TM-29

REPRESENTATIONS OF RIGID SOLID OBJECTS

by

A. A. G. Requicha

PRODUCTION AUTOMATION PROJECT

June 1980

Reprinted, April 1981

Reprinted, December 1981

TECHNICAL MEMORANDUM 29

PRODUCTION AUTOMATION PROJECT
College of Engineering and Applied Science
University of Rochester
Rochester, New York 14627

TM-29

REPRESENTATIONS OF RIGID SOLID OBJECTS

by

A. A. G. Requicha

June 1980

The work described in this document was supported primarily by the National Science Foundation under Grants GI-34274X, APR76-01034 and DAR78-17064. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

This paper will be published by Springer Verlag as Chapter 1 of COMPUTER AIDED DESIGN, Lecture Notes in Computer Science No. 89. It has been released as a Technical Memorandum for early distribution and should not be duplicated.

REPRESENTATIONS OF RIGID SOLID OBJECTS

CONTENTS

1.	Introduction.....	1
2.	Characterization of Representation Schemes.....	4
2.1	Definitions.....	4
2.2	A Methodology for Defining Specific Representation Schemes.....	6
2.3	Examples.....	9
2.4	Formal Properties of Representation Schemes.....	12
2.4.1	Domain.....	12
2.4.2	Validity.....	12
2.4.3	Completeness.....	14
2.4.4	Uniqueness.....	14
2.5	Informal Properties of Representation Schemes.....	16
2.5.1	Conciseness.....	16
2.5.2	Ease of Creation.....	17
2.5.3	Efficacy in the Context of Applications.....	17
2.6	Multiple Representations: Consistency, Equivalence, and Conversion.....	19
2.7	Representation Hierarchies.....	23
3.	Survey of Representation Schemes for Rigid Solids.....	25
3.1	Preview.....	25
3.2	Ambiguous Representations.....	26
3.3	Pure Primitive Instantiating Schemes.....	28
3.4	Spatial Occupancy Enumeration.....	30
3.5	Cell Decompositions.....	32
3.6	Constructive Solid Geometry.....	34
3.6.1	CSG Trees.....	34
3.6.2	Primitive Leaves.....	35
3.6.3	Formal Properties.....	38
3.6.4	Informal Properties.....	41
3.6.5	Use of CSG and Related Schemes in GMS's.....	41
3.7	Sweep Representations.....	43
3.7.1	Translational and Rotational Sweeping.....	43
3.7.2	Volumes Swept by Motions of a Solid.....	46
3.7.3	General Sweeping.....	47
3.8	Boundary Representations.....	48

3.8.1	Design Issues.....	50
3.8.2	Formal Properties.....	54
3.8.3	Informal Properties.....	57
3.9	Hybrid Schemes.....	58
3.10	Conversion Between Representations.....	59
4.	Summary and Conclusions.....	61
	References.....	62

1. INTRODUCTION

Most mechanical products are collections of rigid solids ("parts") which are manufactured and assembled via processes whose effects are primarily geometrical. Geometrical information -- for specifying parts and products, and for controlling processes -- obviously is of crucial importance in the mechanical industries, and COMPUTATIONAL means for handling such information therefore are required for automatic or computer-aided design and production.

This section discusses the specification of object geometry in the context of Computer Aided Design (CAD) systems. Figure 1-1 provides a high-level, functional view of a CAD system. Observe that --

- 1) The paramount goal is the creation of REPRESENTATIONS (symbol structures) which designate (define) physical solids. While some of the information required to define a solid mechanical part is non-geometrical (e.g. material, hardness, ...) the bulk of it is geometrical.
- 2) Representations are the sources of data for programs. Some programs may be regarded as CAD-system components because they generate displays or compute other important properties of objects, while other programs (e.g. for manufacturing planning) are properly external to the CAD system.

FIGURE 1-1

A Functional View of a Mechanical CAD System

The Geometric Modelling (sub)System (GMS) shown in Fig. 1-1 provides facilities for entering, storing, and editing representations of the geometry of solid objects. At present most of the geometric information contained in a GMS's representations must be supplied by human users of the CAD system in which the GMS is imbedded, because the available automatic-design procedures (e.g. for selecting quasi-optimal parameters for gear trains or heat exchangers) are very specialized. Powerful automatic design systems driven from functional specifications are coming into use in the electronic industries, but analogous systems for the mechanical industries cannot be developed until reasonably general means are found to specify formally the function of mechanical artifacts, and to relate function to geometry. (Automatic design procedures are not shown in Fig. 1-1.)

In this paper we shall be concerned with the characterization and design of representations for solids, because the properties of the representations in a GMS determine its ultimate capabilities and limitations.

We attack the representation problem in the two stages shown in Fig. 1-2. First we define a MATHEMATICAL MODELLING SPACE (shown at the center of Fig. 1-2) by replacing the real-world entities under study (physical solids) with mathematical entities which we call MATHEMATICAL MODELS of solids (or simply ABSTRACT SOLIDS); we then associate REPRESENTATIONS, i.e. SYMBOL STRUCTURES constructed over a finite alphabet, to the mathematical models. The choice of an abstract modelling space is discussed in [58] where it is argued that the appropriate mathematical models are subsets of E^3 (three-dimensional Euclidean space) which are bounded, closed, regular and semi-analytic. We call such sets R-SETS. Intuitively, r-sets may be viewed as "solid curved polyhedra bounded by well-behaved surfaces".

This paper focusses on the problem of associating representations to r-sets. The main goals of the paper are to establish a precise framework for characterizing schemes for representing rigid solid objects (Section 2), and to describe the known schemes in terms of the framework (Section 3). While a discussion of specific GMS's is beyond the scope of this paper, we shall provide brief comments on a wide range of illustrative extant systems.

FIGURE 1-2

A Two-Stage Approach to Geometric Modelling

The paper deals directly only with the representation of **NOMINAL** (i.e. untoleranced) objects; the practically important problem of representing tolerancing information is addressed elsewhere [57] [34]. Tolerance specifications define classes of solids which are functionally equivalent and interchangeable in assembly. Therefore, to represent toleranced objects one must design representation schemes for **EQUIVALENCE CLASSES** of solids. This involves representing a nominal solid plus allowable variations. The characterization of representation schemes developed in Section 2 applies equally to the representation of (single) nominal objects or of classes of solids, but Section 3 deals exclusively with nominal objects and does not cover the representation of variations.

We have endeavored to make the paper understandable to a spectrum of readers, without sacrificing mathematical rigor, by providing intuitive interpretations for all of the concepts discussed. Thus, readers who are familiar with the elementary concepts of sets, relations and mappings, but not with r-set theory [58] [60], should be able to follow the main lines of reasoning simply by ignoring technical mathematical details and focussing on the intuitive meanings of technical terms.

* * *

2. CHARACTERIZATION OF REPRESENTATION SCHEMES

This section defines representation schemes and discusses some of their important formal and informal properties.

2.1 DEFINITIONS

A mathematical modelling space enables us to define and study the properties of generic elements of the space, but provides us with no direct means for "talking about" or for computing properties of specific elements of the space. It is REPRESENTATIONS that provide means for designating (referring to, "naming") individual elements of a mathematical modelling space, and they constitute the raw material on which algorithms operate.

Representations are finite structures constructed with symbols from a finite alphabet according to syntactical rules. Such symbol structures are equivalent to strings of symbols, and therefore a set of syntactically admissible (i.e. syntactically correct) representations may be viewed as a language. Section 2.2 below discusses briefly the means we use for describing sets of admissible representations. Our main concern is the meaning or semantics of representations, which we define below by means of representation schemes. In essence, a representation scheme associates representations (symbol structures) with the entities (models) they designate in a mathematical modelling space.

Before defining representation schemes formally we shall review briefly some elementary mathematical concepts. A (binary) RELATION z between two sets X and Y is simply a set of ordered pairs (x, y) where x is an element of X and y is an element of Y . We use $z : X \rightarrow Y$ to denote a relation z from X to Y . The set of all x which occur as first elements in a pair is the DOMAIN D of the relation, and the set of all y which occur as second elements in a pair is the RANGE V of the relation. Generally, D and V are proper subsets of X and Y , i.e. the relation need not be defined for all points in the spaces X and Y . The inverse of a relation z , denoted z^{-1} , is the relation obtained by inverting the order of the elements in each pair. Finally, a function (also called a mapping, transformation, etc.) is a single-valued relation, i.e. a relation in which no element x of X occurs more than once as the first element of a pair.

A REPRESENTATION SCHEME is defined as a relation $s : M \rightarrow R$, where M is a mathematical modelling space and R is a REPRESENTATION SPACE, i.e. a set of syntactically correct symbol structures over some finite alphabet. We denote the domain of s by D , and the image of D under s , i.e. the range of s , by V . Any representation in the range V is said to be VALID. Valid representations must be both syntactically and semantically correct, i.e. they must have corresponding elements in M (see Fig. 2-1).

FIGURE 2-1

Domain and Range of a Representation Scheme

Observe in Fig. 2-1 that we neither assume that all objects are representable (i.e. D need not equal M) nor that all syntactically correct representations are valid (i.e. V need not equal R). Our rationale is the following. The range V may be viewed as a language, but the ranges of practical representation schemes seldom may be defined by "reasonable" (e.g. context-free) grammars. Therefore it is more useful to consider representation spaces R that may be defined easily by formal means, with the understanding that not all elements of R are valid representations. This implies that the syntax of representations is not an essential property of a representation scheme s because any syntax that defines a superset of V leads to a scheme that differs from s only in a trivial formal sense. In the remainder of the paper we shall use the terms "syntax" and "semantics" somewhat informally, and we shall assume that the syntax is chosen sensibly so that R is not unnecessarily larger than V .

A "syntactical variant" R' of a representation space R may be obtained by establishing a one-to-one correspondence between R and R' via a mapping v' that does not depend on the space M being represented. The correspondence between elements of R and R' must be independent of the meaning of the representations. For example, if representations in R are tree structures one may construct a syntactical variant R' by a mapping v' which replaces each tree structure of R by its postfix string equivalent.

Observe in this example that the mapping v' is indeed independent of the meaning of the representations. While syntactical variants may be useful for certain purposes (e.g. to increase readability by humans), they will be ignored in this paper because the properties of representation schemes discussed in the paper are essentially independent of the schemes' syntax.

We conclude by defining two formal properties of representations and representation schemes: completeness and uniqueness. A representation r in V is UNAMBIGUOUS or COMPLETE if it corresponds to a single object, i.e. if the set $s^{-1}(r)$ is a single-element subset $\{m\}$ of M . It is unique if its corresponding objects do not admit representations other than r in the scheme, i.e. if $s(s^{-1}(r)) = \{r\}$.

A representation scheme $s : M \rightarrow R$ is COMPLETE or UNAMBIGUOUS if all its valid representations are unambiguous, i.e. if the inverse relation s^{-1} is a function. A scheme is UNIQUE if all its valid representations are unique, i.e. if s is a function. Note that completeness does not imply nor is implied by uniqueness. An unambiguous and unique representation scheme establishes a one-to-one correspondence between its domain and its range. (In the mathematical literature the term "representation" is used mainly to denote such one-to-one mappings, and representations are not restricted to finite-length symbol structures.)

We shall discuss the importance of these notions in Section 2.4 below, but first we shall introduce a useful methodology for defining representation spaces and schemes, and present examples to illustrate what these almost trivial but abstract definitions mean in concrete terms.

* *

2.2 A METHODOLOGY FOR DEFINING SPECIFIC REPRESENTATION SCHEMES

We introduce the required notions informally, mainly by means of an example, because they are quite simple and may be familiar to many readers.

Consider a family of solid squares in E^2 having the following properties. Each square has sides of unit length, each side is parallel to either the X or the Y axis, and each vertex has integer coordinates. Figure 2-2 shows several members of the family. Each member may be characterized by the coordinates of (say) its left lower corner, and therefore the family depends on two parameters. (Parametric families of entities often are called "entity schemata" or GENERIC ENTITIES.)

FIGURE 2-2

A Simple Scheme for Representing a Particular Class of 2-D Sets

A member of a family of entities is called an entity **INSTANCE**. The process of **INSTANCING** (or instantiating), i.e. of selecting a specific member of the family, corresponds to assigning specific values to the family's parameters. The square instances in Fig. 2-2a may be represented by the strings $SQ(1,2), \dots$ shown in the figure. Such strings are called instances of a **GENERIC TUPLE** $SQ(x,y)$, where x and y are (integer) variables. Observe that a tuple instance in our example (and in common practice) contains a "family name" or "type code" which designates a specific family, and a set of parameter values which designates a specific member of the family.

We may devise a scheme for representing more complex 2-D sets by constructing **STRUCTURED** representations which contain tuple instances, each of which represents a square instance. For example, we may define structured representations to be ordered sets of tuple instances. Fig. 2-2b shows an L-shaped 2-D set and its corresponding representation in such a scheme.

The example illustrates the basic technique we shall use for defining specific representation schemes:

- 1) Define families of entities -- called **GENERIC PRIMITIVES** -- whose elements are called **PRIMITIVE INSTANCES**. (When no confusion is likely to arise we shall use "primitive" to

refer both to generic primitives and their instances; "tuple" will be used similarly.) A representation scheme must possess a FINITE number of generic primitives.

- 2) Associate with each primitive for representational purposes a GENERIC TUPLE, and with each primitive instance a TUPLE INSTANCE ($q_0, q_1, \dots, q_N$), where q_0 is an integer or character string usually called a TYPE CODE, and $q_1, \dots, q_N$ are numeric parameter values, which often are real numbers. (We shall ignore the difficulties associated with representing real numbers, and use as a model of computation a real random access machine whose registers can store real numbers [63].) The type codes for distinct generic primitives in a representation scheme must be distinct. The "length" $N+1$ of a tuple is fixed for each generic tuple, but may vary among generic tuples.
- 3) Define STRUCTURING RULES for constructing STRUCTURED REPRESENTATIONS from tuple instances. Structured representations generally have "variable length", i.e. they may contain an arbitrary but finite number of tuple instances.
- 4) Define the semantics of structured representations by exhibiting a mathematical rule that associates geometric entities to such representations (or conversely).

We note in passing that one may define representation schemes in which single tuple instances are the only admissible representations. Such schemes -- called "pure primitive instancing schemes" or "trivially structured schemes" -- are discussed further in Section 3.3 below.

Observe that the relation $t : T \rightarrow T'$, where T is a generic primitive (a family of geometric entities), and T' is the corresponding set of tuple instances, is a representation scheme in its own right and therefore the definitions of Section 2.1 apply. In particular, it follows that tuple instances that correspond to certain values of the parameters $q_1, \dots, q_N$ may be invalid. In practice, the validity of tuple instances may be verified easily because the primitives usually are simple geometric entities.

Generic tuples provide convenient means for representing parameterized families of geometric entities, regardless of whether such entities are primitives in a scheme. Therefore it is useful to consider GENERIC STRUCTURED ENTITIES (variously called macro-, procedure- or parametric-entities, or entity schemata) representable by tuples whose semantics are defined by structures with non-instantiated parameters.

* *

2.3 EXAMPLES

Suppose we wish to represent simple planar polygons. Let M be the space of all simple polygons in E^2 , where by a simple polygon we mean precisely a subset of E^2 that is homeomorphic (topologically equivalent) to a disc and whose boundary is the union of a finite number of bounded line segments. (Intuitively, a homeomorphism is an "elastic deformation".) The polygon's vertices are the boundary's "corners", i.e. the points where the tangent to the boundary is not single-valued. This definition of simple polygon coincides with the usual notion of a polygon with no holes and no self-intersecting sides. Simple polygons are "2-D solids" containing all of the points "inside" and "on" the polygonal boundary.

The acknowledgement of a modelling space M enables us to define general properties of polygons. For example we may define convexity: a polygon is convex if it contains every line segment whose endpoints belong to the polygon. We may also define the diameter ("maximal dimension") of the polygon as the maximal distance between any two points of the polygon. Note that these are "representation-free" (representation-independent) properties of polygons because they do not depend on the particular means chosen to designate specific polygons.

A representation scheme for polygons -- which we shall call Scheme I for later reference -- may be defined as follows. Let the domain D be M , and let the syntactically correct representations in R be ORDERED sets of points in E^2 , each represented by a 2-tuple of Cartesian coordinates.

Let us first rephrase the definition of the scheme's syntax to illustrate the notions introduced in Section 2.2 above. Consider a single generic primitive which is the family of E^2 points, and a single generic tuple (q_1, q_2) , where q_1 and q_2 are real variables. (Because there is but one primitive, the type code q_0 is unnecessary and will be omitted.) Primitive instances are individual points of E^2 , and therefore the space T of primitive instances coincides with E^2 . The space T' of tuple instances is the space of all pairs of real numbers. The relation $t: T \rightarrow T'$ is simply the one-to-one and onto function which assigns to each point of E^2 its coordinates in a fixed Cartesian coordinate system. It is clear that all of the tuple instances are valid with respect to t since a pair (x, y) , where x and y are arbitrary real numbers, always corresponds to some point of E^2 . The structuring rule amounts simply to constructing ordered sets of tuple instances.

To complete the definition of Scheme I we must specify the meaning of structured representations. We define the relation $s: M \rightarrow R$ by the rule: construct an ordered set of points by listing in arbitrary order the vertices of the polygon, ensuring that each vertex appears once and only once in the list. Figure

2-3 shows a polygon and three representations constructed according to the above rule.

FIGURE 2-3

A Polygon and Three Admissible Vertex Lists

Clearly, s is mathematically well defined. The range V is defined implicitly since both D and the association rule are known. It is easy to see that V is a proper subset of R , i.e. there are syntactically correct representations that are not (semantically) valid. For example, a list of collinear points cannot represent a polygon. But is any finite set of isolated non-collinear points in E^2 the vertex set of a simple polygon? A "constructive" answer to this question -- which we leave as a non-trivial exercise for the reader -- implies a test for determining the validity of our representations.

Figure 2-3 shows that the representations are not unique, since a single polygon has several representations which correspond to different vertex permutations.

Figure 2-4 shows that the representation scheme also is not complete: the polygons in Fig. 2-4a and 2-4b are different and yet they have the same vertices (and hence the same representations).

We conclude that Scheme I is neither complete nor unique. We shall discuss two exemplary modifications that achieve completeness.

Scheme II: We restrict the domain to the class of convex polygons, and keep our initial association rule. The modified representation scheme is complete because (1) the represented polygon is the convex hull of its vertices and (2) a set of points has a unique convex hull. Note, however, that we have decreased the "coverage" (the domain) of our representation scheme.

FIGURE 2-4

Two Polygons that Admit Under
Scheme I the Representation P1 P2 P3 P4 P5

Scheme III: We revert to the domain of Scheme I but add information to the representations. Suppose specifically that we modify the association rule by requiring that vertices be listed in the order they are encountered when one traverses the polygon's boundary without backtracking. The modified scheme is complete because ordering the vertices is equivalent to defining the polygon's edges (and hence its boundary), and by Jordan's curve theorem [2] [32] the boundary defines a unique polygon.

The three schemes for representing planar polygons warrant a few final remarks.

- 1) All three have the same representational syntax. Schemes I and II differ only in their domain, while Schemes I and III differ only in their association rules. This should convince readers that representation schemes must be defined fully, i.e. both syntactically (the properties of elements of R) and semantically (the M, R pairings), if one is to assess their properties in a meaningful way.
- 2) We defined the schemes by defining their domains and association rules; we shall see later that it is often convenient to define representation schemes "backwards", i.e. to define the association rule and the range, rather than the domain.
- 3) One could represent polygons in many other ways. The formal properties developed below provide one basis for comparing alternative schemes, and the intended applications provide a second, complementary basis -- as we shall show in Section 3.

* *

2.4 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.

2.4.1 DOMAIN

The DOMAIN of a representation scheme characterizes the DESCRIPTIVE POWER of the scheme, i.e. the set of entities representable in the scheme.

*

2.4.2 VALIDITY

The RANGE of a representation scheme is the set of representations which are VALID. Representational validity is of obvious importance in ensuring the (syntactic as well as semantic) integrity of data bases, in that data bases should not contain symbol structures which correspond to "nonsense objects". (M. C. Escher's well-known prints provide examples of graphical representations of nonsense 3-D objects.) Evoking algorithms for computing object properties on invalid object representations usually has unpleasant effects which may range from system crashes to seemingly reasonable but meaningless results.

In the past, the responsibility for ensuring the validity of representations in a GMS has rested with humans. Typically, a human creates an object representation and then checks visually that the object "makes sense" by examining graphic displays. This is an error-prone procedure. More importantly, if representations of solids are created by automata (programs) rather than by humans as a step in the algorithmic solution of a problem, visual checking is usually impossible. Programs that create such representations for "internal purposes" already exist, as the following examples show.

- 1) The PADL-1.0/2 system generates section views of objects by creating a "new" sectioned solid and displaying it. Interference between parts in an assembly is handled similarly [70], [71].
- 2) A part-program verification system operating in the PADL-1 environment synthesizes solids from cutter path and cutter shape data [39].

Program-created object representations undoubtedly will proliferate in future systems. For example, systems for automatic manufacturing planning will need to deal with partially finished workpieces not specified by humans, and in due course humans will be able to specify the mechanical function of (non-aesthetic) objects to programs which will synthesize

automatically the objects' geometry.

These considerations lead us to believe that future GMS's cannot rely on human validation, and must ensure representational validity by automatic (algorithmic) means. Validity may be ensured automatically either by testing syntactically correct representations after they are constructed, or by embedding validity constraints in the procedures which create object representations. The cleanest way of ensuring validity is to design representation schemes such that ALL syntactically correct representations are valid (see Section 3.6 for an example).

Validity is also important when families of objects -- "generic objects" or "object schemata" -- are defined via representations which contain uninstantiated symbolic parameters. (The primitives of Section 2.2 are an important special class.) Representations of object schemata are akin to procedures or macros in algorithmic languages, and appear to be very useful in GMS's.

The two general approaches for ensuring validity cited earlier also apply to generic objects. Thus, validity may be tested after numeric values are bound to a schema's parameters, or (preferably) validity constraints may be built into the facilities for defining object schemata.

We shall illustrate the issues by means of our Scheme II for representing convex polygons via arbitrarily ordered vertex lists. Suppose that we want to deal with a family of 4-sided convex polygons having three fixed vertices and a variable vertex, as shown in Fig. 2-5. (This is a contrived example. Natural examples may be found in mechanical engineering, e.g. counterbored and countersunk holes, keyways, and standard fasteners, all of which may be defined generically in terms of symbolic parameters.) We may represent the family by a generic tuple $Q(a,b)$ which is defined by ("macro-expands" into) the vertex list

$(0,0) (0,1) (1,0) (a,b).$

Figures 2-5a and 2-5b show the polygons which correspond to $Q(2,2)$ and $Q(0.5,-0.5)$; Fig. 2-5c shows that $Q(0.5,0.2)$ is NOT a valid representation of a convex polygon in the scheme because the point (a,b) is not a vertex of the convex hull of the four points. We conclude that validity constraints are not built automatically into our definition of $Q(a,b)$, and that to ensure it one must find the appropriate constraints on the parameters (this may be difficult in less trivial examples) and embody them in conditionals within the schemata definitional facility.

FIGURE 2-5

Instances of Polygons in a Convex Polygon Scheme
that Admits One Variable Vertex

*

2.4.3 COMPLETENESS

Completeness (as defined in Section 2.1) is probably the most important characteristic of representation schemes because it captures precisely the intuitive notion of "informational completeness". Specifically, each representation in a complete scheme corresponds to exactly one entity in the mathematical modelling space M , and hence contains enough information to distinguish that entity from all other entities in the domain D of the scheme. Representations in incomplete schemes generally correspond to several modelling-space entities, i.e. do not contain enough information to distinguish single elements in D .

The importance of completeness is best understood in the context of applications, where one must compute properties of represented entities. An example will illustrate the issues. Suppose that one wishes to compute (1) the diameters and (2) the areas of possibly concave polygons in the domain $D = M$ of Schemes I and III. Earlier arguments show that polygons' diameters can be computed correctly from representations in the incomplete Scheme I, whereas polygons' areas cannot (see Fig. 2-4). On the other hand, representations in the complete Scheme III suffice for both calculations -- and indeed for the calculation of any mathematically defined property of any polygon representable in Scheme III -- because each representation designates exactly one polygon-entity in M . (We ignore here properties defined by functions that are not computable at all.)

It should be evident that incomplete representations can be adequate sources of data for specific applications, but that completeness should be sought when there is a WIDE RANGE of applications to be supported by a practical modelling system, and

especially when the range of applications is not known in advance.

*

2.4.4 UNIQUENESS

The importance of representational uniqueness derives mainly from the role of uniqueness in assessing the equality of two objects. Equality assessment is important in a number of applications. An example: repeated representations may be culled from a data base only if one can determine that two representations (in the same scheme) correspond to the same object. A second example: automatic planning algorithms which search a space of alternatives must be able to recognize previously encountered situations ... or else the search may loop indefinitely.

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". Equality assessment in schemes which are unambiguous but not unique requires more elaborate techniques.

Unambiguous but non-unique representations of r-sets may be tested for equality as follows. Given two r-sets A and B, construct the regularized symmetric difference $S = (A -* B) \cup (B -* A)$, and determine if S is the null set. (The operator $-*$ above denotes regularized set difference [59], [58].) It is easy to see that $S = \emptyset$ if and only if $A = B$. This method of determining equality requires a capability for deciding whether a regularized set composition is null.

An example will illustrate two types of non-uniqueness which are quite common in schemes for representing geometric entities. We shall modify our earlier polygon schemes as follows.

Scheme IV: The elements of the domain D are equivalence classes of congruent convex polygons rather than single polygons. A representation of a congruence class is an arbitrarily ordered list of the vertices of an arbitrary member of the class, with each vertex represented by its cartesian coordinates as in Section 2.3. The modification implies that we view all polygons of the same "shape" (i.e. congruent polygons) as the same single object regardless of the polygons' positions in the cartesian coordinate system. The representation scheme is not unique for two reasons.

- 1) Vertices in a list may be permuted.
- 2) Vertices may correspond to different members of a congruence

class which differ only by the member-polygons' positions (see Fig. 2-6).

FIGURE 2-6

Positional Non-uniqueness of Congruent Polygons Represented by Vertex Lists

"Permutational" non-uniqueness (case 1 above) is relatively trivial; nevertheless, determining whether two lists contain the same elements may be computationally expensive when the lists are large. "Positional" non-uniqueness is more pernicious because generally it is not easy to design algorithms for deciding whether two geometric entities are congruent.

* *

2.5 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.

2.5.1 CONCISENESS

Conciseness refers to the "size" of representations in a scheme. Concise representations are convenient to store and to transmit over data links whereas "verbose" representations are not. Concise representations contain relatively few "redundant" data. The validity of such representations is usually easy to ensure because non-redundancy implies that the entities which constitute 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 than computing, often-needed data derivable from a concise representation.

The notions of conciseness and redundancy have a strong information-theoretic flavor, and probably can be formulated precisely in terms of information theory. However, it is not clear that such a formalization is practically useful.

*

2.5.2 EASE OF CREATION

The ease with which (valid) representations may be created by users of modelling 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.

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

*

2.5.3 EFFICACY IN THE CONTEXT OF APPLICATIONS

Representations of objects (e.g. solids) should be viewed for practical purposes as sources of data for algorithms. Algorithms are computational embodiments of mathematical functions which associate values (e.g. diameter, volume,...) with objects. An unambiguous representation is guaranteed to be a sufficient source of data, but may not support some specific application efficaciously because it may not be a "convenient" or "efficient" source of data for ANY algorithm that computes the value of some specific function.

For each mathematical function it is often possible to design several representation schemes for elements of the function's domain and range, and several algorithms which "evaluate" the function, i.e. which map representations of elements in the function's domain into representations of corresponding elements in the function's range (see Fig. 2-7). The design of representation schemes that permit the use of "good" algorithms for evaluating practically useful functions is a central issue in the design of modelling systems. Good algorithms should be CORRECT, EFFICIENT, and should also exhibit subtler but important characteristics such as ROBUSTNESS in the presence of numerical errors, and EXTENSIBILITY.

FIGURE 2-7

Alternative Representations and Algorithms for Evaluating a Single Function

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 the representation schemes for the domain and range. The proper specification of geometric algorithms has not received the attention it deserves. For example, published algorithms for computing set-theoretical intersections of "polygons" or "polyhedra" do not agree with the algorithms' specifications (see [67] for further discussion).

The classical methods for analyzing and comparing algorithms (see e.g. [3] and recent work on geometric algorithms, e.g. [63] and [43]) ignore the issues of robustness and extensibility, and apply mainly to algorithms which operate on a FIXED pair of input (domain) and output (range) representation spaces. The larger problem of comparing alternative "function-evaluation triples"

consisting of algorithms AND input and output representation schemes (see Fig. 2-7) 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 modelling indicates that no single object representation 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" GMS's are likely to contain MULTIPLE representations of objects, in different schemes, with each representation tailored to a specific class of applications [20].

* *

2.6 MULTIPLE REPRESENTATIONS: CONSISTENCY, EQUIVALENCE, AND CONVERSION

The presence of multiple representations (in different schemes) to support a diversity of applications introduces a strong form of redundancy in a modelling system and raises the issue of consistency (defined formally below). 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. Mechanisms for ensuring consistency usually involve the ability to convert the representation of an object in one scheme into its representation in another scheme. The design of algorithms for representation conversion is a topic of current research in geometric modelling (see e.g. [72]).

Consistency may be defined formally as follows. Let s and s' in Fig. 2-8 be representation schemes for entities in a mathematical modelling space M . We shall say that a representation r in R and a representation r' in R' are CONSISTENT if there exists an element m of M such that r belongs to the set $s(m)$ and r' belongs to $s'(m)$. Therefore consistency requires the existence of (at least) one object m having representations r and r' in the two schemes. The dashed lines in Fig. 2-8 indicate that m may have several representations in R and R' (when the schemes are not unique), and that other elements of M may correspond to r and r' (when the schemes are not complete).

FIGURE 2-8

Representational Consistency

Consistency generally is not an equivalence relation between valid representations in schemes s and s' because, while reflexive (a valid representation is consistent with itself) and symmetric (if r is consistent with r' then r' is consistent with r), it is generally not transitive (i.e., r consistent with r' and r' consistent with r'' need not imply r consistent with r'').

We define representational equivalence as follows. Two representations r and r' in schemes s and s' are **EQUIVALENT** if the sets $s^{-1}(r)$ and $s'^{-1}(r')$ are equal. Observe that consistency requires only that $s^{-1}(r)$ and $s'^{-1}(r')$ have a non-empty intersection, and therefore equivalence is a strong form of consistency. It is easy to see that representational equivalence is an equivalence relation, and that two **UNAMBIGUOUS** representations are equivalent if and only if they are consistent.

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 conversely. 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 practically important concept when dealing with ambiguous representations. In geometric modelling, for example, it is important to require that the top and front views of an object be consistent (both are ambiguous graphic representations of solids). It would be unreasonable to require that all solids exhibiting a common top view also exhibit a common front view -- that is, that the top and front views be equivalent.

REPRESENTATION CONVERSION ALGORITHMS provide the basic means for enforcing representational consistency. A conversion algorithm is a specification of a sequence of operations for computing a SYMBOL MAPPING $a : R_a \rightarrow R_a'$. The domain R_a and range R_a' of 'a' are subsets of two representation spaces R and R' which correspond to representation schemes s and s' for entities in a modelling space M . We shall distinguish between consistent (theoretically exact) and approximate conversions. Loosely, a consistent conversion produces a representation of the original entity while an approximate conversion produces a representation of an entity which approximates the original one. Consistent conversions between unambiguous representations are lossless (theoretically invertible). Formal definitions follow; the conditions in the definitions apply for all VALID representations r within R_a .

A conversion is CONSISTENT (or THEORETICALLY EXACT) when $a(r)$ is consistent with r . A conversion is APPROXIMATE when it is not theoretically exact but satisfies the following condition: there exist two elements m and m' of M such that $r \in s(m)$, $a(r) \in s'(m')$, and m' approximates m . (An entity m' approximates another entity m when their distance $d(m, m')$ in an appropriate metric is less than a positive number δ .) For example, a representation of any polygon in Scheme III may be converted into a representation of a "cellular" approximation to the polygon in the scheme illustrated in Fig. 2-2.

A conversion is LOSSLESS (or THEORETICALLY INVERTIBLE) when the set $s(s^{-1}(r))$ is the inverse image under 'a' of $s'(s'^{-1}(r'))$, where $r' = a(r)$. Figure 2-9 illustrates the definition when s and s' are unambiguous but non-unique representation schemes (this is the practically interesting case). Observe that $s(s^{-1}(r))$ is the set of all representations of a single m in scheme s . The definition implies that 'a' maps all of the representations of m in R_a into representations of m' in R_a' , and that representations of objects other than m are not mapped into representations of m' . Essentially 'a' establishes a one-to-one, and hence invertible, correspondence between m and m' . Therefore "no information is lost", i.e. m is "recoverable" from (representations of) m' . Note that m' need not equal m ; for example, m' may be the result of applying to m a known deformation to make m' easily representable in s' . This implies that a lossless conversion need not be a consistent conversion. On the other hand, a consistent conversion between equivalent

unambiguous schemes is always lossless. It should be clear also that a conversion from an unambiguous representation to an ambiguous one cannot be lossless because $s'(s'^{-1}(r'))$ is "too large".

FIGURE 2-9

Lossless Conversion between Complete but Non-Unique Representations

The interplay between representational consistency and representation conversion arises in practical systems in two slightly different guises.

- 1) When given a representation, it may be necessary to construct another consistent representation in a different scheme (e.g. to enable some property of a represented object to be computed efficiently).
- 2) When given several consistent representations, it may be necessary to update (consistently) all when one is edited.

Experience in the practical solution of these problems is limited because most existing modelling systems are strongly biased towards particular applications, and therefore do not contain multiple representations. The PADL-1 system, which possesses representations for solids in three unambiguous schemes and one ambiguous scheme, approaches the problem of consistency

maintenance as follows [71].

- 1) A complete representation (in one scheme) for each object is considered fundamental.
- 2) All other representations are derived from the fundamental representation by conversion algorithms.
- 3) Only the fundamental representation may be edited; algorithms propagate changes to the other representations.

This approach is safe but relatively inflexible. Some of the difficulties that arise when one attempts to provide more flexible means for maintaining consistent multiple representations are listed below.

- 1) It is usually impossible to update consistently an unambiguous representation to correspond to modifications made to an ambiguous representation. Direct editing of ambiguous representations therefore should be prohibited.
- 2) Conversion between representations in two unambiguous schemes is always possible theoretically PROVIDED that the schemes are equivalent. When the two schemes have different domains it is possible that editing a representation in one scheme will create an object not representable in the other. Note that the theoretical possibility of conversion does not imply that algorithms to effect such conversions are known. In practice one may be able to convert representations from a scheme s into a scheme s' but not vice-versa. (Later sections provide examples.)

* *

2.7 REPRESENTATION HIERARCHIES

It is useful to consider representations at various levels of abstraction. Figure 2-10 illustrates the notion; observe that the single representation space of Fig. 1-2 has been replaced by a hierarchy of representation spaces linked through mathematical relations (usually mappings) which we call representation SUB-SCHEMES. Formal properties of representation sub-schemes analogous to those of representation schemes may be defined easily.

FIGURE 2-10

A Representational Hierarchy

The lower-level representations in Fig. 2-10 may be viewed as "implementations" or as "refinements" of their higher-level counterparts, and thus Fig. 2-10 merely formalizes the widely accepted notion that various levels of abstraction are useful at different stages in the design and implementation of systems.

An exemplary hierarchy for representing polygons via Scheme II may be constructed as follows. Let representations at the highest level be ordered sets of real numbers; for example

$(0.0, 0.0) \quad (0.0, 1.0) \quad (1.0, 0.0)$

represents a triangle. At the second level let the ordered sets be represented by arrays containing the x and y coordinates of the vertices. (Alternatively, one could use a linked list of records, each having two fields for the point coordinates and a link field.) Finally, at the third level the arrays (or linked lists) are represented by bit patterns in certain memory locations.

The discussion in the remainder of this paper is couched in terms of high-level representations. The computational implementation of such representations is largely a straightforward programming exercise.

* * *

3. SURVEY OF REPRESENTATION SCHEMES FOR RIGID SOLIDS

The following sections discuss the known methods for representing classes of r-sets that model physical solids. First we study the various methods by discussing the properties of "pure" schemes, each based on a single method, and then we focus on hybrid schemes which combine several methods, and on the conversion between schemes.

3.1 PREVIEW

Ambiguous schemes for representing solids are discussed briefly in Section 3.2, but the bulk of Section 3 is devoted to the study of unambiguous schemes. We distinguish seven basic methods that may be used to construct unambiguous representation schemes for solids.

- 1) Pure primitive instantiation.
- 2) Spatial occupancy enumeration.
- 3) Cell decomposition.
- 4) Constructive solid geometry (CSG).
- 5) Sweeping.
- 6) Interpolation.
- 7) Boundary representation.

Pure template instantiation is based on the principles discussed in Section 2.2. All of the other methods above lead to structured schemes whereby solids are represented by aggregates of representations of primitive entities. The following is a brief description of each of the methods 2-7 and of their theoretical foundations.

SPATIAL OCCUPANCY ENUMERATION. Space is divided into a grid of 3-D cells (usually cubes), and a solid is represented by a list of the cells which it occupies, i.e., of the cells which contain "material".

CELL DECOMPOSITION. A solid is decomposed into simple solid cells with no holes and whose interiors are pairwise disjoint. A solid is the result of "glueing" component cells that satisfy certain "boundary-matching" conditions discussed in Section 3.5. Cell decomposition may be viewed as a generalization of spatial occupancy enumeration in that cells need neither lie on a fixed grid nor have pre-specified size and shape. Cell decomposition methods are based on the theory of triangulation and its extensions discussed in [58] and elementary textbooks in combinatorial (algebraic) topology (e.g. [2] and [32]).

CSG. Solids are defined as combinations of solid "building blocks" via operations akin to volumetric addition and subtraction. CSG may be viewed as a generalization of cell decomposition; the "glueing operator" of cell decomposition methods is replaced by the more powerful CSG operators which need

not satisfy "boundary-matching" conditions, may be applied to solids whose interiors are not disjoint, and may add as well as remove volume. CSG is based on the mathematics of regular sets and regularized set operators [59], [60], [58].

SWEEPING. A solid or a bounded surface moving along a trajectory sweep a "volume". Therefore a solid may be represented by a pair (moving body, trajectory). Sweeping is unrelated to the methods 2-4 above; it may be studied formally by using the mathematical notion of set product or through set operations on an infinite number of sets.

INTERPOLATION. A solid may be defined (for example) as the union of all the line segments PQ whose endpoints P and Q lie, respectively, in two given 2-D sets A and B. Therefore the solid may be represented by the pair (A,B). Interpolation rules other than the "lofting" rule just described are possible. Interpolation schemes for solids will not be discussed further because they have neither been studied theoretically nor used practically in a geometric modelling context.

BOUNDARY REPRESENTATION. While all of the methods above represent solids directly, boundary representations are indirect, in that they represent explicitly (directly) the solid's topological boundary rather than the solid itself. The theoretical basis for boundary representation lies on certain separation theorems which generalize the E2 Jordan's curve theorem to more general spaces [58].

Observe that 2-6 above essentially are standard mathematical methods for building sets from other sets. Such methods are not restricted to E3 solids; with trivial modification they may be applied also (e.g.) to surfaces and curves. Boundary representation replaces the problem of representing a set with that of representing another set of reduced dimensionality; the initial set may be a solid, as suggested above, but it may also be (e.g.) a surface or a curve. Therefore one can design schemes for representing solids which use a combination of the methods above. Such "hybrid" or "nonhomogeneous" schemes will be discussed in Section 3.9 below.

* *

3.2 AMBIGUOUS REPRESENTATIONS

We shall discuss, following semi-historical lines, a few widely-used ambiguous representation.

The traditional means for specifying mechanical parts -- 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 interpret "correctly"

drawings which contain geometric errors (e.g. extraneous lines) without noticing that such errors exist.

We know of no formal definition of drafting as a representation scheme; usually drafting textbooks state that a "sufficient" number of views, section views, details, and "notes" should be supplied in a drawing to avoid "ambiguity". This "definition" is too informal for a precise study of drafting as a representation scheme; it also makes it virtually impossible to specify and design RELIABLE algorithms for computing properties of solids represented by drawings. Nevertheless, drawings are an effective means of communication among humans and have been the major medium for geometric specification used by industry until recently. The first computer aided drafting systems, some of which are still in use, 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 one or both of the following directions.

1) The essential notion in drafting is the representation of solids by a collection of projections of solids on planes. If one wishes to retain this basic notion and seek drafting-like unambiguous representation schemes one must either

- (a) define a domain D of solids, a rule s which associates drawings (i.e. collections of projections) to solids in D , and prove that s^{-1} is a function, or
- (b) define a range V of valid drawings, and exhibit a mapping s^{-1} between drawings and solids.

Both alternatives involve proving that a single solid corresponds to each drawing. Establishing such a correspondence is a difficult problem, closely related to scene analysis studies which attempt to infer 3-D shape information from 2-D projections. It is known, for example, that it is possible to construct unambiguously the set of edges of any rectilinear (i.e. plane-faced) polyhedron (see [58] for a precise definition of rectilinear polyhedron) given a sufficiently large set of its planar projections [64], but we do not know of any results which show that reasonably general classes of curved objects can be represented UNAMBIGUOUSLY by a finite set of projections.

2) Abandon the use of projections as representations, and seek suitable collections of 3-D entities. Drafting practice suggests the use of "edges" (whose projections are the lines and curves which constitute drawings). This leads to the so-called wireframe representations wherein a solid is represented by a collection of "edges". It can be shown by counter example (see [70]) that wireframes are ambiguous representations even for the

domain of rectilinear polyhedra; an interesting algorithm for finding all the rectilinear polyhedra which have a given wireframe is reported in [46]. (Representations of rectilinear polyhedra by vertex lists also are ambiguous -- see [70].) The majority of the commercially available "3-D" geometry systems are based on (curved) wireframe representations, although some include a capability for defining certain types of "faces". Such systems cannot support reliably (for example) a fully automatic elimination of hidden lines or computation of section views.

If one goes further and associates "edges" with "faces" that cover completely an object's boundary, then one can get an unambiguous representation...as we shall see later. Thus ends a line of development which may be termed the graphics approach to solid representation.

Another line of development has led to a distinct type of ambiguous representations. The development was motivated by the following practical problem: given a PHYSICAL solid, such as a human head or a clay model of an automobile body, construct a "reasonably accurate" computer representation of the solid. The usual approach to solving this problem consists in 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 finite set of measured points as a representation of the object. The measuring operation is a "3-D digitization" that may be performed in various ways; to obtain data points in the interior of an object may involve physically slicing the object or using X-ray or ultrasonic scanning techniques.

It is clear that such finite collections of points generally are not complete 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 distinct construction methods may lead to distinct r-sets because of the ambiguity of the initial representation. An interesting algorithm for constructing boundary representations from finite sets of boundary points is described in [27].

The remainder of Section 3 is devoted to a study of unambiguous schemes for representing solids.

* *

3.3 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. It is based on the notion of families of objects, each member of a family being distinguishable by a few parameters. Such schemes are pure primitive instancing schemes (see 2.2) whose primitive instances are r-sets (i.e. solids).

Figure 3-1 provides an example: a scheme possessing three families of objects. The parameters of the family of toothed wheels having type code 1 (Fig. 3-1a) are the wheel's diameter D and the number N of equally spaced teeth. Family 2 (Fig. 3-1b) has an additional parameter D' . Family 3 (Fig. 3-1c) has two parameters: the overall pin's length L , and the pin's diameter D_p . All of the other characteristics of the objects are not specified explicitly because either (1) they are constant throughout the family (e.g. the height and the diameter of the wheels' shafts), or (2) they depend on the specified parameters (e.g. the diameter of the pin's head is proportional to the pin's diameter D_p). Figure 3-1 also shows equivalent but more readable representations in which the type codes have been replaced by family string identifiers.

FIGURE 3-1

A Pure Primitive Instanting Scheme

The distinguishing characteristic of pure primitive instancing schemes is the impossibility of "combining" existing representations to create new and more complex ones. Such schemes are akin to languages defined by grammars in which it is not possible to combine words to form sentences. Pure primitive instancing schemes therefore have relatively small domains, because large domains require an inordinate number of families and/or parameters.

The validity of representations may be checked trivially when the objects in a family are relatively simple. Thus, for example, any representation in the scheme illustrated in Fig. 3-1 is valid provided that the parameters are positive. (Note, however, that one might want to restrict the domain by requiring e.g. that $D' < D$ in Fig. 3-1b; this would add another condition to be satisfied for validity.)

Pure primitive instancing schemes may be defined so as to be both complete and unique. The scheme of Fig. 3-1, however, is not unique because $GEAR1(D,N) = GEAR2(D,D',N)$ when D' is the value of the diameter of the shaft.

In principle, pure primitive instancing schemes are concise and easy to use, and they 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. When the catalog is large the schemes become unwieldy for both humans and machines.

The other main drawback of pure primitive instancing schemes is the difficulty of writing algorithms for "processing" representations, i.e. for computing properties of the solids represented. A considerable amount of knowledge specific to object families must be built into such algorithms, and this leads to algorithms which treat each object family as a special case. Algorithms for processing "structured" representations can seek to exploit structural knowledge in order to process structured representations "uniformly", rather than on a case by case basis.

We suspect that the pure primitive instancing schemes now used in industry will be replaced by the more powerful structured schemes discussed below, because these latter provide the same "part family" capabilities through "generic" or "macro" structured object representations (see Sections 2.2 and 2.4.2).

* *

3.4 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 which lie on a fixed spatial grid. Each cell may be represented by the coordinates of a single point, e.g. the cell's centroid. (For simplicity we shall discuss only the cubical grid, although others can be used, because the type of grid does not affect the basic properties of the schemes.)

A representation in a spatial occupancy enumeration scheme is simply a (possibly unordered) set of three-tuples. A specific spatial scanning order is convenient for a variety of practical reasons, and therefore spatial enumerations usually are ordered sets, called SPATIAL ARRAYS. The semantics of such representations is the obvious one: a cell corresponds to each three-tuple, and the solid represented is the union of all such cells. The representation scheme is defined fully since we defined its range and its association rule.

The domain D is a subset of the space of rectilinear polyhedra. A representation is valid provided that the three-tuples correspond to centroids of cells, and therefore validity may be established trivially.

It is clear from the definitions that the representation schemes are unambiguous. They also are unique except for permutational and positional non-uniqueness (see Section 2.4.4). Spatial arrays do not exhibit permutational non-uniqueness because of the implied spatial scanning rule.

Spatial arrays are potentially quite verbose, but verbosity depends largely on how well the class of physical objects of interest matches the domain of polyhedra representable by spatial occupancy enumeration. Thus, in certain architectural applications buildings are sufficiently "modular" for one to use relatively short spatial array representations based on a coarse spatial grid [45]. Spatial arrays also appear to be a reasonable representation for r-sets which provide approximate models for "irregular" biological objects such as those encountered in computed tomography; the representations may be created directly by measuring equipment operating on the physical objects. (Spatial occupancy enumeration is one of the methods for constructing an unambiguous representation from a finite-set-of-points ambiguous representation -- see Section 3.2.) For such objects as mechanical parts, spatial occupancy enumeration is too verbose for practical use as a "master" definitional medium. (Ponder, for example, the problem of representing an r-set that approximates, to within 0.001", a cylindrical pin with a diameter of 0.5" and a length of 2".) However, spatial-array representations of objects which are COARSE APPROXIMATIONS of mechanical parts can be used effectively, e.g. to improve the average performance of geometric algorithms.

Spatial occupancy enumeration shares certain interesting

properties with other schemes that are based on the decomposition of solids into cells with non-intersecting interiors. These properties will be discussed at the end of Section 3.5 in the context of cell decomposition schemes.

Schemes have been proposed which are hybrid combinations of a restricted form of CSG and spatial occupancy enumeration [56]. Such schemes are similar to the "quad-tree" representations (see e.g. [23]) used in image processing. They avoid some of the verbosity of pure spatial occupancy enumeration and have other practical advantages as well.

* *

3.5 CELL DECOMPOSITIONS

A solid triangulation of a rectilinear polyhedron is a decomposition of the polyhedron into tetrahedra which are bounded by triangles, which in turn are bounded by edges whose endpoints are the vertices of the triangulation. (Triangulations are discussed in precise mathematical terms in [58], and in topology texts such as [2] and [32].) The decomposition must satisfy the following condition: any two tetrahedra in the triangulation must either be disjoint or must meet precisely at a common triangle, edge or vertex. The condition implies that tetrahedra must not interpenetrate, and that they be "glued" only at common triangles, edges, or vertices. Curved polyhedra also may be triangulated by decomposing them into "curved tetrahedra" which satisfy a condition analogous to that above.

Cell decompositions are a generalization of triangulations wherein cells may have an arbitrary number of "faces". The definitions are similar, but tetrahedra and triangles are replaced by 3-cells and 2-cells, respectively. A 3-cell is a set topologically equivalent to a convex rectilinear 3-D polyhedron (e.g. an octahedron); 3-cells are bounded by an arbitrary but finite number of 2-cells. A 2-cell is a set topologically equivalent to a simple polygon (as defined in Section 2.3); 2-cells are bounded by an arbitrary but finite number of edges. Note that cells may not have "holes"; thus, for example, a solid torus is not a 3-cell. Cell decompositions are more convenient than triangulations for practical purposes because the number of cells in a cell decomposition usually is much smaller than the number of tetrahedra in a triangulation of a solid.

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

Cell decomposition schemes are potentially capable of representing any domain of r-sets because all r-sets are

triangulable [58] [36]. The domain represented by a specific cell decomposition scheme is determined by the types of cells allowed by the scheme.

The validity of cell-decomposition representations of solids is difficult to establish. The problem is essentially the 3-D analog of the validity problem for cell decompositions of boundaries which will be discussed in Section 3.8.2 below.

Cell decompositions (and hence triangulations) are obviously unambiguous representations of r-sets, but they are not unique because each r-set admits many distinct cell decompositions.

Cell decomposition representations generally are neither concise nor easy to create. Valid cell decompositions of CURVED solids are hard to construct directly by humans, and algorithms for constructing such representations from others are a topic of current research [5].

Cell-decomposition representations provide convenient means for computing certain topological properties of objects. For example, there are known algorithms for determining whether a solid is "one-piece" (i.e. connected), whether it has "voids", "holes", and so forth [58].

Cell decompositions, spatial arrays (Section 3.4), and certain restricted forms of CSG (Section 3.6) are based on decomposing a solid into components whose interiors are pairwise disjoint. Because objects are defined as UNIONS of cells, an object is null if and only if all cells correspond to empty sets. (As noted in Section 2.4.4 tests for object nullity are helpful for establishing object equality.) Cell disjointness facilitates the computation of a solid's volume, moments of inertia, and other properties which involve volume integrals over the solid [44]. (Because the cells have disjoint interiors the contribution of each cell may be computed separately.) Finally, disjointness plays a key role in the modelling of non-homogeneous objects -- an important topic (e.g. in finite element analysis) that is beyond the scope of this paper.

Solid triangulation schemes have been used for the internal representation of rectilinear polyhedra in the GMS described in [37], and also in an algorithm for computing the volume of convex rectilinear polyhedra [22]. Cell decompositions are the primary representations used in 3-D (solid) finite element analysis.

* *

3.6 CONSTRUCTIVE SOLID GEOMETRY

Constructive solid geometry (CSG) connotes a family of schemes for representing rigid solids as "constructions" or "combinations" (via the regularized set operators) of solid components [59]. CSG and boundary representations (Section 3.8) are the best understood and currently most important representation schemes for solids, and therefore are treated in this survey more thoroughly than the other known schemes. Section 3.6.1 discusses the tree structures characteristic of CSG representations, and Section 3.6.2 discusses the primitives which correspond to some of the CSG tree leaves. Formal and informal properties of CSG schemes are discussed in Sections 3.6.3 and 3.6.4. We conclude by discussing briefly the use of CSG-related representations in GMS's.

* *

3.6.1 CSG TREES

CSG representations of objects are (ordered) binary trees whose non-terminal nodes represent operators, which may be either rigid motions (translations and/or rotations) or regularized set operators (regularized union, intersection and difference, denoted respectively by $\cup$, $\cap$, and $-$), and whose terminal nodes (leaves) may be either primitive leaves which represent subsets of E^3 , or transformation leaves which contain the defining arguments of rigid motions. CSG trees may be defined by the following BNF.

```
<CSG tree> ::= <primitive leaf> |
               <CSG tree> <set-operator node> <CSG tree> |
               <CSG tree> <motion node> <motion arguments>.
```

(In certain CSG schemes, e.g. that embodied in PADL-1 [71], subtrees may be shared and therefore representations are acyclic graphs rather than trees; data sharing is unimportant for the purposes of this paper and will be ignored.)

The semantics of CSG-tree representations is obvious (see Fig. 3-2): each subtree that is not a transformation leaf represents a set which is the result of applying the indicated motional and combinational operators to the sets represented by the primitive leaves. (An ordering convention is needed for associating a meaning to nodes which contain non-symmetric operators such as $-$.) Rigid motions are well understood and therefore motion nodes and transformation leaves need not be discussed further. Regularized set operators are modified versions of their usual counterparts; they are discussed at length in [59], [58], and [60]. Primitive leaves are dealt with in the following section.

FIGURE 3-2

A CSG Tree and the Solids Represented by its Subtrees
(Solids are shown in orthographic projection.)

*

3.6.2 PRIMITIVE LEAVES

The primitive leaves of a CSG tree are tuples which represent subsets of E^3 . Each CSG scheme has a fixed set of generic primitives (see Section 2.2) whose instances are represented by primitive leaves. The generic primitives are called (generic) HALF-SPACES. It is useful to distinguish two types of half-spaces, simple and composite, defined as follows.

Let $F(Q, p)$, where $Q = (q_1, q_2, \dots, q_N)$ is a parameter vector and $p = (x, y, z)$ is a generic point of E^3 , be a real-valued analytic function in E^3 for each value of the parameter vector Q in some set D_q . The parameters $q_1, q_2, \dots$ are called CONFIGURATION PARAMETERS, Q is a CONFIGURATION VECTOR and D_q is a CONFIGURATION DOMAIN. For each fixed value of Q in D_q the set

$$I(Q) = k\{p : F(Q, p) < 0\},$$

where k and i denote closure and interior in the usual E^3 topology, is a regular semi-analytic set [58]. Now let us construct a family of sets as follows. For each value of Q in D_q apply to $I(Q)$ a rigid motion that belongs to a set D_p of motions

called a POSITION DOMAIN. The family S of sets which results when the construction above is repeated for all values of Q in D_q , and for all motions in D_p is called a GENERIC SIMPLE HALF-SPACE, and the members of the family are called SIMPLE HALF-SPACE INSTANCES.

Because a rigid motion of $E3$ may be defined by a POSITION VECTOR $P = (dX, dY, dZ, aX, aY, aZ)$, where dX , dY , and dZ are displacements that define a translation, and aX , aY , and aZ are angles that define a rotation, a simple half-space instance corresponds to a specific choice of values for the configuration and position vectors Q and P . Instancing may be viewed as a two-step process. Firstly one "configures" the half-space by assigning values to the configuration vector Q , and then one "positions" the configured half-space by assigning values to the position vector P . The configuration and position vectors, together with their respective domains, are said to provide a PARAMETERIZATION for the family of sets which constitute a generic half-space.

An example:

$$F(R, p) = x^2 + y^2 - R^2,$$

where R is the single-component configuration vector, is an algebraic (and hence analytic) function for R in the configuration domain D_q defined by $R > 0$. The set

$$I(R) = \{ (x, y, z) : x^2 + y^2 - R^2 < 0 \}$$

is an unbounded solid cylinder of radius R whose axis coincides with the z -axis of $E3$. If we let D_p be the set of all rigid motions of $E3$, $F(R, p)$ together with D_q and D_p defines a cylindrical simple half-space as the collection of all unbounded solid cylinders of $E3$. Each instance is a cylinder with a specific radius placed at a specific location in $E3$.

Simple half-spaces may be combined via regularized set operators and used to construct GENERIC COMPOSITE HALF-SPACES by a procedure analogous to that described above. Figure 3-3 illustrates the notion. A generic bounded cylindrical composite half-space is the family of sets congruent to sets of the form

$$H = S1 \cap S2 \cap S3,$$

where $S1$ and $S2$ are instances of a planar simple half-space and $S3$ is an instance of a cylindrical simple half-space, in the relative positions shown in the figure.

FIGURE 3-3

**An Instance of a Cylindrical Composite Half-Space
Defined as the Intersection of Three Simple
Half-Space Instances**

A generic composite half-space can also be parameterized in terms of configuration and position parameters. The following is an adequate parameterization for the example shown in Fig. 3-3: the configuration vector is (R, H) with a configuration domain defined by $R > 0$ and $H > 0$, and the position vector is a 6-tuple that defines a rigid motion in the position domain of all rigid motions of E^3 . Observe that the configuration and position parameters of the simple half-spaces used to define a composite, together with appropriate domains, provide a (perhaps non-minimal) parameterization for the composite.

Half-spaces are primitives of CSG schemes. Each CSG scheme possesses an associated finite collection $\{H_i\}$, $i=1, \dots, M$, of (simple or composite) generic PRIMITIVE HALF-SPACES. Instances of half-spaces in $\{H_i\}$ are represented by tuples which contain a type code that designates a particular H_i , and contain also the values of configuration and position vectors Q_i and P_i . (The discussion above shows that such a parameterization is always possible.)

We shall distinguish two types of CSG schemes. The first type -- called CSG BASED ON GENERAL PRIMITIVE HALF-SPACES -- has unrestricted $\{H_i\}$, some of which may be unbounded. The second type -- called CSG BASED ON PRIMITIVE SOLIDS, or simply CSG because it is the most common and useful of the two -- possesses a set of H_i which are BOUNDED and therefore are r-sets. The second type is a particular case of the first but has different properties, as we shall see below.

*

3.6.3 FORMAL PROPERTIES

The domain of CSG representation schemes is determined by the following.

- 1) The specific set of primitive half-spaces $\{H_i\}$. (Recall that a generic half-space definition includes a specification of its configuration and position domains.)
- 2) The available motional operators.
- 3) The available regularized set operators.

The domain of a CSG scheme with primitive half-spaces $\{H_i\}$ is always a subset of the REGULARIZED BOOLEAN CLASS defined by $\{H_i\}$. The regularized Boolean class contains all of the sets definable by a finite number of motions and regularized set operations (complement excluded) on arbitrary instances of the half spaces H_i . (Observe that such a class of sets is mathematically well defined and independent of the means used to represent the sets.) The domain of a CSG scheme equals the regularized class defined by its $\{H_i\}$ when (1) the motional operators are general, i.e. any rigid motion of E^3 is allowable, (2) the three regularized set operators U^* , $\cap^*$ and $-^*$ are available, and (3) both the motional and combinational operators are general, i.e. applicable to any object representable in the scheme. When the primitives are composite rather than simple half-spaces usually the domain is determined by the set of SIMPLE half-spaces $\{S_j\}$ which underlie the $\{H_i\}$ (see Section 3.6.2). Thus the two representation schemes with general motional and combinational operators and primitives depicted in Figs. 3-4a and 3-4b have the same domain, which is the regularized Boolean class defined by planar and cylindrical simple half-spaces.

FIGURE 3-4

Two CSG Schemes Having Different
Primitives but the Same Domain

Every CSG tree whose primitive leaves are valid representations of half-spaces is guaranteed to define a single regular semi-analytic subset of E^3 . (This follows immediately from the algebraic closure of regular semi-analytic sets [58].) Primitive validity is ensured when the configuration and position vectors are within their specified domains D_q and D_p . For example, a cylindrical half-space instance in our earlier example is valid if and only if the half-space's radius is positive.

The primitives of CSG schemes based on general half-spaces need not be bounded, and therefore a set defined by a CSG tree may be unbounded (and hence not an r-set). It follows that primitive validity does not suffice to ensure validity of a composition in such schemes. Boundedness of compositions is difficult to verify. The only viable approach known to us involves re-representing the set via its boundary (see Section 3.8), and testing for "face" boundedness. This is computationally expensive and requires non-trivial algorithms.

Because primitive solids are bounded, and hence r-sets, the algebraic properties of r-sets guarantee that ANY CSG tree on primitive solids is a VALID representation of an r-set when the primitive leaves are valid. Because in practice primitive leaf validity is satisfied trivially as noted earlier, the validity of CSG representations based on solid primitives may be ensured essentially at the syntactical level. Thus any representation that satisfies the syntactical rules described by the BNF of Section 3.6.2 is valid if its leaves are valid.

It is important to realize that the "guaranteed validity" of CSG schemes based on solid primitives applies only to CSG 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 3.5); expensive computations may be required to ensure that the conditions for operator applicability are not violated. It is worth noting that, in practice, restrictions are placed on operators largely to facilitate "boundary evaluation", i.e. the computation of the boundaries of the objects represented in the scheme; this goal is defeated if one seeks to provide automatic validity testing because the computations required for validity testing and (general) boundary evaluation have comparable complexities.

CSG representations are obviously unambiguous since both rigid motions and regularized set operations produce single well-defined sets.

CSG representations are not unique because generally it is possible to construct the same solid in many ways. For example, the solid of Fig. 3-2 may be defined alternatively as the

difference of two blocks.

To determine whether two objects are equal one may represent in the CSG scheme the regularized symmetric difference of the two objects, and test the difference for nullity (see Section 2.4.4). However, null tests appear to be expensive, as do approaches based on re-representing the difference object via its boundary.

It is worth remarking that operators in a CSG tree provide non-redundant information: generally they may not be inferred from a knowledge of the half-space instances, as the following discussion will show.

Consider a collection of pairs (bH_i, N_i) satisfying the following conditions.

- 1) bH_i is the boundary of an instance of a primitive half-space H_i .
- 2) Each bH_i has a subset G_i which is a 2-D polyhedron (i.e. has an "area") and lies on the boundary of an object S .
- 3) The union of all of the G_i is the object's boundary.
- 4) N_i indicates the "material side" of bH_i with respect to the object. (N_i could be, for example, an inward unit normal vector.)

A collection $\{(bH_i, N_i)\}$ with such properties does not suffice to define an r-set unambiguously: see Fig. 3-5, which shows two distinct solids possessing the same collection of pairs (bH_i, N_i) . Therefore a list of (possibly unbounded) "oriented surfaces" does not suffice to define solids unambiguously, and a fortiori neither does a list of "un-oriented surfaces".

FIGURE 3-5

Two Solids Possessing the Same "Oriented Surfaces"

Work reported in [52] shows that a collection of pairs (bH_i, P_i) , where P_i is a point which belongs to both bH_i and the object's boundary, also does not define a solid unambiguously. We conclude that approaches based on lists of "surfaces", oriented or not, augmented or not with "points on the object's

boundary" lead to ambiguous representations of solids. We shall see in Section 3.8 below that boundary-based unambiguous representations of solids require that the boundary itself be represented unambiguously.

*

3.6.4 INFORMAL PROPERTIES

CSG schemes whose primitives are well matched to the domain of objects to be represented are very concise. Schemes based on primitive solids usually are more concise than those based on general half-spaces, because the latter usually have "lower-level" primitives. For example, the object represented via three primitive "blocks" in Fig. 3-2 would require eight primitive instances in a scheme whose primitives are planar simple half-spaces.

The limited experience accrued to date in implemented CSG schemes indicates that humans create easily CSG representations of certain types 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 interaction operations (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. the volume) of objects [29] [30] [44]. Some of these algorithms are slow in sequential machines but are inherently parallel, and therefore are promising candidates for hardware implementation. It appears that CSG representations will be of considerable importance for manufacturing automation, e.g. in the study of rough machining operations.

CSG is a powerful and general descriptive medium. The majority of GMS's implemented or designed recently use some form of CSG, as we shall see in the following section.

*

3.6.5 USE OF CSG AND RELATED SCHEMES IN GMS'S

Spatial arrays (Section 3.4), cell decompositions (Section 3.5), and certain schemes used in architecture which are based on "block" primitives [47] 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.)

CSG schemes possessing a single but general union operator and a single primitive have been used in computer graphics and

computer vision. Specifically, the scheme described in [33] has a single ellipsoidal primitive while the scheme described in [6] has a single spherical primitive. Schemes possessing a single associative operator (e.g. union) obviously do not need to represent the operator explicitly, and thus CSG trees in such schemes may be collapsed into lists or sets of primitive leaves.

Many GMS's which use a boundary representation scheme as their primary or "archival" definitional medium also offer CSG-like input facilities. For example, there may be a "union command" which, when executed, produces a boundary representation of the union of two boundary-represented solid arguments. The majority of such systems, at their present stage of development, cater only for rectilinear polyhedra. Systems with these characteristics (insofar as we know) include EUCLID [9], EUKLID [26], GEOMAP [38], GEOMED [8], GLIDE [24], 3DFORM [74], a system developed at the University of Michigan [12], and two systems developed at Ohio State University [53], [75]. Systems which cater also for curved objects having certain types of quadric surfaces are BUILD-1 [16] [17], BUILD-2 [18], COMPAC [65], [66], GIPSY [61] and PROREN-2 [62].

Several GMS's use a CSG-related representation as their primary definitional medium. SHAPES [41] [42] and TIPS-1 [50] [51] are based on generally unbounded primitive half-spaces, while GDP [31] [73], GMSOLID [13] [15], PADL-1 [69] [71], and Synthavision [29] [30] are based on bounded solid primitives. GMSOLID and PADL-1 use CSG representations of the form discussed in Sections 3.6.1-3.6.4; the other systems just listed use somewhat different representations. For example, TIPS-1 uses 3-level structures rather than binary trees; TIPS-1 representations are of the form

$$\text{OBJECT} = (A_1 \cup A_2 \cup \dots) - (B_1 \cup B_2 \cup \dots),$$

where the A_i and B_i are intersections of primitive half-spaces or their complements.

Set operator generality has a strong impact on the characteristics of CSG schemes (e.g. validity -- see Section 3.6.3), and is important also in the context of manufacturing automation applications [70]. GMSOLID, PADL-1, and (apparently) BUILD-2 possess regularized set operators which are applicable to any object representable in the systems. The literature on the other systems cited in this Section does not allude to regularity issues, and therefore many of the systems are likely to contain operators that either cannot be applied to certain objects or yield non-solid (invalid) results. (Non-solid objects may be created when, for example, non-regularized intersection is applied to two solids having partially coincident, overlapping boundaries [59].)

3.7 SWEEP REPRESENTATIONS

The basic notion embodied in sweep representation schemes is very simple: a set moving through space may sweep a "volume" (a solid) that may be represented by representing the "moving object" plus the "trajectory".

Sweeping may be formulated rigorously in terms of either set products or unions of an infinite number of sets, as we shall see below. Readers are warned, however, that much less is known about sweep representations than about the other schemes discussed in this paper. The underlying theory has not been worked out fully, but there is no doubt that the conclusions below apply for "well-behaved" sets.

The following section studies two simple and useful forms of sweeping. Section 3.7.2 deals with a type of sweeping important in manufacturing automation. Finally, general sweeping is discussed briefly in Section 3.7.3.

* *

3.7.1 TRANSLATIONAL AND ROTATIONAL SWEEPING

Firstly we consider a TRANSLATIONAL SWEEPING scheme illustrated in Fig. 3-6. Let A be a well-behaved bounded set lying on a plane (A is a "solid curved polygon" which may contain "holes"), and let B be a bounded straight line segment normal to the plane of A and having an endpoint on the plane. Construct the set product $C = A \times B$. (Recall that the elements of C are ordered pairs (a,b) , where ' a ' is an element of A and ' b ' is an element of B .) C is an abstract space; to obtain a solid S we must IMMERSE (or place) C in E^3 by constructing a mapping, called an IMMERSION, $i: C \rightarrow E^3$. The solid is the image of C under ' i ', i.e. $S = i(C)$.

The immersion used by translational sweeping schemes is the natural one illustrated in Fig. 3-6. Assuming, for simplicity, that B lies on the Z axis and A lies on the XY plane, the immersion mapping is defined per $i(a,b) = (x,y,z)$, where $(x,y,0)$ are the coordinates of ' a ' and $(0,0,z)$ are the coordinates of ' b '. (Sweeps along a direction oblique to the plane of A may be defined in a similar manner, and have essentially the same properties as orthogonal sweeps.) S is the volume swept when all points of A are translated along a trajectory parallel to B . Alternatively, and equivalently, S may be defined as the union, for all ' b ' in B , of the sets $A_b = T_b(A)$ where T_b is a translation by ' b ' along the direction of the segment B .

FIGURE 3-6

Translational Sweeping

Clearly S may be represented by a pair (representation of A , representation of B); the latter is trivial and therefore the problem of representing a 3-D solid S is reduced to that of representing a 2-D set A . This is done usually by means of a 2-D boundary representation scheme, i.e. by representing the bounding edges of A .

The domain of a translational sweeping scheme is restricted to solids exhibiting "translational symmetry". A representation of S is valid when A 's representation is valid, and therefore validity may be tested in 2-D. A representation is unambiguous when A is represented unambiguously. Translational sweep representations usually are not unique; Figs. 3-7b and 3-7c show two translational sweeps which correspond to the parallelepiped shown in Fig. 3-7a.

The definition of rotational sweeping schemes is similar to that of translational sweeping schemes. Fig. 3-8 illustrates the notion. A set S is defined as $S = i(A \times B)$, where 'i' is the "natural" immersion, A is a 2-D set as before, but B is an arc of a circle rather than a line segment. The solid S is the volume swept when A is rotated about the Y axis through an angular interval defined by the arc B . Usually A is represented by its bounding edges. (The rotational sweeping scheme described in [11] is an exception: the 2-D set is represented by a cell decomposition.)

Non-Uniqueness of Translational Sweep Representations

FIGURE 3-7

SWEEP
Z
B
b

FIGURE 3-8

Rotational Sweeping

The domain of rotational sweeping schemes is restricted to solids having "rotational symmetry". The properties of rotational sweeping schemes are essentially those of translational sweeping schemes.

Rotational sweeping is used by many GMS's which deal exclusively with turned (lathe) parts. Translational sweeping is used in systems for describing flat sheet metal parts. Both types of sweeping are used as a means to enter object descriptions in GMS's based on boundary representations (e.g. in the CADD system marketed by McDonnell-Douglas, St. Louis, MO, and in the ROMULUS system marketed by Shape Data, Cambridge, U.K.). The representation of 2-D sets by bounding edges leads to convenient interfaces for traditionally-trained draftsmen.

*

3.7.2 VOLUMES SWEEPED BY MOTIONS OF A SOLID

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

- 1) 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 [14].
- 2) 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 [68] [70].

A representation scheme based on "solid sweeping" may be constructed as follows. Let A be an r -set that models a rigid solid. It is well-known from Classical Mechanics that a rigid solid has six degrees of motional freedom. The coordinates (x, y, z) of the solid's centroid (or another agreed point of the solid) plus three orientation angles (a_1, a_2, a_3) measured in some agreed manner (e.g. the Euler angles) suffice to describe completely the position (location plus orientation) of the solid at each instant of time. Therefore a solid's position corresponds to a six-tuple of parameters (x, y, z, a_1, a_2, a_3) , i.e. to a point in 6-D space. The motion of any specific solid is completely determined by the temporal evolution of its representative point in 6-D space. To define a swept volume we let B be a curve in 6-D space, and we construct the product $A \times B$. Finally we immerse the 9-D space $A \times B$ in E_3 by a mapping 'i' defined as follows. Let 'a' be a point of A , and let $b = (x, y, z, a_1, a_2, a_3)$ be a point of B . Observe that to each point 'b' corresponds a set A_b congruent to A but at a position defined by the six-tuple 'b' (see Fig. 3-9); denote by T_b a rigid transformation of E_3 which takes A onto A_b , i.e., such that $A_b = T_b(A)$. Then $i(a, b) = T_b(a)$, i.e., a point 'a' is mapped onto its "corresponding" point when the solid is in the position

defined by 'b'. Note that the immersion 'i' is not one-to-one. $S = i(A \times B)$ equals the union of the A_b sets for all 'b' in B, i.e., S is the union of all of the volumes which the moving object occupies at each instant of time during its motion.

FIGURE 3-9

Mapping the Points of A Onto Points of the Volume Swept By the Motion of A

The domain of solid sweeping schemes depends on the solids and trajectories available and is difficult to characterize in other terms. The formal properties of solid sweeping schemes are essentially those of translational and rotational sweeping schemes, but validity may not be tested in 2-D since A is a 3-D solid.

The completeness of solid sweeping schemes implies that a cutter trajectory specified by a part program written in a language such as APT [40], 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 manufacturing it. The major disadvantage of such a representation is the lack of known algorithms for computing properties of represented solids.

*

3.7.3 GENERAL SWEEPING

Very little is known about general sweep representations. Figure 3-10 shows that sweeping a homogeneously 1-D polyhedron along another may produce a set which is not homogeneously 2-D. (Similar examples may be constructed in 3-D.) Mathematical

conditions on the "component" sets A and B, and on the immersion mapping 'i' for ensuring that the resulting set $i(A \times B)$ is an r-set do not appear to be known.

FIGURE 3-10

General Sweeping May Produce Non-Regular Sets

The three examples of sweeping discussed in Sections 3.7.1 and 3.7.2, as well as the example depicted in Fig. 3-10 amount to moving a rigid object (not necessarily a solid) through space. General sweeping may involve also the motion of a non-rigid object which undergoes deformations as it travels through space. Non-rigid sweeping schemes may be defined as in Section 3.7.2 by using transformations T_b that are not rigid motions. Binford and others have implemented modelling systems which use so-called "generalized cylinders" or "generalized cones" [1] [10]; such systems are based on non-rigid sweeping schemes. Generalized cones appear to be quite useful for describing objects such as airplanes, animals and dolls which may be naturally decomposed into a few "tube-like" components.

General sweep representations also are used extensively in a recent proposal for an ANSI standard representation of solid objects [4].

* *

3.8 BOUNDARY REPRESENTATIONS

Boundary representations of solids are familiar to most computer scientists because they are used extensively in computer graphics [28] [29]. 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 3-11 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, i.e. the segmentation of an object's boundary into

non-overlapping triangles. It is clear that any rectilinear polyhedron may be represented in such a scheme.

FIGURE 3-11

A Boundary Representation for a Rectangular Pyramid

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

* *

3.8.1 DESIGN ISSUES

Most people have the same intuitive notion of "face" for rectilinear polyhedra. Observe, however, that the faces in the representation of Fig. 3-11 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 are the faces of CURVED objects: see Fig. 3-12. The examples of Figs. 3-11 and 3-12 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 shall propose certain characteristics which reasonable definitions of "face" should possess.

FIGURE 3-12

What Are the Faces?

Firstly we shall assume that each boundary representation scheme has a finite number of generic PRIMITIVE SURFACES bH_i which are the boundaries of regular semi-analytic half-spaces H_i . (To avoid certain mathematical subtleties we shall assume in the sequel that the bH_i are 2-manifolds without boundaries [2].) Faces should satisfy the following conditions.

- 1) A face of an object is a subset of the object's boundary.
- 2) The union of the faces of an object equals the object's boundary.
- 3) Each face is a subset of some primitive surface bH_i .
- 4) A face must be a homogeneously 2-D topological polyhedron [2], i.e., it must have "area" and must not have "dangling edges" or "isolated points".

In particular schemes, faces may be required to satisfy additional, non-essential conditions such as connectedness, simple connectedness, maximality (in the set inclusion sense) or quasi-disjointness. Thus, in most boundary schemes used in

computer graphics faces must be quasi-disjoint, connected and maximal while in the boundary scheme used in PADL-1 [71] none of these properties are required. An additional condition often imposed on faces is that each face possess a "constant normal". Figure 3-13 shows that faces with non-intuitive properties may result if this condition is ignored. Faces are defined in the scheme illustrated in the figure as maximal subsets which satisfy conditions 1-4 above; the (disconnected) hatched face of Fig. 3-13a exhibits "opposing normals", and so do the (connected) hatched faces of Fig. 3-13b. (These last exhibit another anomaly: they intersect on a line which is not a bounding edge of either face.)

FIGURE 3-13

Subsets of a Single Surface May Possess Opposing Normals

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 Fig. 3-11, but non-planar faces require that their "host surface" bHi also be represented, typically by primitive instancing. Figure 3-14 shows that additional information may be needed to represent unambiguously faces which lie in certain surfaces. (To distinguish between the shaded spherical triangle of Fig. 3-14 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 [7] for an entry into the extensive literature on the subject. Some of the techniques amount to defining the patches' edges together with an interpolation rule, while other techniques are based on approximating sets of curves with specific types of curved surfaces. Surface patches also may be defined via a collection of cross-sections plus an interpolation rule [48], or by sweeping a curve along another [35].

FIGURE 3-14

A Face Lying in a Spherical Surface

Face representation via edges is the 2-D analog of solid representation via faces, and edges are representational artifacts in the same sense that faces are representational artifacts. Thus the entity "edge" must be defined mathematically and appropriate edge-representation schemes must be designed.

Figure 3-15 illustrates some of the issues raised by edge representation. The entire intersection of the two cylindrical surfaces may be represented unambiguously by "pointing" to the surfaces, i.e. as a pair $(\text{Surf1}, \text{Surf2})$. However, subsets of the intersection generally are not represented unambiguously by their end points together with the surface pair. A common solution to this problem involves segmenting the intersection. Observe in the figure that removal of the "critical points" where the two ellipses intersect separates the intersection of the two cylinders into four connected components, each homeomorphic to an open line segment and representable unambiguously by a triple $(\text{Surf1}, \text{Surf2}, P)$ where P is some point in the interior of the component. If one defines edges as curve segments that are homeomorphic to closed line segments and lie entirely within the closure of an intersection-curve component, it is clear that edges can be represented unambiguously by their endpoints together with the triple which represents its "host" component.

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 completely a solid's boundary (and hence the solid), most schemes store with each face the direction of an outward normal to the face because such information is easy to store and is useful for generating graphic displays.

FIGURE 3-15

Endpoints P1 and P2 Bound Four Curved Segments
(1-simplices) of the Intersection of
Two Cylindrical Surfaces

Other redundant information takes the form of adjacency relations between faces, edges, and vertices; such information sometimes is stored to avoid expensive searches that otherwise would be required by certain algorithms. One of the most elaborate schemes containing explicit adjacency information is that described in [8]. 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 discussion of such matters would take us too far afield.

*

3.8.2 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 whose primitive surfaces are (a subset of) $\{bH_i\}$.

Boundary representation schemes are unambiguous (complete) 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 [58]. (General (non r -set) subsets of E^3 are not defined unambiguously by their boundaries.) Uniqueness, except for permutational and positional non-uniqueness, may obtain in particular schemes. For example, faces that are maximal and have properties 1-4 lead to unique representations in schemes whose primitive half-spaces have the following "non-overlap" property: when $bH_i \cap bH_j$ is a 2-D polyhedron then either $H_i = H_j$ or $H_i = c^*H_j$.

The validity of boundary representations raises interesting issues which we shall discuss in the remainder of this section. To study validity one proceeds "bottom-up", e.g. by investigating conditions for vertex nodes to represent points of E^3 , for edge sub-graphs (i.e. edge nodes plus their corresponding vertex nodes) to represent line segments, and so on. Validity conditions may be classified into two generic categories, "combinatorial" and "metric". We shall present the validity conditions for two exemplary schemes. (Such conditions follow directly from elementary algebraic topology [58] [2].)

Scheme A: the triangulation scheme illustrated in Fig. 3-11. 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 faces form a 2-cycle and that each face's edges form a 1-cycle [58] [2]; intuitively, they imply that the surface "closes" and that the edges form "loops". It is not difficult to write efficient algorithms for testing the combinatorial conditions above, or to imbed such conditions in the procedures which construct and update the representations. The so-called "Euler operators" which are used in some of the GMS's based on boundary representation schemes more elaborate than Scheme A [8] [19] [25] appear to play the role of ensuring that combinatorial validity 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.)

Satisfaction of the foregoing combinatorial conditions does not ensure that representations in Scheme A are valid. Mathematically, the problem is the following. A combinatorial structure (as in Fig. 3-11) that satisfies conditions 1-4 defines a combinatorial entity called an "abstract complex". Metric information associated with the entities of the representation must be compatible with the combinatorial structure in the following sense: there must exist a sub-polyhedron of E3 with a triangulation whose faces, edges, and vertices may be put in a one-to-one correspondence with those of the abstract complex. Therefore, the problem is that of constructing a realization of an abstract complex in E3. The following metric conditions must be satisfied.

- 5) Each three-tuple of vertex coordinates must represent a distinct point of E3.
- 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 [8], [18], [24], who advocate the "separation of topology from geometry" in GMS's. 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. Observe in Fig. 3-16 that a change of coordinates of the vertex P converts a valid representation in Scheme A which corresponds to the object of Fig. 3-16a into an invalid representation, because the faces of the object of Fig. 3-16b intersect in lines which have no corresponding edge nodes in the combinatorial structure. (For clarity, triangulations of the rectangular faces in Fig. 3-16 are not shown.) Of course the solid indicated in cross-section in Fig. 3-16c could be represented in Scheme A in a valid manner by segmenting the "self-intersecting faces" of Fig. 3-16b, but this would alter extensively both the combinatorial structure and the metric information of the initial representation.

FIGURE 3-16

**Illustration of the Interdependence of
Combinatorial Structure and Metric Information**

Scheme A may be generalized in several directions. The following is a simple generalization whose validity conditions follow directly from existing mathematical results [58].

Scheme B: Faces are simple planar polygons without holes (see Section 2.3) rather than triangles, but otherwise the definition of Scheme B coincides with that of Scheme A. This leads to cell-decompositions rather than to triangulations of the boundary. Validity condition 1 does not apply, but conditions 2-7 are applicable and the following additional conditions are required.

- 4a) The union of the edges in a face must be a connected set. Together with condition 4 this ensures that a face's edges form a "single loop" and therefore (by Jordan's curve theorem [2]) that the face is a cell if the metric conditions 5-8 are satisfied.
- 8) All of the vertices in a face must lie in a plane.

Schemes A and B cover only rectilinear polyhedra. Curvature introduces a variety of interesting and delicate validity issues. For example, it is no longer sufficient to show that the edges associated with a face node form a loop (1-cycle) to prove that they, together with host surface information, represent a face unambiguously. Figure 3-17 shows that closed curves lying in a torus need not bound a face. Such curves are called "non-bounding cycles" in algebraic topology. (The number of distinct types of non-bounding cycles -- two for the torus as shown in the figure -- is a topological invariant which equals twice the number of "handles" or "holes" of the (orientable) surface [58] [2].) The known algorithms for determining whether a 1-cycle bounds a 2-D subset of a surface are non-trivial; they require the generation of a cell decomposition of the whole surface such that the 1-cycle's edges also are edges of the decomposition.

FIGURE 3-17

Two Closed Curves Which Do Not Bound
Subsets of a Toroidal Surface

Clearly we have not exhausted boundary representation schemes. (For example, Scheme B may be further generalized by allowing faces to have "holes".) We shall not study validity conditions for additional examples because those above suffice to reach the following conclusions.

- Standard results from elementary algebraic topology lead directly to validity conditions.
- Boundary representation schemes have a complex set of validity conditions.
- Validity conditions of a combinatorial nature are usually easy to test but metric conditions are complex and computationally expensive to test.

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 [55].

*

3.8.3 INFORMAL PROPERTIES

Boundary representations are verbose, e.g. 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 known GMS's that rely on boundary representations attempt to provide such assistance via a user-oriented input language (see the references cited in Section 3.6.5).

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

We noted in Section 2.4.4 that object equality may be detected in non-unique 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 representation schemes use methods closely related to cell decomposition OF OBJECT BOUNDARIES. This implies that it is relatively easy to compute from such representations "homological properties" of objects, i.e. connectedness, number of "holes", and other properties which can be inferred from the homology groups of an object [58] [2].

*

3.9 HYBRID SCHEMES

HYBRID, or NON-HOMOGENEOUS, representation schemes may be designed by combining the approaches discussed in the preceding sections. Three examples follow.

- 1) A CSG/boundary hybrid: representations are CSG-like trees whose leaves are either tuples which represent primitive solids or boundary representations of non-primitive solids.
- 2) A CSG/solid-sweep hybrid: representations are CSG-like trees whose leaves may be solid-sweep representations of non-primitives.
- 3) A 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 GMS's (see Section 3.6.5); systems such as BUILD-2 [18] and GLIDE [24] can store such representations. Scheme 2 is useful for the verification of programs for numerically-controlled machine tools [39]. Scheme 3 is the major representation scheme used by ACRONYM [10], a 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 via conversion algorithms because application algorithms then may be designed to operate on a single (pure) scheme.

*

3.10 CONVERSION BETWEEN REPRESENTATIONS

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

Figure 3-18 summarizes the current state of knowledge about algorithmic conversion between representations of solids. There are two main reasons for the lack of algorithms for bidirectional exact conversion. The first is of a fundamental nature: 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. The second reason is practical rather than fundamental: 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 which compute modulo 2 combinatorial boundaries [2]; this amounts to selecting those cell faces which 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 (called "boundary evaluation" in Rochester jargon) requires non-trivial algorithms. Boundary evaluation algorithms used in the PADL-1 processor [71] are based on the notion of "set membership classification" discussed in [67] and a forthcoming report [72]. Work related to "inverse boundary evaluation", i.e. conversion of boundary representations into CSG, is described in [54].

FIGURE 3-18

Representation of Conversion: Current State of Knowledge

Approximate conversion into spatial enumeration may be accomplished via algorithms which compute the so-called point membership classification function [67] [59], i.e. which determine whether a point of E3 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.)

The relative paucity of known conversion algorithms poses significant constraints on the GMS's that can be built today.

* * *

4. SUMMARY AND CONCLUSIONS

This paper focusses on the representational issues that arise in the design and analysis of GMS's. 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 distinction has several advantages. Specifically, (1) mathematical models can be studied essentially 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 modelling (e.g. to derive conditions for representational validity).

The formal aspects of representation-scheme theory currently are reasonably well understood, although much remains to be learned about sweeping and interpolation schemes. Many of the major gaps in our present knowledge are centered on APPLICATIONS, which cover a broad spectrum. Our brief discussions of applications uncovered a fact that has important implications for GMS design: none of the known representation schemes has characteristics that are uniformly better than all of the others. Therefore a GMS that is intended to support a broad and open-ended range of applications should contain MULTIPLE representations of each object, in different schemes, to achieve flexibility and versatility. The first industrially viable GMS's that maintain multiple object representations are currently being implemented [13] [20] [21]. Careful consideration of validity and consistency issues in the design of such GMS's has yielded designs capable of ensuring automatically the integrity of representations. This is a major advance over earlier modelling systems which relied on user validation.

The design, analysis and implementation of industrially useful geometry systems raise a variety of issues in addition to those addressed in this paper. The following is a non-exhaustive list of such additional issues.

- The representation and use of tolerancing information [57] [34].
- The design of congenial user interfaces.
- The use of (geometric) representations for practical applications in automatic manufacturing, robotics, and so forth.
- The archiving and management of object representations.
- Implementation issues and hardware/software support.

* * *

REFERENCES

- [1] G. J. Agin and T. O. Binford, "Computer descriptions of curved objects", IEEE TRANS. ON COMPUTERS, vol. C-25, no. 4, pp. 439-449, April 1976.
- [2] M. K. Agoston, ALGEBRAIC TOPOLOGY. New York: Marcel Dekker, Inc., 1976.
- [3] A. V. Aho, J. E. Hopcroft and J. D. Ullman, THE DESIGN AND ANALYSIS OF COMPUTER ALGORITHMS. Reading, MA: Addison-Wesley Publishing Co., 1976.
- [4] "Digital representation of physical object shapes", Proposed Standard ANSI Y14.26.1, American Society of Mechanical Engineers, New York, 1978.
- [5] D. Arnon, "A cellular decomposition algorithm for semi-algebraic sets", Tech. Report No. 353, Computer Science Dept., Univ. of Wisconsin, June 1979.
- [6] N. Badler, J. O'Rourke and H. Toltzis, "A spherical representation of a human body for visualizing movement", PROC. IEEE, vol. 67, no. 10, pp. 1397-1403, October 1979.
- [7] R. E. Barnhill and R. F. Riesenfeld, Eds., COMPUTER AIDED GEOMETRIC DESIGN. New York: Academic Press, 1974.
- [8] B. G. Baumgart, "Geometric modelling for computer vision", Report STAN-CS-74-463, Stanford Artificial Intelligence Lab., Stanford Univ., 1974.
- [9] Y. J. Bernascon and J. M. Brun, "Automated aids for the design of mechanical parts", Tech. Paper MS75-508, Society of Manufacturing Engineers, 1975.
- [10] T. O. Binford, R. A. Brooks and R. Greener, "Progress report on a model-based vision system", WORKSHOP ON THE REPRESENTATION OF THREE-DIMENSIONAL OBJECTS, Univ. of Pennsylvania, Philadelphia, PA, May 1-2, 1979.
- [11] O. Bjorke, "Topological characteristics of product definitions", CIRP JOURNAL--MANUFACTURING SYSTEMS, vol. 1, no. 1, pp. 33-46, 1972.
- [12] H. J. Borkin, J. F. McIntosh and J. A. Turner, "The development of three-dimensional spatial modeling techniques for the construction planning of nuclear power plants", COMPUTER GRAPHICS, vol. 12, no. 3, pp. 341-347, August 1978.
- [13] J. W. Boyse, "Preliminary design for a geometric modeller", Research Publication GMR-2768, Computer Science Dept., General Motors Research Laboratories, July 1978.

- [14] J. W. Boyse, "Interference detection among solids and surfaces", **COMMUNICATIONS OF THE ACM**, vol. 22, no. 1, pp. 3-9, January 1979.
- [15] J. W. Boyse, "Data structure for a solid modeller", Research Publication GMR-2933, Computer Science Dept., General Motors Research Laboratories, March 1979.
- [16] I. C. Braid, "Designing with volumes", Ph.D. Dissertation, Univ. of Cambridge, U.K., 1973.
- [17] I. C. Braid, "The synthesis of solids bounded by many faces", **COMMUNICATIONS OF THE ACM**, vol. 18, no. 4, pp. 209-216, April 1975.
- [18] I. C. Braid, "New directions in geometric modelling", **PROC. GEOMETRIC MODELING PROJECT MTG.**, CAM-1, Inc., St. Louis, MO, March 14-16, 1978.
- [19] I. C. Braid, R. C. Hillyard and I. A. Stroud, "Stepwise construction of polyhedra in geometric modelling", C. A. D. Group Document No. 100, University of Cambridge, U.K., October 1978.
- [20] C. M. Brown, A. A. G. Requicha and H. B. Voelcker, "Geometric modelling systems for mechanical design and manufacturing", **PROC. 1978 ANN. CONF. ACM**, Washington, DC, pp. 770-778, December 4-6, 1978.
- [21] C. M. Brown and H. B. Voelcker, "The PADL-2 project", **PROC. 7TH NSF CONF. ON PRODUCTION RESEARCH AND TECHNOLOGY**, Ithaca, NY, pp. F1-F6, September 25-27, 1979.
- [22] J. Cohen and T. Hickey, "Two algorithms for determining volumes of convex polyhedra", **JOURNAL OF THE ACM**, vol. 26, no. 3, pp. 401-414, July 1979.
- [23] C. R. Dyer, A. Rosenfeld and H. Samet, "Region representation: boundary codes from quadtrees", **COMMUNICATIONS OF THE ACM**, vol. 23, no. 3, pp. 171-179, March 1980.
- [24] C. Eastman and M. Henrion, "GLIDE: a language for design information systems", **COMPUTER GRAPHICS**, vol. 11, no. 2, pp. 24-33, July 1977.
- [25] C. M. Eastman and K. Weiler, "Geometric modelling using the Euler operators", Research Report No. 78, Institute of Physical Planning, Carnegie-Mellon Univ., February 1979.
- [26] M. E. Engeli, "A language for 3D graphics applications", in A. Gunther et al., Eds., **INTERNATIONAL COMPUTING SYMPOSIUM**. Amsterdam: North-Holland, 1973, pp. 459-466.

- [27] H. Fuchs, Z. M. Kedem, and S. P. Uelton, "Optimal surface reconstruction from planar contours", COMMUNICATIONS OF THE ACM, vol. 20, no. 10, pp. 693-702, October 1977.
- [28] W. K. Giloi, INTERACTIVE COMPUTER GRAPHICS. Englewood Cliffs, NJ: Prentice-Hall, 1978.
- [29] R. A. Goldstein and R. Nagel, "3-D visual simulation", SIMULATION, vol. 16, no. 1, pp. 25-31, 1971.
- [30] R. Goldstein and L. Malin, "3D modelling with the Synthavision system", PROC. 1ST ANN. CONF. ON COMPUTER GRAPHICS IN CAD/CAM SYSTEMS, Cambridge, MA, pp. 244-247, April 9-11, 1979.
- [31] D. D. Grossman, "Procedural representation of three-dimensional objects", IBM J. RESEARCH AND DEVELOPMENT, vol. 20, pp. 582-589, November 1976.
- [32] M. Henle, A COMBINATORIAL INTRODUCTION TO TOPOLOGY. San Francisco: W. H. Freeman and Co., 1979.
- [33] D. Herbison-Evans, "NUDES 2: A numeric utility displaying ellipsoid solids, version 2", COMPUTER GRAPHICS, vol. 12, no. 3, pp. 354-356, August 1978.
- [34] R. C. Hillyard, "Dimensions and tolerances in shape design", Ph.D. Dissertation, Univ. of Cambridge, U.K., June 1978.
- [35] J. K. Hinds and L. P. Kuan, "Surfaces defined by curve transformations", PROC. 15TH ANN. MTG. NC SOCIETY, Chicago, IL, pp. 325-340, April 9-12, 1978.
- [36] H. Hironaka, "Triangulations of algebraic sets", in ALGEBRAIC GEOMETRY, ARCATA 1974, PROC. OF SYMPOSIA IN PURE MATHEMATICS. Providence, RI: American Mathematical Society, 1975.
- [37] M. Hosaka, F. Kimura and N. Kakishita, "A unified method for processing polyhedra", in INFORMATION PROCESSING '74. Amsterdam: North-Holland, 1974, pp. 768-772.
- [38] M. Hosaka and F. Kimura, "An interactive geometrical design system with handwriting input", in B. Gilchrist, Ed., INFORMATION PROCESSING '77. Amsterdam: North-Holland, 1977, pp. 167-172.
- [39] W. A. Hunt and H. B. Voelcker, "Toward automatic verification of programs for numerically controlled machine tools", Tech. Memo. No. 34, Production Automation Project, Univ. of Rochester, 1980.

- [40] IIT Research Institute, APT PART PROGRAMMING. New York: McGraw-Hill, 1967.
- [41] J. H. Laning, D. A. Lynde, and V. Moreggia, "SHAPES User's Manual", Charles Stark Draper Lab., Cambridge, MA, May 1973.
- [42] J. H. Laning and S. J. Madden, "Capabilities of the SHAPES system for computer aided mechanical design", PROC. 1ST ANN. CONF. ON COMPUTER GRAPHICS IN CAD/CAM SYSTEMS, Cambridge, MA, pp. 223-231, April 9-11, 1979.
- [43] D. T. Lee and F. P. Preparata, "Location of a point in a planar subdivision and its applications", PROC. 8TH ACM SYMPOSIUM ON THE THEORY OF COMPUTING, Hershey, PA, pp. 231-235, May 1976.
- [44] Y. T. Lee and A. A. G. Requicha, "Algorithms for computing the volume and other integral properties of solid objects", Tech. Memo. No. 35, Production Automation Project, Univ. of Rochester, February 1980.
- [45] L. March and P. Steadman, THE GEOMETRY OF ENVIRONMENT. Cambridge, MA: M.I.T. Press, 1974.
- [46] G. Markowsky and M. A. Wesley, "Fleshing out wireframes", Research Report No. RC 8124, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1980.
- [47] W. J. Mitchell and M. Oliverson, "Computer representation of three-dimensional structures for CAEADS", Report CERL-TR-P-86, Construction Engineering Research Lab., U.S. Army Corps of Engineers, 1978.
- [48] R. B. Morris and D. B. Welbourn, "Computer graphics and numerically controlled machine tools for pattern, mould and die production", PROC. 16TH INT. MACHINE TOOL DESIGN AND RESEARCH CONF., Manchester, U.K., pp. 137-141, 1976.
- [49] W. M. Newman and R. F. Sproull, PRINCIPLES OF INTERACTIVE COMPUTER GRAPHICS. New York: McGraw-Hill, 2nd. ed., 1979.
- [50] N. Okino, Y. Kakazu, and H. Kubo, "TIPS-1: technical information processing system for computer-aided design, drawing and manufacturing", in J. Hatvany, Ed., COMPUTER LANGUAGES FOR NUMERICAL CONTROL. Amsterdam: North-Holland, 1973, pp. 141-150.
- [51] N. Okino et al., "TIPS-1", Institute of Precision Engineering, Hokkaido University, Sapporo, Japan, 1978.

- [52] I. Oyake, "Parts representation in CAD/CAM", PROC. OF NATIONAL COMPUTER CONFERENCE, Anaheim, CA, pp. 801-804, 1975.
- [53] R. E. Parent, "A system for sculpting 3-D data", COMPUTER GRAPHICS, vol. 11, no. 2, pp. 138-147, July 1977.
- [54] R. J. Popplestone, C. M. Brown, A. P. Ambler and G. F. Crawford, "Forming models of plane-and-cylinder faceted bodies from light stripes", PROC. 4TH INT. JOINT CONF. ON ARTIFICIAL INTELLIGENCE, Tbilisi, Georgia, USSR, pp. 664-668, September 3-8, 1975.
- [55] K. Preiss, "A procedure for checking the topological consistency of a 2-D or 3-D finite element mesh", PROC. 16TH DESIGN AUTOMATION CONF., San Diego, CA, pp. 200-206, June 25-27, 1979.
- [56] D. R. Reddy and S. Rubin, "Representation of three-dimensional objects", Report CMU-CS-78-113, Computer Science Dept., Carnegie-Mellon Univ., April 1978.
- [57] A. A. G. Requicha, "Part & assembly description languages I: Dimensioning & tolerancing", Tech. Memo. No. 19, Production Automation Project, Univ. of Rochester, May 1977.
- [58] A. A. G. Requicha, "Mathematical models of rigid solid objects", Tech. Memo. No. 28, Production Automation Project, Univ. of Rochester, November 1977.
- [59] A. A. G. Requicha and H. B. Voelcker, "Constructive solid geometry", Tech. Memo. No. 25, Production Automation Project, Univ. of Rochester, November 1977.
- [60] A. A. G. Requicha and R. B. Tilove, "Mathematical foundations of constructive solid geometry: General topology of regular closed sets", Tech. Memo. No. 27, Production Automation Project, Univ. of Rochester, March 1978.
- [61] R. Schuster, "System und Sprache zur Behandlung graphischer Information im rechnergestützten Entwurf", Report KFK-2305, Kernforschungszentrum, Karlsruhe, W. Germany, August 1976.
- [62] H. Seifert, N. Bargele, and B. Fritsche, "Different ways to design three-dimensional representations of engineering parts with PROREN2", PROC. CONF. INTERACTIVE TECHNIQUES IN COMPUTER AIDED DESIGN, Bologna, Italy, pp. 335-343, September 21-23, 1978.
- [63] M. I. Shamos, "Computational geometry", Ph.D. Dissertation, Yale Univ., May 1978.

- [64] R. Shapira, "A technique for the reconstruction of a straight-edge, wire-frame object from two or more central projections", **COMPUTER GRAPHICS AND IMAGE PROCESSING**, vol. 3, no. 4, pp. 318-326, December 1974.
- [65] G. Spur and J. Causemeier, "Processing of workpiece information for producing engineering drawings", **PROC. 16TH INT. MACHINE TOOL DESIGN AND RESEARCH CONFERENCE**, Manchester, U.K., pp. 17-21, 1976.
- [66] G. Spur, "Status and further development of the geometric modelling system COMPAC", **PROC. GEOMETRIC MODELING PROJECT MTG.**, CAM-1, Inc., St. Louis, MO, March 14-16, 1978.
- [67] R. B. Tilove, "Set membership classification: A unified approach to geometric intersection problems", **IEEE TRANS. ON COMPUTERS**, vol. C-29, no. 10, October 1980.
- [68] H. B. Voelcker et al., "Discrete part manufacturing: Theory & practice", Part I of Tech. Report No. 1, Production Automation Project, Univ. of Rochester, December 1974.
- [69] H. B. Voelcker et al., "An introduction to PADL: Characteristics, status, and rationale", Tech. Memo. No. 22, Production Automation Project, Univ. of Rochester, December 1974.
- [70] H. B. Voelcker and A. A. G. Requicha, "Geometric modelling of mechanical parts and processes", **COMPUTER**, vol. 10, no. 12, pp. 48-57, December 1977.
- [71] H. B. Voelcker et al., "The PADL-1.0/2 system for defining and displaying solid objects", **COMPUTER GRAPHICS**, vol. 12, no. 3, pp. 257-263, August 1978.
- [72] H. B. Voelcker et al., "Boundary evaluation procedures for objects defined via constructive solid geometry", Tech. Memo. No. 26, Production Automation Project, Univ. of Rochester, 1980.
- [73] M. A. Wesley, T. Lozano-Perez, L. I. Lieberman, M. A. Lavin, D. D. Grossman, "A geometric modelling system for automated mechanical assembly", **IBM J. RESEARCH AND DEVELOPMENT**, vol. 24, no. 1, pp. 64-74, January 1980.
- [74] T. C. Woo and R. F. Poshadlo, "A feature oriented design modification system", **1ST ANN. CONF. ON COMPUTER GRAPHICS IN CAD/CAM SYSTEMS**, Cambridge, MA, April 9-11, 1979.
- [75] C. I. Yessios, "A notation and system for 3-D constructions", **PROC. 15TH DESIGN AUTOMATION CONF.**, Las Vegas, NE, pp. 125-132, June 19-21, 1978.