity, combinatorial and metric. The following are the combinatorial conditions.

(1) Each face must have precisely three edges.
(2) Each edge must have precisely two vertices.
(3) Each edge must belong to an even number of faces.
(4) Each vertex in a face must belong precisely to two of the face's edges.

Conditions 1 and 2 are obvious. Conditions 3 and 4 ensure, respectively, that the surface "closes" and that each face's edges form "loops." It is not difficult to write efficient algorithms for testing the combinatorial conditions above, or to embed such conditions in the procedures which construct and update the representations. The so-called Euler operators, which are used in some of the GMSs based on boundary representations [BAUM74, BRAI78b, EAST79], appear to play the role of ensuring that combinatorial conditions analogous to those above are met. (An exemplary Euler operator: 'SplitEdge' divides an edge into two by introducing a new vertex in the interior of the original edge.)

Combinatorial conditions, however, do not suffice to ensure validity; the following metric conditions also must be satisfied.

(5) Each three-tuple of vertex coordinates must represent a distinct point of E^3.
(6) Edges must either be disjoint or intersect at a common vertex.
(7) Faces must either be disjoint or intersect at a common edge or vertex.

Conditions 6 and 7 are computationally unpleasant because they require face/face comparisons and other expensive calculations.

Combinatorial structure and metric information have been called, respectively, "topology" and "geometry" by other authors [BAUM74, BRAI78a, EAST77], who advocate the separation of topology from geometry in GMSs. Such a separation is sometimes convenient in that one may (for example) represent a translated rectilinear polyhedron simply by translating all of its vertex coordinates without altering its combinatorial structure. It should be obvious, however, that combinatorial structure and metric information are not independent. For example, by changing the coordinates associated with vertex node v_5 of Figure 12 one can easily render the representation invalid; it suffices to force v_5 to lie in the plane of the pyramid's base.

Validity conditions for boundary representation schemes which cater for curved solids also may be established with the aid of elementary algebraic topology, although curvature introduces a variety of interesting and delicate issues [REQU80].

Validity conditions for solid triangulations and cell decompositions are 3-D generalizations of the validity conditions discussed above. It is worth noting that the verification of finite element meshes essentially amounts to checking the validity of cell decompositions [PREI79].

2.7.3 *Informal Properties*

Boundary representations are verbose, an order of magnitude longer than corresponding CSG representations. The sheer volume of required data and the delicate validity conditions make it difficult for humans to construct, without computer assistance, boundary representations of even moderately complex objects. Essentially all of the

TABLE 1. SUMMARY OF REPRESENTATION SCHEME PROPERTIES AND APPLICABILITY

		PROPERTIES						APPLICATIONS											
		DOMAIN	VALIDITY	COMPLETENESS	UNIQUENESS	CONCISENESS	EASE OF CREATION	NULL OBJECT	OBJECT EQUALITY	LINE DRAWINGS	GRAPHIC INTERACTION	SHADED DRAWINGS	MASS PROPERTIES	HOMOLOGICAL PROPERTIES	FINITE ELEMENT ANALYSIS	INTERFERENCE ANALYSIS	ROUGH MACHINING	FINISH MACHINING	NC TAPE VERIFICATION
PRIMITIVE INSTANCING			G	G	G	G	G	G	G	G		?	G	G	?		G	G	G
SPATIAL ENUMERATION			G	G	G			G	G	?	?	?	G	G	G	G			
CELL DECOMPOSITION		G		G				G		?	?	?	P	G	P	?	?	?	?
CSG	GENERAL 1/2 SPACES	G		G				?	?		P	?	?			P	P	?	P
CSG	BOUNDED PRIMITIVES	G	G	G		G	G	?	?		P	?	?			P	P	?	P
SWEEP	TRANSLATIONAL & ROTATIONAL		G	G	?	G	G	G	?	G	G	?	G	?	P				
SWEEP	GENERAL	G		G				G											
BOUNDARY		G		G	?			G	?	G	G	P	?	P				P	

KEY G = GOOD P = PROMISING ? = UNCLEAR BLANK= UNPROMISING OR POOR

known geometric modeling systems that rely on boundary representations attempt to provide such assistance, as noted in Section 3.1, via a user-oriented input language.

The main virtue of boundary representations lies in the ready availability of representations for faces, edges, and the relations between them; these data are important for generating line drawings and graphic displays, for supporting graphic interaction, and for other purposes as well. Thus it is not surprising that boundary representations have been used extensively in computer graphics.

We noted in Section 1.5 that object equality may be detected in nonunique schemes by testing for null symmetric differences. Because boundaries are represented as *unions* of faces, a boundary is null (and hence the object is null) if and only if it has no faces. Therefore testing for nullity is trivial in boundary representation schemes. (However, computing the symmetric difference of two objects represented by their boundaries is *not* trivial.)

The majority of boundary schemes use representations closely related to cell decompositions *of object boundaries.* There are known algorithms to compute topological properties (e.g., connectedness) of objects from such representations [REQU77b, AGOS76].

2.8 Summary of the Characteristics of Schemes for Representing Solids

Table 1 attempts to summarize and extend the previous discussion of unambiguous representation schemes for solids. Readers are warned that the table involves some oversimplification, and that there is almost no definitive information on the suitability of the different schemes for applications; the table's entries on applicability are best viewed as the current opinions of the author. The table should be self-explanatory except for the following two entries.

(1) Null object: Given a representation of a solid, is the solid null, that is, the empty set?

(2) Homological properties: These are the connectedness, number of "holes," and similar topological properties that can be inferred from the homology groups of an object [REQU77b, AGOS76].

2.9 Hybrid Schemes

Hybrid, or *nonhomogeneous,* representa-

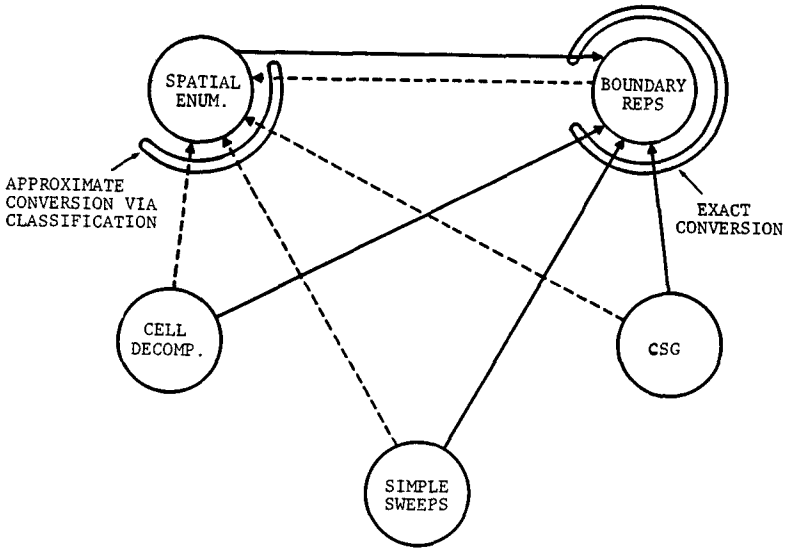


FIGURE 15 Representation conversion current state of knowledge.

tion schemes may be designed by combining the approaches discussed in the preceding sections. Three examples follow.

(1) CSG/boundary hybrid: Representations are CSG-like trees whose leaves are either tuples that represent primitive solids or boundary representations of nonprimitive solids.
(2) CSG/solid-sweep hybrid: Representations are CSG-like trees whose leaves may be solid-sweep representations of nonprimitives.
(3) CSG/general-sweep hybrid: Representations are CSG-like trees whose leaves are general-sweep representations (e.g., generalized cylinders).

Scheme 1 is used as the basis for the input language of some GMSs (see Section 3.1). Scheme 2 is useful for the verification of programs for numerically controlled machine tools. Scheme 3 is the major representation scheme used in ACRONYM [BROO79], a modeling system designed for applications in computer vision and manipulation.

Hybrid schemes pose a serious practical problem: It is difficult to design algorithms for computing properties of objects represented in such schemes. For example, an algorithm for computing the volume of objects represented in scheme 1 would have to be able to deal with pure CSG representations, pure boundary representations, and combinations of the two. Usually it is preferable to *homogenize* hybrid representations and to design algorithms for a single (pure) scheme.

2.10 Conversion Between Representations

The previous sections show that none of the extant schemes for representing solids have properties that are uniformly better than those of other schemes. The ability to *convert* representations in a scheme into corresponding representations in other schemes is therefore of great practical importance for well-engineered modeling systems. (The homogenization of representations in hybrid schemes is a particular case of representation conversion.)

We distinguish between *consistent* (theoretically *exact*) and *approximate* conversions. Loosely, a consistent conversion produces a representation of the original solid, while an approximate conversion produces a representation of a solid which approximates the original one. Consistent conversions between unambiguous representations are *lossless* (theoretically invertible). (Formal definitions may be found in REQU80.)

Figure 15 illustrates the current state of knowledge on algorithmic conversion between representations of solids. There

are two main reasons for the lack of algorithms for bidirectional exact conversion: (1) schemes such as spatial enumeration or simple (translational or rotational) sweeping generally have domains which are smaller than those of CSG, boundary representations, or cell decompositions; and (2) algorithms for effecting certain conversions are not known at present.

Cell decompositions (and therefore also spatial enumerations) may be converted exactly into boundary representations via simple algorithms [AGOS76] which select those cell faces that belong to only one cell in the decomposition. Algorithms for converting exactly translational and rotational sweeps into boundary representations also are simple because there is a straightforward correspondence between the bounding edges of the 2-D "moving set" and the faces in a boundary representation of the swept solid. Exact conversion of CSG to boundary representations ("boundary evaluation" in University of Rochester jargon) requires nontrivial algorithms. Boundary evaluation algorithms used in the PADL-1 processor [VOEL78] are based on the notion of "set membership classification" discussed in TILO80a and VOEL80. Work related to "inverse boundary evaluation," that is, conversion of boundary representations into CSG, is described in POPP75.

Approximate conversion into spatial enumeration may be accomplished by algorithms which compute the so-called point membership classification function [TILO80a, REQU77c], that is, which determine whether a point is in the interior, on the boundary, or in the outside of a solid. (For restricted domains of rectilinear polyhedra the conversions are exact rather than approximate.)

We see below that the relative paucity of known conversion algorithms poses significant constraints on the GMSs that can be built today.

3. GEOMETRIC MODELING SYSTEMS

A *geometric modeling system* or *geometric modeler* is a computer-based system for creating, editing, and accessing representations of solid objects (see Figure 1). Display generators and other processes that are intimately connected with geometric representations sometimes are considered components of a GMS, but the majority of the processes which *use* geometry (e.g., to compute mass properties of objects) are *application programs*, external to a GMS.

Representations in a GMS are created and modified through an *input language*. Input languages may range from simple interactive-graphic command languages to elaborate block-structured languages similar to conventional (algorithmic) programming languages. The essential requirement is that they contain facilities for instancing primitives and for creating structures in one or more of the representation schemes discussed in Section 2.

A sequence of input-language statements is itself an object representation. But it is important to distinguish between such representations and a GMS's *internal representations* because it is the latter that are accessible to applications programs and therefore determine, to a large extent, the usefulness of the GMS.

The remainder of this section surveys briefly extant GMSs and discusses the design of advanced modelers.

3.1 A Brief Survey of Extant Systems

Table 2 provides a summary characterization of the implemented GMSs known to us that attempt to cover domains appropriate for engineering applications. The table does not include systems which are (1) based on wireframe representations, (2) based on spatial enumeration, or (3) devoted primarily to the representation of sculptured surfaces.

Each system is characterized by a primary (internal) representation scheme. The domain of the primary scheme is defined in terms of half-spaces (loosely, surfaces). (Specifically, the domain is the regularized Boolean class—see the appendix—defined by the half-spaces listed in the table.) "Other consistent representations" denotes representations that are maintained by the system, are accessible to application programs, and are *guaranteed* to be consistent (Section 1.6) with the system's primary representations. (Ambiguous representations are ignored in the table.) Each system has input-language facilities

TABLE 2. GEOMETRIC MODELING SYSTEMS

SYSTEM NAME OR ORIGINATOR	PRIMARY REP SCHEME	DOMAIN (DEFINED VIA 1/2 SPACES)	OTHER CONSISTENT REPS	INPUT LANGUAGE BASED ON	REFERENCES
"SHAPES"	CSG: GENERAL 1/2 SPACES	PL & QUAD			LANI73, LANI79
"TIPS"	CSG: GENERAL 1/2 SPACES	"ARBITRARY"	APPROX. ENUM.		OKIN73, OKIN78
"GDP"	CSG: BOUNDED PRIMITIVES	PL & QUAD	APPROX. B-REP		GROS76, WESL80A-B
"GMSOLID"	CSG: BOUNDED PRIMITIVES	PL & QUAD	B-REP		BOYS78, BOYS79B
"PADL-1"	CSG: BOUNDED PRIMITIVES	PL & CYL (⊥)	B-REP		VOEL74B, VOEL78
"SYNTHAVISION"	CSG: BOUNDED PRIMITIVES	PL & QUAD			GOLD71, GOLD79
BORKIN (UM)	B-REPS	PL		CSG, SWEEP*	BORK78
"BUILD" (OLD)	B-REPS	PL & CYL		CSG*, SWEEP*	BRAI73, BRAI75A
"BUILD" (NEW)	B-REPS	PL & CYL		CSG	BRAI78A
"CADD"	B-REPS	PL, QUAD, SS		SWEEP*	
"COMPAC"	B-REPS	PL & QUAD		CSG, SWEEP*	SPUR76, SPUR78
"EUCLID"	B-REPS	PL		CSG	BERN75
"EUKLID"	B-REPS	PL		CSG	ENGE73
"GEOMAP"	B-REPS	PL	CELL DEC.	CSG	HOSA74, HOSA77
"GEOMED"	B-REPS	PL		CSG	BAUM74
"GIPSY"	B-REPS	PL & QUAD		CSG*	SCHU76
"GLIDE"	B-REPS	PL		CSG, SWEEP*	EAST77
PARENT (OSU)	B-REPS	PL		CSG	PARE77
"PROREN2"	B-REPS	PL & QUAD		CSG, SWEEP*	SEIF78
"ROMULUS"	B-REPS	PL & CYL		SWEEP*	
YESSIOS (OSU)	B-REPS	PL		CSG	YESS78
"3DFORM"	B-REPS	PL		CSG	WOOD79
"ACRONYM"	HYBRID CSG/SWEEP	PL & QUAD			BROO79

KEY: PL = PLANE
CYL = CYLINDER
QUAD = QUADRIC
SS = SCULPTURED SURFACE
⊥ = ORTHOGONAL POSITIONING

CSG* = CSG WITH NON-GENERAL OPS.
SWEEP* = TRANSLATIONAL AND/OR ROTATIONAL SWEEP

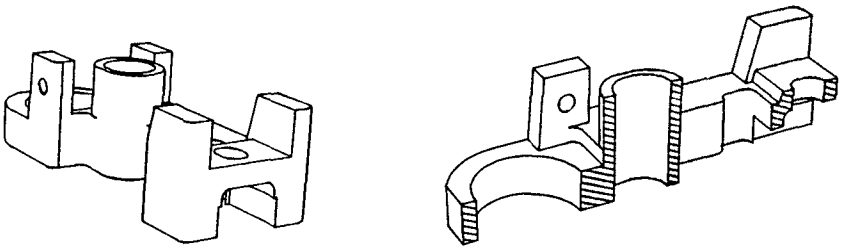
based on its primary representation scheme, but some systems have additional input-language facilities. For example, many systems having a primary boundary representation have additional CSG-based input facilities which provide a means for creating boundary representations of the union and (closed) difference of other objects. The "input language" column of the table lists *only* those input language capabilities which are *not* based on the system's primary representation.

The table is derived from information available in the cited literature. It may not reflect the current state of some of the systems because many of the systems are in continuous evolution, and because systems descriptions in the literature often ignore central issues such as the following.

- Are CSG-like operators regularized and general?
- Are representational validity and completeness guaranteed by the system?
- Is the input language sufficiently powerful to cover the entire domain of the primary representation scheme (i.e., the regularized Boolean class defined by the half-spaces listed in the table)?

Of the systems listed only four are commercially available: Synthavision (MAGI, Elmsford, N.Y.), CADD (McAuto, St. Louis, Mo.), Euklid (Fides, Zurich, Switzerland), and Romulus (Shape Data, Cambridge, England). Only CADD seems to have been used extensively for production rather than experimental applications.

Geometric modeling technology is poised for takeoff, with industrial awareness and acceptance growing rapidly. At a seminar [CART79] organized recently by an international organization (Computer Aided Manufacturing-International, Inc., of Arlington, Texas) the capabilities of several GMSs were demonstrated by using them to model a "typical" mechanical part (see Figure 16) supplied by the German firm Messerschmitt-Bolkow-Blohm (MBB). The exercise showed clearly that several of the experimental GMSs can accommodate "real" mechanical parts and can support a



(a) (b)

FIGURE 16. Automatically computed perspective and section views of an industrial part. (Displays generated by the PADL-1 system.)

range of applications. (The majority of the systems automatically produced drawings in a variety of styles and computed mass properties automatically.) G. Armstrong, of the University of Leeds, England, provided a dramatic example of GMS power by exhibiting a copy of the MBB part produced on an NC milling machine that was driven by instructions derived *automatically* from the part's definition in the PADL-1 language.

Production-grade versions of some of the experimental GMSs listed in Table 2 are currently being designed and implemented, and several second-generation modelers based on different system-organization principles should become commercially available in the early 1980s. In the next section we discuss, through an example, the major issues that arise in designing such systems.

3.2 A Design for an Advanced Modeling System

Useful geometry systems should (1) solve *reliably* (correctly) and *automatically* a well-defined set of problems, (2) be extensible with respect to geometric coverage and with respect to the range of applications supported *efficiently*, and (3) exhibit a variety of other characteristics which range from being congenial to users to being transportable. The technical implications of (3) are not well understood in a formal sense; they imply good design and implementation practices.

The discussions in the preceding sections show that characteristics (1) and (2) have the following technical implications. Reliability requires that all representations in a GMS be valid and that all algorithms be correct. Geometric extensibility implies that the internal representation schemes of the GMS must be potentially capable of covering a large domain. To ensure that representations in a GMS have sufficient information to support new geometric applications, at least one of the major representation schemes must be unambiguous. Efficient support of a variety of applications requires the existence of multiple representations, which must be kept consistent.

Figure 17 depicts a high-level architecture for a GMS which has the characteristics just discussed, and which can be built by stretching safely the current technology [BROW78]. Dashed lines in the figure indicate means to *edit* existing geometry, rather than to create new geometry.

The rationale for the design may be explained succinctly with the aid of Table 1 and Figure 15. CSG is the only scheme with a potentially large domain and syntactically guaranteed validity. CSG representations also are concise and easy to create. These considerations lead to the choice of CSG as one of the representation schemes used internally by the GMS, and of a CSG-based input language as the main facility for creating new geometry. Reliable algorithms for converting CSG into boundary representations (but not the converse) are known; consistency requirements therefore imply that boundary representations be derived from CSG and that direct editing of boundary representations be prohibited. Approximate spatial occupancy enumerations may be computed from either CSG or boundary representations and used for internal purposes, for example, to facilitate spatial search or the approximate calculation of mass properties. Observe from Table 1 that a system containing CSG, boundary representations, and spatial enumerations

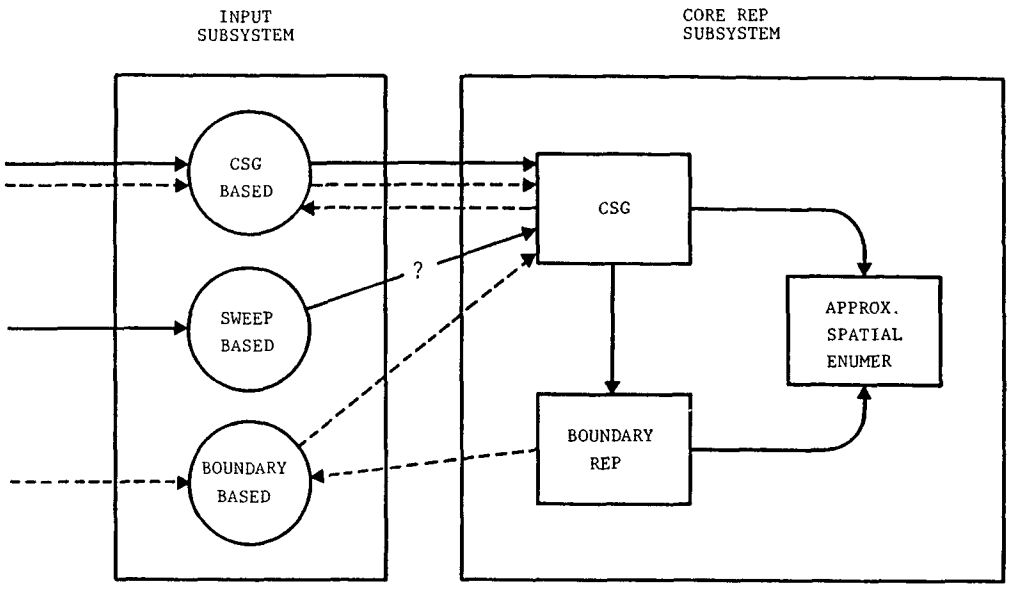


FIGURE 17. High-level architecture for an advanced modeler (dashed lines indicate editing facilities).

can support a large class of applications efficiently.

Some form of simple sweeping is desirable as an input medium because simple sweeps are congenial to human users. However simple sweeps cannot be part of the core subsystem of Figure 17 because their domains are not sufficiently rich. The interrogation mark in Figure 17 indicates that the conversion from sweep to CSG representations requires further research. (Observe that the conversion of sweeps into boundary representations does not solve the consistency problem.)

Finally, Figure 17 indicates the possibility of editing existing geometry by "talking" in terms of boundary entities (e.g., by graphically pointing at edge projections on a CRT screen). Note that boundary-based editing commands must be translated into changes in CSG representations. Although general means for effecting such "cross-representational" editing are not known, there appears to be a range of useful editing commands that may be supported reliably.

The architecture shown in Figure 17 is a significant departure from that used in the majority of the GMSs currently being designed and implemented, which rely on a single internal representation scheme (usually a boundary representation). Single-representation schemes are easier to design and implement than multiple-representation schemes because consistency issues can be ignored, but they are less flexible and extensible because all of the algorithms which archive or use geometry must be based on the single scheme.

4. SUMMARY AND REMARKS

Representations play a central role in the applications of computer science to the solution of problems that arise in the physical world. They are the sources of data for algorithms that answer questions (solve problems) about the corresponding physical entities. Thus representations of (the geometrical aspects of) physical solid objects are essential components of computer systems—such as GMSs—which deal with spatial phenomena.

This paper focuses on the representational issues that arise in the design and analysis of GMSs. The distinguishing feature of the present treatment is the clear distinction between physical solids, abstract (and representation-independent) mathematical models of solids, and representations for abstract solids. The distinc-

tion has several advantages. Specifically, (1) mathematical models can be studied independently of computational considerations, (2) such important concepts as representational validity and ambiguity can be defined mathematically, and (3) a rich body of mathematical knowledge can be applied to the study of geometric modeling (e.g., to derive conditions for representational validity).

The representation schemes used in a GMS are the major determinants of the system's characteristics. Thus, by using representation-scheme concepts, we are able to discuss GMSs in terms that are more precise and concise than those used previously [BADL78, BAER77, BRAI75b, OYAK76, SHU76].

The point of view presented in the paper is constructive in that it suggests means (multiple representations) for achieving a degree of versatility and efficiency higher than that of extant modelers,[9] and it also delineates gaps in our current knowledge about geometric modeling. The most prominent of these are centered on *applications*, which cover a very broad spectrum. Little is known in abstract terms about classes of applications, how applications may be defined as functions on geometric models, how such functions may be associated with algorithms that are matched to representations, and so forth. Other knowledge gaps include algorithms for representation conversion, fundamental properties of certain schemes, and methodologies for assessing the relative merits of algorithms which compute the same mathematical function but use significantly different input representations.

A final note: Elegantly designed representations will not ensure the success of a practical GMS; careful attention also must be paid to man-machine interaction, the design of archival facilities, and implementation issues.

[9] An industrially viable representational subsystem with the architecture proposed in Section 3.2 but catering also for tolerancing information [BROW78] is being built as a collaborative endeavor by the University of Rochester, the National Science Foundation, and a group of ten industrial users and vendors of geometry systems [BROW79].

APPENDIX. MATHEMATICAL DEFINITIONS

Let W be a set (the universe) and T a topology on W, that is, the collection of all open subsets of W. In the topological space (W, T) a subset X of W is a (closed) *regular* set if it equals the closure of its interior, that is,

$$X = kiX,$$

where k and i denote, respectively, closure and interior [KURA76]. The *regularized* set union ($\cup^*$), intersection ($\cap^*$), difference ($-^*$), and complement (c^*) of two subsets X and Y of W are defined as

$$X \cup^* Y = ki\,(X \cup Y),$$

$$X \cap^* Y = ki\,(X \cap Y),$$

$$X -^* Y = ki\,(X - Y),$$

$$c^*X = ki\,c\,X,$$

where c denotes the usual complement with respect to W. The regular subsets of W, together with the regularized set operators, form a Boolean algebra [KURA76, REQU78].

A function $F : E^3 \rightarrow E^1$ is *algebraic* if $F(x, y, z)$ is a polynomial in the coordinates x, y, z; it is *analytic* in a domain if it can be expanded in a convergent power series about each point of the domain. A set is *semialgebraic* if it can be expressed as a finite Boolean combination (via the set operators $\cup$, $\cap$, $-$, c) of sets of the form

$$\{(x, y, x) : F_i(x, y, z) \leq 0\},$$

where the F_i are algebraic. *Semianalytic* sets are defined similarly, but the F_i are required to be analytic rather than algebraic. The class of regular semialgebraic sets is closed under regularized set operations; similar results hold for semianalytic sets [HIRO74].

Let $\{H_i\}$ be a collection of regular semianalytic sets. The *regularized Boolean class* defined by $\{H_i\}$ is the class of all sets that may be expressed as a finite Boolean combination (via the regularized operators $\cup^*$, $\cap^*$, $-^*$) of sets in $\{H_i\}$. When all the sets in $\{H_i\}$ are bounded, and hence r-sets, all the sets in the regularized Boolean class defined by $\{H_i\}$ also are r-sets.

ACKNOWLEDGMENTS

I would like to thank all of my colleagues in the Production Automation Project, and especially Herbert Voelcker and Robert Tilove, for the many helpful discussions on representational issues that we have had over the past few years Adele Goldberg's and John Boyse's comments, and Rachel Rutherford's red marks on an earlier version of this paper, led to very significant improvements.

The work reported in this paper was supported primarily by the National Science Foundation under grants GI-34274X, APR76-01034, and DAR78-17064.

REFERENCES

AGIN76 AGIN, G. J., AND BINFORD, T. O. "Computer descriptions of curved objects," *IEEE Trans. Comput.* **C-25,** 4 (April 1976), 439–449.

AGOS76 AGOSTON, M. K. *Algebraic topology*, Marcel Dekker, New York, 1976.

AHO74 AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. *The design and analysis of computer algorithms*, Addison-Wesley, Reading, Mass., 1974.

ANSI78 "Digital representation of physical object shapes," Proposed American National Standard ANSI Y14.26 1, American Society of Mechanical Engineers, New York, 1978.

ARNO79 ARNON, D. "A cellular decomposition algorithm for semi-algebraic sets," Tech Rep. 353, Computer Science Dep., Univ. Wisconsin, Madison, June 1979.

BADL78 BADLER, N., AND BAJCSY, R. "Three-dimensional representations for computer graphics and computer vision," *ACM Comput. Gr.* **12,** 3 (Aug. 1978), 153–160.

BADL79 BADLER, N., O'ROURKE, J., AND TOLTZIS, H. "A spherical representation of a human body for visualizing movement," *Proc. IEEE* **67,** 10 (Oct. 1979), 1397–1403.

BAER77 BAER, A., EASTMAN, C., AND HENRION, M. "A survey of geometric modeling," Res. Rep. 66, Institute of Physical Planning, Carnegie-Mellon Univ., Pittsburgh, Pa., March 1977.

BARN74 BARNHILL, R. E., AND RIESENFELD, R. F. (EDS.). *Computer aided geometric design*, Academic Press, New York, 1974.

BAUM74 BAUMGART, B G "Geometric modelling for computer vision," Rep. STAN-CS-74-463, Stanford Artificial Intelligence Lab., Stanford Univ., Stanford, Calif., 1974.

BERN75 BERNASCON, Y J., AND BRUN, J. M. "Automated aids for the design of mechanical parts," Tech. Paper MS75-508, Society of Manufacturing Engineers, Dearborn, Mich., 1975.

BORK78 BORKIN, H. J., MCINTOSH, J. F, AND TURNER, J. A. "The development of three-dimensional spatial modeling techniques for the construction planning of nuclear power plants," *ACM Comput. Gr.* **12,** 3 (Aug. 1978), 341–347.

BOYS78 BOYSE, J. W. "Preliminary design for a geometric modeller," Res. Pub. GMR-2768, Computer Science Dep., General Motors Research Labs., Warren, Mich., July 1978.

BOYS79a BOYSE, J. W. "Interference detection among solids and surfaces," *Commun. ACM* **22,** 1 (Jan. 1979), 3–9.

BOYS79b BOYSE, J. W. "Data structure for a solid modeller," Res. Pub. GMR-2933, Computer Science Dep., General Motors Research Labs, Warren, Mich., March 1979.

BRAI73 BRAID, I. C. "Designing with volumes," Ph.D. Dissertation, Univ. Cambridge, England, 1973.

BRAI75a BRAID, I. C. "The synthesis of solids bounded by many faces," *Commun. ACM* **18,** 4 (April 1975), 209–216.

BRAI75b BRAID, I C. "Six systems for shape design and representation—A review," in *Proc. CAM-I's Int. CAM Seminar*, CAM-I, Inc., Bournemouth, England, April 1975, pp. 60–67.

BRAI78a BRAID, I. C. "New directions in geometric modelling," in *Proc Geometric Modeling Project Mtg* , CAM-I, Inc., St. Louis, Mo., March 1978.

BRAI78b BRAID, I. C., HILLYARD, R. C., AND STROUD, I. A. "Stepwise construction of polyhedra in geometric modelling," C.A.D. Group Document No. 100, Univ. Cambridge, England, Oct. 1978.

BROO79 BROOKS, R. A., GREINER, R., AND BINFORD, T. O. "The ACRONYM model-based vision system," in *Proc. 6th Int. Conf. on Artificial Intelligence*, vol. 1, Tokyo, Japan, Aug. 1979, pp 105–113.

BROW78 BROWN, C. M., REQUICHA, A. A. G , AND VOELCKER, H. B. "Geometric modelling systems for mechanical design and manufacturing," in *Proc 1978 ACM Annual Conf.*, Washington, D.C , Dec. 1978, pp. 770–778.

BROW79 BROWN, C. M., AND VOELCKER, H B. "The PADL-2 project," in *Proc 7th NSF Conf. on Production Research and Technology*, Ithaca, N.Y., Sept. 1979, pp. F1–F6.

CART79 CARTER, W. A. (ED.) *Geometric Modelling Seminar*, CAM-I, Inc., Bournemouth, England, Nov 1979.

CLOW71 CLOWES, M. "On seeing things," *Artif. Intell.* **2,** 1 (1971), 79–116.

COHE79 COHEN, J., AND HICKEY, T "Two algorithms for determining volumes of convex polyhedra," *J. ACM* **26,** 3 (July 1979), 401–414.

EAST77 EASTMAN, C., AND HENRION, M. "GLIDE: A language for design information systems," *ACM Comput. Gr.* **11,** 2 (July 1977), 24–33.

EAST79 EASTMAN, C. M., AND WEILER, K. "Geometric modelling using the Euler operators," Res. Rep. 78, Institute of Physical

Planning, Carnegie-Mellon Univ., Pittsburgh, Pa., Feb. 1979.

ENGE73 ENGELI, M. E. "A language for 3D graphics applications," in *Int. Computing Symposium*, A. Gunther et al. (Eds.), North-Holland, Amsterdam, 1973, pp. 459–466.

FUCH77 FUCHS, H., KEDEM, Z. M., AND USELTON, S. P. "Optimal surface reconstruction from planar contours," *Commun. ACM* **20**, 10 (Oct. 1977), 693–702.

GALL73 GALLAGHER, C. C., AND KNIGHT, W. A. *Group technology*, Butterworths, London, 1973.

GOLD71 GOLDSTEIN, R. A., AND NAGEL, R. "3-D visual simulation," *Simulation* **16**, 1 (1971), 25–31

GOLD79 GOLDSTEIN, R., AND MALIN, L. "3D modelling with the Synthavision system," in *Proc. 1st Ann. Conf. Computer Graphics in CAD/CAM Systems*, Cambridge, Mass, April 1979, pp. 244–247

GROS76 GROSSMAN, D. D. "Procedural representation of three-dimensional objects," *IBM J. Res. Dev* **20** (Nov. 1976), 582–589.

HERB78 HERBISON-EVANS, D. "NUDES 2: A numeric utility displaying ellipsoid solids, version 2," *ACM Comput. Gr.* **12**, 3 (Aug. 1978), 354–356.

HIRO74 HIRONAKA, H. "Triangulations of algebraic sets," in *Algebraic Geometry, ARCATA 1974, Proc. of Symposia in Pure Mathematics*, American Mathematical Society, Providence, R.I., 1975.

HOSA74 HOSAKA, M., KIMURA, F., AND KAKISHITA, N. "A unified method for processing polyhedra," in *Information processing '74*, North-Holland, Amsterdam, 1974, pp. 768–772.

HOSA77 HOSAKA, M., AND KIMURA, F. "An interactive geometrical design system with handwriting input," in *Information processing '77*, B. Gilchrist (Ed.), North-Holland, Amsterdam, 1977, pp. 167–172.

HUFF71 HUFFMAN, D. "Impossible objects as nonsense sentences," in *Machine intelligence*, vol. 6, B. Meltzer and D. Michie (Eds), Edinburgh University Press, Edinburgh, Scotland, 1971

IITR67 IIT RESEARCH INSTITUTE *APT part programming*, McGraw-Hill, New York, 1967.

KURA76 KURATOWSKI, K., AND MOSTOWSKI, A. *Set theory*, North-Holland, Amsterdam, 1976.

LANI73 LANING, J. H., LYNDE, D. A., AND MORREGGIA, V. "SHAPES user's manual," Charles Stark Draper Lab., Cambridge, Mass., May 1973.

LANI79 LANING, J. H., AND MADDEN, S. J "Capabilities of the SHAPES system for computer aided mechanical design," in *Proc. 1st Ann. Conf. Computer Graphics in CAD/CAM Systems*, Cambridge, Mass., April 1979, pp. 223–231

LEE76 LEE, D. T., AND PREPARATA, F P. "Location of a point in a planar subdivision and its applications," in *Proc. 8th ACM Symp Theory of Computing*, May 1976, pp. 231–235.

LEE80 LEE, Y. T, AND REQUICHA, A. A. G. "Algorithms for computing the volume and other integral properties of solid objects," Tech. Memo. 35, Production Automation Project, Univ. Rochester, Rochester, N.Y., Feb. 1980.

MARC74 MARCH, L., AND STEADMAN, P. *The geometry of environment*, M.I.T. Press, Cambridge, Mass., 1974.

MARK80 MARKOWSKY, G., AND WESLEY, M. A. "Fleshing out wireframes," Res. Rep. RC 8124, IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y., 1980.

MITC78 MITCHELL, W. J., AND OLIVERSON, M. "Computer representation of three-dimensional structures for CAEADS," Rep. CERL-TR-P-86, Construction Engineering Research Lab., U.S. Army Corps of Engineers, Champaign, Ill., 1978

NOWA80 NOWACKI, H. "Curve and surface generation and fairing," in *Computer aided design, lecture notes on computer science no. 89*, J. Encarnacao (Ed.), Springer-Verlag, New York, 1980.

OKIN73 OKINO, N., KAKAZU, Y., AND KUBO, H. "TIPS-1 Technical information processing system for computer-aided design, drawing and manufacturing," in *Computer languages for numerical control*, J. Hatvany (Ed.), North-Holland, Amsterdam, 1973, pp. 141–150.

OKIN78 OKINO, N., ET AL. "TIPS-1," Institute of Precision Engineering, Hokkaido Univ., Sapporo, Japan, 1978.

OYAK76 OYAKE, I., AND SHU, H. "Object synthesis for design/manufacturing," in *Proc 13th NC Society Ann. Mtg. and Tech. Conf.*, Cincinnati, Ohio, March 1976, pp. 541–553.

PARE77 PARENT, R. E. "A system for sculpting 3-D data," *ACM Comput. Gr.* **11**, 2 (July 1977), 138–147.

POPP75 POPPLESTONE, R. J, BROWN, C M., AMBLER, A. P, AND CRAWFORD, G. F. "Forming models of plane-and-cylinder faceted bodies from light stripes," in *Proc. 4th Int. Joint Conf. Artificial Intelligence*, Tbilisi, Georgia, U.S.S.R., Sept. 1975, pp. 664–668.

PREI79 PREISS, K "A procedure for checking the topological consistency of a 2-D or 3-D finite element mesh," in *Proc. 16th Design Automation Conf.*, San Diego, Calif, June 1979, pp 200–206.

REDD78 REDDY, D. R., AND RUBIN, S. "Representation of three-dimensional objects," Rep. CMU-CS-78-113, Dep Computer Science, Carnegie-Mellon Univ., Pittsburgh, Pa., April 1978.

REQU77a REQUICHA, A. A. G. "Part and assembly description languages I: Dimensioning and tolerancing," Tech. Memo. 19, Production Automation Project, Univ Rochester, Rochster, N.Y., May 1977.

REQU77b REQUICHA, A. A. G. "Mathematical models of rigid solid objects," Tech. Memo. 28, Production Automation Project, Univ. Rochester, Rochster, N.Y., Nov. 1977

REQU77c REQUICHA, A. A. G., AND VOELCKER, H. B. "Constructive solid geometry," Tech. Memo. 25, Production Automation Project, Univ. Rochester, Rochester, N.Y., Nov 1977.

REQU78 REQUICHA, A. A. G., AND TILOVE, R. B. "Mathematical foundations of constructive solid geometry. General topology of regular closed sets," Tech. Memo. 27, Production Automation Project, Univ Rochester, Rochester, N.Y., March 1978.

REQU80 REQUICHA, A. A G. "Representations of rigid solid objects," in *Computer aided design, lecture notes on computer science*, no. 89, J. Encarnacao (Ed.), Springer-Verlag, New York, 1980

SHAM78 SHAMOS, M. I. "Computational geometry," Ph.D. Dissertation, Yale Univ., New Haven, Conn., May 1978.

SCHU76 SCHUSTER, R. "System und Sprache zur Behandlung graphischer Information in rechnergestutzten Entwurf," Rep. KFK-2305, Kernforschungszentrum, Karlsruhe, West Germany, 1976.

SEIF78 SEIFERT, H., BARGELE, N., AND FRITSCHE, B. "Different ways to design three-dimensional representations of engineering parts with PROREN2," in *Proc. Conf. Interactive Techniques in Computer Aided Design*, Bologna, Italy, 1978, pp. 335–343.

SHU76 SHU, H "Geometric modeling for mechanical parts," in *Proc 4th NSF Conf Production Research and Technology*, Chicago, Ill , Dec. 1976, pp. 10–15

SIMO69 SIMON, H A. *The sciences of the artificial*, M.I.T. Press, Cambridge, Mass., 1969.

SPUR76 SPUR, G., AND GAUSEMEIER, J. "Processing of workpiece information for producing engineering drawings," in *Proc. 16th Int. Machine Tool Design and Research Conf.*, Manchester, England, 1976, pp. 17–21.

SPUR78 SPUR, G. "Status and further development of the geometric modelling system COMPAC," in *Proc. Geometric Modelling Project Mtg.*, CAM-I, Inc., St Louis, Mo , March 1978.

TILO80a TILOVE, R. B. "Set membership classification: A unified approach to geometric intersection problems," *IEEE Trans Comput* **C-29,** 10 (Oct. 1980), 874–883.

TILO80b TILOVE, R. B., AND REQUICHA, A. A. G. "Closure of Boolean operations on geometric entities," *Comput. Aided Des.* **12,** 5 (Sept. 1980) 219–220.

VOEL74a VOELCKER, H. B., MIDDLEDITCH, A. E., ZUCKERMAN, P. R , FISHER, W. B., NELSON, T. S., REQUICHA, A. A G., AND SHOPIRO, J. E. "Discrete part manufacturing Theory and practice," Part I, Tech. Rep. 1, Production Automation Project, Univ Rochester, Rochester, N Y., Dec. 1974.

VOEL74b VOELCKER, H. B., REQUICHA, A A. G., HARTQUIST, E. E., FISHER, W. B., SHOPIRO, J. E., AND BIRRELL, N. K. "An introduction to PADL. Characteristics, status, and rationale," Tech. Memo. 22, Production Automation Project, Univ. Rochester, Rochester, N.Y., Dec. 1974

VOEL77 VOELCKER, H. B. AND REQUICHA, A. A. G. "Geometric modelling of mechanical parts and processes," *IEEE Comput.,* **10,** 12 (Dec. 1977), 48–57.

VOEL78 VOELCKER, H., REQUICHA, A., HARTQUIST, E., FISHER, W., METZGER, J., TILOVE, R., BIRRELL, N., HUNT, W , ARMSTRONG, G., CHECK, T , MOOTE, R., AND MCSWEENEY, J. "The PADL-1.0/2 system for defining and displaying solid objects," *ACM Comput. Gr.,* **12,** 3 (Aug. 1978), 257–263.

VOEL80 VOELCKER, H. B., REQUICHA, A. A. G. "Boundary evaluation procedures for objects defined via constructive solid geometry," Tech. Memo. 26, Production Automation Project, Univ. Rochester, Rochester, N.Y., 1980.

WALT75 WALTZ, D "Understanding line drawings of scenes with shadows," in *The psychology of computer vision*, P. H. Winston (Ed), McGraw-Hill, New York, 1975, pp. 19–91

WESL80a WELSEY, M. A., LOZANO-PEREZ, T., LIEBERMAN, L. T., LAVIN, M. A., AND GROSSMAN, D D. "A geometric modelling system for automated mechanical assembly," *IBM J Res. Dev.* **24,** 1 (Jan 1980), 64–74.

WESL80b WESLEY, M. A. "Construction and use of geometric models," in *Computer aided design, lecture notes on computer science*, no. 89, J. Encarnacao (Ed.), Springer-Verlag, New York, 1980.

WOO79 WOO, T. C., AND POSHADLO, R. F. "A feature oriented design modification system," in *Proc. 1st Ann. Conf. Computer Graphics in CAD/CAM Systems*, Cambridge, Mass , April 1979.

YESS78 YESSIOS, C. I "A notation and system for 3-D constructions," in *Proc 15th Design Automation Conf.*, Las Vegas, Nev , June 1978, pp 125–132.

RECEIVED DECEMBER 1978; FINAL REVISION ACCEPTED AUGUST 1980