A Formal Structure for Nonequivalent Solid Representations

R. Stouffs
Architecture and CAAD,
Swiss Federal Institute of Technology, Zurich, CH-8093, Switzerland
Tel:+41.1. 633 2997 Fax:+41.1.633 1050 stouffs@arch.ethz.ch
R. Krishnamurti
Department of Architecture
Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, U.S.A
Tel:(412) 268 2360 Fax:(412) 268 7819 ramesh@arc.cmu.edu
C. M. Eastman
College of Architecture
Georgia Institute of Technology, Atlanta, Georgia 30332, U.S.A
Tel:(404) 894 9110 Fax:(404) 894 1629 Chuck.Eastman@gatech.edu

Abstract

This work is based on the recognition that there will always be a need for different representations of the same entity, albeit a building or building part, a shape or other complex attribute. Different representations support different sets of operations with varying efficiencies. Given our expectation that such multiple representations will always exist, there is a need, formally, to define the relations between alternative representations, in order to support translation and identify where exact translation is or is not possible, and to define the coverage of different representations. A method for the analysis of representations is developed, which is applied to four different solid modeling representations.

Keywords

Solid modeling, representation, data exchange.

1 INTRODUCTION

This work is based upon the assumption that the long term evolution of building product information will be toward three dimensional shape representation that include various properties, such as material and finish information during design, job and fabrication information during construction, and maintenance information during operation. The advantages for such a representation: for design, construction and operation, appear overwhelming (Eastman, 1993).

With the expectation that there will always be a need for multiple representations, even within the same, or, for related knowledge domains, we recognize that a need exists to support translation between alternative representations within a single, but possibly locally or temporally disconnected, environment. Rather than provide specific applications for the translation between particular alternative representations that may serve a same or similar purpose, we are particularly interested in providing a multirepresentational environment. This will enable multidisciplinary collaborative partners to adopt alternative representations that are particularly suited to each discipline or knowledge domain, while providing for the exchange of data between partners. Such an environment must offer general translational support that recognizes the coverage of and relations between different representations. The supporting system must be able to identify when and where exact translation is or is not possible such that the data flow can be assessed and controlled for data integrity.

From a practical consideration, multiple representations poses a problem when one requires a form of communication between two programs that do not share a common representation. There have been a few attempts to bridge this gap, but each comes with specific restrictions and limitations that do not make them generally applicable. A popular data exchange format in architectural applications is AutoCAD™'s DXF drawing interchange file format (Autodesk, 1992). It is, however, limited to graphical and/or surface models, and is incomplete with respect to solid modeling. The format was designed for AutoCAD's own use which, at the time, did not include general solids. AutoCAD's latest release updates the DXF format to include three dimensional models, but only in binary form, readable only by AutoCAD or other licensees of the ACIS™ modeling kernel. A more universal approach to the standardization of graphical data exchange is IGES (Smith, 1988). Successive versions of it have seen the inclusion of different solid models: constructive solid geometry in version 4, and a boundary representation is slated for inclusion in version 5. The general strategy of all current translation approaches is to develop a neutral representation that can represent the capabilities available in current modelers.

It is not our intention here to develop a new standard that would be "up to date" as to current research and usage, but to consider an alternative approach that may take away the need for standards altogether. Given the number of different, often equivalent solid representations, any choice of a single standard is rather subjective and creates a value assignment indifferent to the purposes of the different representations. Moreover, as the field of solid modeling expands and new types of information are included, such as features, surface finishes, etc., such a standard could become quickly outdated. Indeed, solid modelers with new capabilities are being developed, such as the parametric solid modelers found in Pro-Engineer™, SDRC's IDEAS™ and Autodesk's DESIGNER™ for which no adequate translators exist. Instead, we consider a different approach that is not restricted to currently known representations. We distinguish the data classes and subclasses that constitute a

representation and propose the development of a two way communication system using these data classes as the vocabulary elements. This vocabulary is not a priori limited and may be extended by new applications.

The purpose of our paper is threefold:

- To identify the issues of translation among solids.
- To identify those cases where exact translations are possible and those where not.
- To consider the coverage of a solid model in terms of the range of possible shape subclasses it can depict.

2 SOLID MODELS

Solid models fall into three main kinds: decomposition models (e.g., cell decomposition), constructive models (e.g., constructive solid geometry) and boundary models. Hybrid models are also possible. Of these, boundary models are among the most popular, mainly, because many of the questions we seek answers to relate to the surface of the solid. Boundary models represent a solid as a collection of two dimensional boundary surfaces or faces. Even here, a large number of different representations exist, based on a variety of approaches, and more appear as research continues.

Every application that employs a solid model can be expected to interpret information into one or more representations. When applications communicate, received information is interpreted into an application's own representation; a similar interpretation can be expected for the other application. The core of the communication system is limited to the facilitation of such interpretation. Eastman (1994) gave a treatment of two dimensional polygonal graphics information. In this paper we consider a similar decomposition of three dimensional representational information into data types and with respect to scope. To demonstrate such a decomposition, we offer a comparison of representations found in the literature.

We restrict our comparison to boundary representations of polyhedral solids (possibly part of a hybrid model), in particular, and rectilinear segments (volume, plane, line or points) in a three dimensional space, in general. Boundary representations (*breps*) are particularly suited to graphics applications in that, in the main, these directly represent the boundary faces that visually reflect upon the solids.

3 REPRESENTATIONS

Representations of solids are expressed in terms of data classes with subclasses, and defined in a consistent manner across all representations considered. Classes are presented as compositions of their part representations, e.g., faces, edges, vertices, and so on. Given a set of representations, equivalencies across both specialization and composition lattices can be established. Communication between representations can be identified as either a one-to-one (isomorphism) or a one-to-many (homomorphism) equivalency. The following concepts are basic.

3.1 Definitions

Representations are class structures identified by attributes grouped by one or more constructors.

Attributes are named entities identified by a type, **type**(*a*), that specifies its set of possible values. Common types are real, integer, string and boolean.

Constructors are devices for relating attributes to one another; examples include records, arrays, lists, rings, stacks and so forth. Definitions of those constructors we will be using are given in Table 1. Others may be defined, as needed. Constructors specify relations between attributes, where such relations are used and assigned an interpretation within the operations on the representation.

Structures are groupings of constructors; *entities* are attributes or structures. Representations are structures along with the operations supported by the structure.

Table 1 Definitions of example constructors.

Record	An enumerated set of entities, distinguished by membership.
Array	A fixed length sequence of entities, distinguished by position.
List	A variable length sequence of entities, distinguished by position.
Ring	A variable length sequence of entities, distinguished by (relative) position, where the last and first entities are considered sequential.

Representations found in computer aided design or geometrical modeling are, typically, complex structures of attributes and constructors. Moreover, a representation may be a construction of another. Such complex representations may be considered as a whole. Representations may also be considered in terms of nestings of other representations.

3.2 Notation

We introduce a simple notational device to abstract a representation. As an example,

$$\mathbf{R} = C_{\text{record}}(C_{\text{list}}(a_1), a_2, a_3, C_{\text{list}}(C_{\text{record}}(a_4, a_5, a_6)))$$

In this example, representation **R** consists of a record structure with two nested list structures, one over a simple attribute, $C_{\text{list}}(a_1)$, the other over a record structure, $C_{\text{list}}(C_{\text{record}}(a_4, a_5, a_6))$. In addition, the structure **R** has two simple attributes: a_2 and a_3. Equivalently, we can also consider **R** to have two nested representations and describe it as follows:

$$\mathbf{R} = C_{\text{record}}(\mathbf{R_a}, a_2, a_3, \mathbf{R_b})$$
$$\mathbf{R_a} = C_{\text{list}}(a_1)$$
$$\mathbf{R_b} = C_{\text{list}}(\mathbf{R_c})$$

$\mathbf{R_c} = C_{\text{record}}(a_4, a_5, a_6)$

Since we are dealing with only a few constructors, mainly sequences – distinguished by position; either of fixed or variable length; and sequentially cyclical or otherwise – we can adopt a simplified notation. We use angular brackets to denote a record (or enumerated set), and parentheses to denote a sequence. Parentheses without any closing superscript denote a list or variable length sequence; parentheses with a numeric closing superscript, *n*, denote an array or fixed length sequence of size *n*; and, parentheses with a closing superscript '*' denote a ring or cyclical sequence. The same notation also allows us to denote a fixed length cyclical sequence, e.g., $(a)^{*n}$. Then, the previous example becomes:

$\mathbf{R} = \langle(a_1), a_2, a_3, (\langle a_4, a_5, a_6\rangle)\rangle$

3.3 Derivations

A representation explicitly defines and stores one set of attributes and relations between them. But other attributes and relations may be derived from the stored ones, in a deterministic fashion. We call these values *derivations* of the stored data. The representation uniquely determines the derived values. Derivations may be simple attributes. A representation of a polygon as a list of vertices (which are arrays of coordinates) has as possible derivations the area of the polygon, its perimeter (attributes), and centroid (an array of coordinates). Some derivations are parts of other representations, or together with parts of the original representation, can define another whole representation. We denote such derivations as extensions to the original representation, and specify the derived structure as derived from the original structure using the '→' symbol:

$\mathbf{R_d} = \langle(a_1), a_2, a_3, (\langle a_4, a_5, a_6\rangle)\rangle \rightarrow \langle a_7, a_8, (a_9)\rangle$

An example might be that of an arc represented as a center point and two equidistant endpoints, swept counterclockwise, having as derivations the radius, beginning and ending angles from the origin. Another representation may be defined using these derived values:

$\mathbf{R_{arc1}} = \langle\langle x, y\rangle_{\text{center}}, \langle x, y\rangle_{\text{start}}, \langle x, y\rangle_{\text{end}}\rangle \rightarrow \langle a_{\text{b\_angle}}, a_{\text{e\_angle}}, a_{\text{radius}}\rangle$

$\mathbf{R_{arc2}} = \langle\langle x, y\rangle_{\text{center}}, a_{\text{b\_angle}}, a_{\text{e\_angle}}, a_{\text{radius}}\rangle \rightarrow \langle\langle x, y\rangle_{\text{start}}, \langle x, y\rangle_{\text{end}}\rangle$

3.4 Equivalent entities

In the preceding examples, given the center point, the radius, beginning and ending angles are derived from the endpoints; likewise, given the center point, the endpoints are derived from the radius, beginning and ending angles. In terms of their information content with respect to the arc representation, the structures $\langle a_{\text{b\_angle}}, a_{\text{e\_angle}}, a_{\text{radius}}\rangle$ and $\langle\langle x, y\rangle_{\text{start}}, \langle x, y\rangle_{\text{end}}\rangle$ are identical. We say that both entities are *equivalent* within the representation. We use the symbol '↔' to signify equivalent entities within a representation:

$\mathbf{R_{arc3}} = \langle\langle x, y\rangle_{\text{center}}, \langle\langle x, y\rangle_{\text{start}}, \langle x, y\rangle_{\text{end}}\rangle \leftrightarrow \langle a_{\text{b\_angle}}, a_{\text{e\_angle}}, a_{\text{radius}}\rangle\rangle$

3.5 Optional entities

Within the previous representation, one of the two structures: $<a_{b\_angle}, a_{e\_angle}, a_{radius}>$ or $<<x, y>_{start}, <x, y>_{end}>$ is optional, but not both. This is because these entities are equivalent within the representation. Sometimes, representations may have entities that are explicitly designated to be optional, that is the attributes or entities may or may not be specified a value. Examples of optional attributes are the color and crosshatching of figures. We use enclosing square brackets to signify entities that are explicitly specified as optional within the representation.

$$\mathbf{R_{arc4}} = <<x, y>_{center}, <x, y>_{start}, <x, y>_{end}, [a_{color}]>$$

Entities that are not specified as optional are considered mandatory, unless they are equivalent within the representation. Note that it is important that a representation is well formed, that is, that all equivalent or optional entities are explicitly specified as such. Otherwise, these will be considered as mandatory, and the subsumption relationship specified below may not properly apply.

3.6 Alternatives

Sometimes a representation can take different forms, depending on the particular instance, and these forms may not be equivalent. For example, a representation may model arcs as well as straight line segments. Even though each type of figure has its own representation in terms of its attributes and grouping constructors, we may want to denote each figure as an instance of the same overall representation. We use the symbol '|' to divide such alternatives within a representation:

$$\mathbf{R_{figure}} = <\mathbf{R_{arc}} \mid \mathbf{R_{line}} \mid \mathbf{R_{polyline}}>$$

A figure may represent an arc, a straight line or a concatenation of straight line segments. Each figure belongs to exactly one of these three representations, but a list of figures might include all three representations.

4 SUBSUMPTION

Equivalency between entities can be extended to representations. Informally, we may consider two representations with identical information content as *equivalent*. It is obvious that the representations $\mathbf{R_{arc1}}$, $\mathbf{R_{arc2}}$ and $\mathbf{R_{arc3}}$ are equivalent, since these represent the same class of figures. [Note that two representations that are equivalent do not require their entities to be positioned identically in a record.] On the other hand, $\mathbf{R_{arc4}}$ is not equivalent with any of these arc representations, because it can represent ordinary arcs and optionally a family of colored arcs as well. Therefore, we say that $\mathbf{R_{arc4}}$ *subsumes* all three representations $\mathbf{R_{arc1}}$, $\mathbf{R_{arc2}}$ and $\mathbf{R_{arc3}}$. This subsumption only holds because the color attribute is specified as optional. If we define an arc representation with a mandatory color attribute:

$$\mathbf{R_{arc5}} = \langle\langle x, y\rangle_{\text{center}}, \langle x, y\rangle_{\text{start}}, \langle x, y\rangle_{\text{end}}, a_{\text{color}}\rangle$$

then, this new representation no longer subsumes the other arc representations. However, $\mathbf{R_{arc4}}$ still subsumes $\mathbf{R_{arc5}}$.

4.1 Subsumption relation

We can consider the subsumption relation, formally, in set theoretic terms. The *domain* of an entity, **dom**(*E*), is the set of possible distinct individuals it can depict. The domain of an attribute is the set of values it can take. The domain of each component of a representation can be defined. A grouping of attributes has as its domain the Cartesian product of its attribute domains. A record structure has as its domain the Cartesian product of its entity domains. A sequence structure has as its domain all combinations of its entities' domains, as allowed by the constructor; if the structure is of unbounded length, then its domain is indefinite. Using this same approach, the domain of more complex representations can be defined, involving any combination of entities grouped by different constructors.

We can add to the domain of any representation certain derivations. The possible set of derived attributes is open ended; there is no practical way to identify all possible derivations. However, when comparing two representations, derivations of one representation that are equivalent to entities of the other representation identify equivalent partial domains. When determining the domain of a representation, these partial domains as defined by derivations that are equivalent to entities in a subsumed representation, participate in the overall domain. In the arc example above, the domains of the arcs are extended because the derivations are equivalent to entities of the other (subsumed) representation.

We can define the subsumption relation in terms of the domains of the representations. A representation subsumes another representation if the domain of the first representation can be partitioned such that one of the parts corresponds to the domain of the second representation. We use the symbol '≤' to denote the subsumption relationship. That is, given two representations $\mathbf{R_x}$ and $\mathbf{R_y}$, if $\mathbf{R_x}$ is subsumed by $\mathbf{R_y}$ (or $\mathbf{R_y}$ subsumes $\mathbf{R_x}$), we write $\mathbf{R_x} \leq \mathbf{R_y}$. If, additionally, $\mathbf{R_x}$ subsumes $\mathbf{R_y}$, then the two representations are equivalent, $\mathbf{R_x} \equiv_{\text{eq}} \mathbf{R_y}$. Formally,

$$\mathbf{R_x} \equiv_{\text{eq}} \mathbf{R_y} \text{ if and only if } \mathbf{R_x} \leq \mathbf{R_y} \text{ and } \mathbf{R_y} \leq \mathbf{R_x}$$

The subsumption relationship is a partial order relation (reflexive, antisymmetric and transitive) that specifies a partial ordering on a set of representations. Consider $\Re$ the set of all possible representations. The least upper bound for any set of representations defines the operation of *sum* ('+') on representations. Similarly, the greatest lower bound for any set of representations defines the operation of *product* ('·') on representations. The set $\Re$ is closed under sum and product. [Further consideration of the operations of sum or product on representations is the focus of Stouffs and Krishnamurti's forthcoming paper.] The set $\Re$ has a zero or minimal element, **nil**, that is the greatest lower bound for the empty set of representations. The **nil** representation is subsumed by every representation. The subsumption relation defines a lattice on $\Re$.

5 ORDERING RULES

The above conditions can be specified by a set of ordering rules. The first kind of ordering rules applies to groupings of entities or representations:

$$\mathbf{nil} \leq \mathbf{R} \text{ for every representation } \mathbf{R} \tag{1.1}$$

$$\langle a \rangle \leq \langle b \rangle \text{ if and only if } \mathbf{type}(a) = \mathbf{type}(b) \text{ and } \mathbf{dom}(a) \subseteq \mathbf{dom}(b) \tag{1.2}$$

We consider a merge operation '$\wedge$' of two structures. If both are records, this results in a record structure composed of all entities in both structures. Otherwise, if either is not a record structure, the merge treats this structure as a single entity in the other structure, or, in a new record. The merge operation, unlike sum, is not defined over $\Re$: that is, $\mathbf{nil} \wedge \mathbf{R}$ is undefined. The merge operation is introduced as a convenience to simplify the ordering rules.

$$\langle a \rangle \wedge \mathbf{R} \leq \langle b \rangle \wedge \mathbf{S} \text{ if } \langle a \rangle \leq \langle b \rangle \text{ and } \mathbf{R} \leq \mathbf{S} \tag{1.3}$$

Note that this is only a sufficient condition for a record structure to subsume another, not a necessary condition.

$$\mathbf{R} \leq \mathbf{R} \wedge [\mathbf{S}] \tag{1.4}$$

$$\mathbf{R} \wedge \mathbf{S} \leq \mathbf{R} \wedge [\mathbf{S}] \tag{1.5}$$

That is, a structure **R** is subsumed by another structure that is composed of **R** and other optional entities. A structure that includes entities **R and S** is subsumed by another structure that is composed of entities **R and** [**S**].

Another method of combining structures is as alternatives within a same representation. We use the symbol '$\vee$' to denote this operation: $\mathbf{R} \vee \mathbf{S} = \langle \mathbf{R} \mid \mathbf{S} \rangle$. Like the merge operation, this operation is not defined over $\Re$: that is, $\mathbf{nil} \vee \mathbf{R}$ is undefined.

$$\mathbf{R} \leq \mathbf{R} \vee \mathbf{S} \tag{1.6}$$

Each alternative in a representation is subsumed by the representation.

The second kind of ordering rules applies to the constructors that define relations. We denote a constructor relation *C* over entity **R** as $C(\mathbf{R})$. In general:

$$C(\mathbf{R}) \leq C(\mathbf{S}) \text{ if and only if } \mathbf{R} \leq \mathbf{S} \tag{2.1}$$

That is, one structure subsumes a second if both constructors are of the same type and dimensionality, and the entities of the first structure subsume the entities of the second.

The dimensionality of a constructor is the length of the resulting grouping. This dimensionality is definite for some constructors and indefinite for others. For example, a quadrilateral may be defined by exactly four points, a triangle by exactly three points, while a polygon may have a variable number of points.

$$(\mathbf{R})^m \leq (\mathbf{R})^n \text{ if and only if } m \leq n \tag{2.2}$$

Thus, one array structure subsumes a second if the dimensionality of the first array constructor is greater than the dimensionality of the second, and they both apply to the same entity.

In addition, some constructors may subsume others. For example, a list structure subsumes an array structure if they both apply to the same entity:

$$C_{\text{array}}(\mathbf{R}) \leq C_{\text{list}}(\mathbf{R}) \tag{2.3}$$

That is, every relation derived from an array can also be derived from a list, because the array length is fixed. The reverse is not true. [See, for example, the long line of research using lists to deal with sparse arrays (Pooch, 1973). We are, of course, not taking into account practical considerations of efficiency.]

Other rules, apart from those given, may be defined. Subsumption relations form lattices. Lattices are transitive, reflexive and antisymmetric (Stone, 1973), and have a long history in the theory of types (see, for example, Cardelli, 1985).

A simple example may be helpful in showing how these rules can be used to order representations. Consider three types of figures: a two dimensional polygon defined by *n* points, a triangle and a quadrilateral. Furthermore, consider these polygons with, say, optional and mandatory crosshatching. The subsumption lattice would be as shown in Figure 1. The vertices of the quadrilateral and triangle are assumed to be represented by arrays of fixed length, whilst that of the polygon by a variable length list. The representation at the top is the most general, in that it can represent every possible individual of the representations below it. The representations at the bottom are the most limited, not being able to depict conditions in the representations above it. If a general polygon involving three dimensional coordinates was added, it would subsume polygon (p_1, … p_n), based on rules 2.1

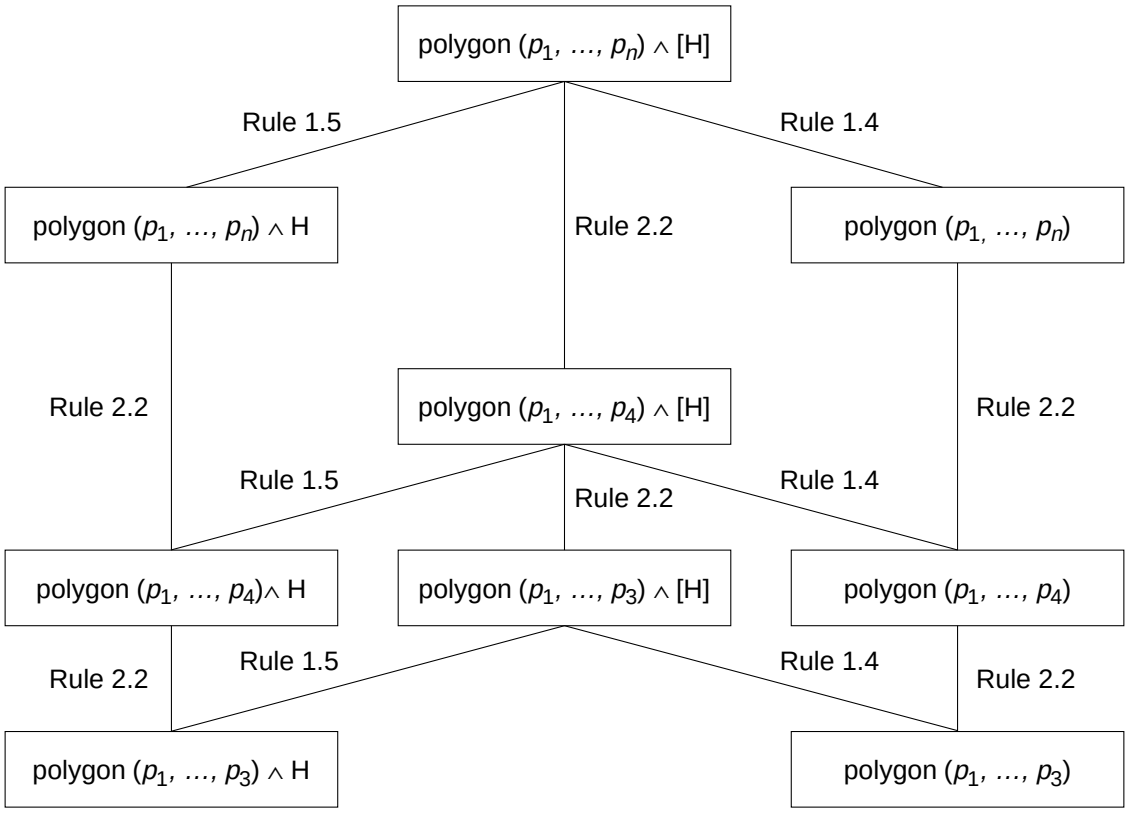

Figure 1 An exemplar subsumption lattice for polygons. The applications of the ordering relations are shown.

and 2.2. This assumes that the coordinate values are carried in an array. If coordinates are carried as named attributes in a vertex record, the same relation would only hold if explicitly specified as such, for instance, 2D coordinates can be mapped to 3D coordinates with the third coordinate equal to 0, e.g., $\langle x, y\rangle \equiv_{eq} \langle x, y, z = 0\rangle$.

6 REGULARIZED REPRESENTATIONS OF SOLID MODELS

Solid modeling representations are complex structures. Some subsume others and some are equivalent to others. Generally, under a *brep*, a solid is represented by its boundary, i.e., a collection of (polygonal) faces. A face is represented by its boundary line segments or edges; an edge is defined by its endpoints or vertices. Each particular brep distinguishes itself depending on the characteristics of the representation. For example, representational elements, i.e., faces, edges and vertices, may be explicitly or implicitly defined and may be uniquely defined. It also depends on the degree of explicitness of the topological structure, the degree of information redundancy, and the elements to which the geometry is attached. Each particular representation influences the domain of representable figures, that is, whether solids or faces may have holes, the degree of manifoldness of the solids, and whether and how the representation allows for nonregular figures.

Each of the representations considered below is an example of one of the three types of boundary models identified by Mäntylä (1988): edge based, vertex based, and face (or polygon) based. The simplest of these is the *winged edge* representation (Baumgart, 1975) which is an example of an edge based boundary model that is explicit in representing all face, edge and vertex elements uniquely and is complete in the representation of the topological structure. In its original formulation it is limited to manifold and regular solids. [Informally, a solid is regular if it does not contain any “dangling” faces or edges, or isolated points. Such a regular solid is manifold if it has no parts that touch at a line or point of contact, nor does it touch itself at a line or point of contact.] Variations of the representation extend to nonmanifold solids such as the *half edge* representation (Mäntylä, 1988). The *winged triangle* representation (Paoluzzi, 1989) is an example of a vertex based boundary model for nonmanifold, regular solids. This representation can be made canonical. The *maximal element* representation (Krishnamurti, 1992; Stouffs, 1994) is a canonical representation that supports a polygon or face based model. The topology is only implicitly represented in a recursive declaration of solids in terms of faces, in turn, in terms of edges, in turn, in terms of vertices. The representation allows for nonregular shapes, yet, the shape operations are always regular within their dimensionality.

6.1 Baumgart’s winged edge representation

The winged edge polyhedron representation is an edge based boundary model that explicitly represents each face, edge and vertex element uniquely and is complete in the representation of the topological structure. The basic representational element is the edge. Each edge has a direction and two bounding vertices assigned. Each edge forms a part of the boundary of exactly two faces. From the direction of the edge and the notion of outside versus inside of a solid with respect to a face (represented by the surface normal of a face), these two faces, related to the specific edge, can be distinguished as clockwise and counterclockwise. The

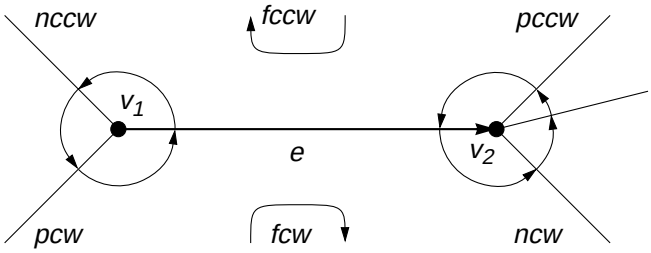

Figure 2 Winged edge data structure.

Table 2 Winged edge representation.

Representation	Edge based boundary representation.
Elements	*Body*, *face*, *edge* and *vertex*. These are all explicit and unique. The topology is attached to *edges*, and the geometry to *vertices*.
Topology	*Body*: list of *faces*, *edges* and *vertices*. *Face*: one of the *edges* on its perimeter. *Edge*: an orientation, two bounding *vertices*, two adjacent *faces*, and four immediate neighboring *edges* clockwise and counter-clockwise about its face perimeters as seen from the exterior side of the surface. These links are consistently oriented with respect to the surface of the polyhedron such that the surface always has two sides: inside and outside. *Vertex*: one of the *edges* on its perimeter.
Geometry	*Face*: an exterior pointing normal vector. *Vertex*: 3D coordinates.
Scope	*Regular*: only regular 3D *bodies*. *Manifold*: each *edge* is adjacent to exactly two *faces*. *Connected surfaces*: bodies with only a single shell. *No faces with holes*: each *face* has a single loop of *edges*.
Extensions	Braid (1980) allows for *faces* with holes. Yamaguchi (1985) specify a bridge edge representation that allows for *faces* with holes.

edges that compose the boundary of a face are linked in cyclical order such that for each edge, the next clockwise, next counterclockwise, previous clockwise and previous counterclockwise edge can be determined. A face is defined by a single edge and its orientation with respect to this face. The geometry is represented in the vertices. The condition of each edge belonging to exactly two faces limits this representation to only manifold solids. In its minimal form the winged edge representation does not allow for faces with holes. Extensions to this representation that allow for faces with holes are present in the literature (Braid, 1980; Yamaguchi, 1985).

The winged edge representation can be described as follows:

$$\mathbf{R}_{\mathbf{winged\_edge}} = \langle(\mathbf{R}_{\mathbf{we\_body}})\rangle$$
$$\mathbf{R}_{\mathbf{we\_body}} = \langle(\mathbf{R}_{\mathbf{we\_face}}), (\mathbf{R}_{\mathbf{we\_edge}}), (\mathbf{R}_{\mathbf{we\_vert}})\rangle$$
$$\mathbf{R}_{\mathbf{we\_face}} = \langle V_{\text{normal}}, \mathbf{R}_{\mathbf{we\_edge}}\rangle$$
$$\mathbf{R}_{\mathbf{we\_edge}} = \langle a_{\text{orientation}}, (\mathbf{R}_{\mathbf{we\_vert}})^2, (\mathbf{R}_{\mathbf{we\_face}})^2, (\mathbf{R}_{\mathbf{we\_edge}})^4\rangle$$
$$\mathbf{R}_{\mathbf{we\_vert}} = \langle\mathbf{R}_{\mathbf{vert}}, \mathbf{R}_{\mathbf{we\_edge}}\rangle$$
$$\mathbf{R}_{\mathbf{vert}} = \langle\langle x, y, z\rangle \leftrightarrow \langle x, y, z, w\rangle\rangle$$

The winged edge representation edge carries pointers to adjacent faces, edges and vertices. Vertices have back pointers to an incident edge. It also allows triangulation of edge rings, as required for certain display operations.

6.2 Mäntylä's half edge representation

Like the winged edge representation, the half edge representation is an edge based boundary model. Each edge is represented by two distinct edge-halves. Thus, the representational element is an edge-half. Each half is contained in only one face. Each face has its own unique set of edge-halves. The clockwise ordering of the edge-halves around a face determines the orientation of the face. Each edge-half has one and only one orientation. Each edge-half is ordered in the opposite orientation of its other half. Both edge-halves are bound by the same pair of vertices.

The half edge representation can be described as follows:

$$\mathbf{R}_{\mathbf{half\_edge}} = \mathbf{R}_{\mathbf{he\_solid}}$$
$$\mathbf{R}_{\mathbf{he\_solid}} = \langle\mathbf{R}_{\mathbf{he\_face}}, \mathbf{R}_{\mathbf{he\_edge}}, \mathbf{R}_{\mathbf{he\_vert}}, (\mathbf{R}_{\mathbf{he\_solid}})^2\rangle$$
$$\mathbf{R}_{\mathbf{he\_face}} = \langle E_{\text{plane}}, (\mathbf{R}_{\mathbf{he\_loop}})^2, (\mathbf{R}_{\mathbf{he\_face}})^2, \mathbf{R}_{\mathbf{he\_solid}}\rangle$$
$$\mathbf{R}_{\mathbf{he\_loop}} = \langle\mathbf{R}_{\mathbf{he\_edge\_half}}, (\mathbf{R}_{\mathbf{he\_loop}})^2, \mathbf{R}_{\mathbf{he\_face}}\rangle$$
$$\mathbf{R}_{\mathbf{he\_edge}} = \langle(\mathbf{R}_{\mathbf{he\_edge\_half}})^2, (\mathbf{R}_{\mathbf{he\_edge}})^2\rangle$$
$$\mathbf{R}_{\mathbf{he\_edge\_half}} = \langle\mathbf{R}_{\mathbf{he\_vert}}, (\mathbf{R}_{\mathbf{he\_edge\_half}})^2, \mathbf{R}_{\mathbf{he\_loop}}, \mathbf{R}_{\mathbf{he\_edge}}\rangle$$
$$\mathbf{R}_{\mathbf{he\_vert}} = \langle\mathbf{R}_{\mathbf{vert}}, (\mathbf{R}_{\mathbf{he\_vert}})^2, \mathbf{R}_{\mathbf{he\_edge\_half}}\rangle$$

The structures in the half edge representation may not be apparent. Solids, faces, loops, edges, edge-halves and vertices are each contained in a doubly linked list. A solid references a face, edge and vertex from the appropriate lists. A face has a single outer loop and a

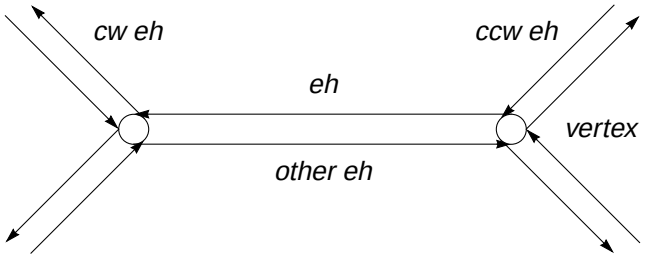

Figure 3 Half edge data structure.

Table 3 Half edge representation.

Representation	Edge based boundary representation.
Elements	*Solid, face, loop, edge, edge-half* and *vertex*. These are all explicit and unique. The topology is attached to *edges*, and the geometry to *vertices* (and *faces*).
Topology	*Solid*: the first element in the lists of *faces, edges* and *vertices*, and the previous and next *solid* in the list of *solids.* *Face*: one outer boundary *loop*, a list of inner boundary *loops*, the previous and next *face* in the list of *faces*, and the parent *solid* *Loop*: one of the *edge-halves* that form the boundary, the previous and next *loop* in the list of *loops*, and the parent *face.* *Edge*: two *edge-halves*, and the previous and next *edge* in the list of *edges.* *Edge-half*: the starting *vertex* of the line segment in the direction of the *loop*, the previous and next *edge-half* in the list, and the parent *loop.* *Vertex*: the previous and next *vertex* in the list of *vertices.*
Geometry	*Face*: equation of a plane *Vertex*:3D coordinates
Scope	*Regular*: only regular 3D *solids.* *Connected surfaces*: bodies with only a single shell. *Pseudo manifold*: manifold representation of nonmanifold *solids.* Each *edge* has exactly two *edge-halves* and is part of two *face* boundaries. However, an *edge* may coincide with another *edge* or a *face*, and a *vertex* may coincide with another *vertex*, an *edge* or a *face.*

reference into a list of inner loops. Each face, loop, edge-half and vertex points back to its parent solid, face, loop and edge-half, respectively.

The half edge representation includes faces bounded by multiple loops. It does not include nested shells. It can have derivations that triangulate each of its faces, as implemented in many graphical display algorithms (Foley, 1992). Thus, it can derive Paoluzzi's winged triangle representation, but does not subsume it (see below).

The half edge representation subsumes Baumgart's winged edge representation as described below. Winged edges are the aggregation into one record of two half edges. It can also derive the vertex back pointers of the winged edge.

$$\begin{aligned}
\mathbf{R_{we_edge}} &= \langle a_{\text{orientation}}, (\mathbf{R_{we_vert}})^2, (\mathbf{R_{we_face}})^2, (\mathbf{R_{we_edge}})^4 \\
&\leq \langle (\mathbf{R_{he_vert}})^2, (\mathbf{R_{he_face}})^2, (\mathbf{R_{he_edge}})^4 \rangle \\
&\equiv_{\text{eq}} \langle (\mathbf{R_{he_vert}})^2, (\mathbf{R_{he_loop}})^2, (\mathbf{R_{he_edge}})^4 \rangle \\
&\equiv_{\text{eq}} \langle (\mathbf{R_{he_edge_half}})^2, (\mathbf{R_{he_edge}})^2 \rangle \\
&= \mathbf{R_{he_edge}}
\end{aligned}$$

Moreover, the half edge representation can define faces made up of multiple loops of edges, which the winged edge structure cannot represent. Thus, the subsumption ordering relation is one way and the two representations are not equivalent.

6.3 Paoluzzi's winged triangle representation

The winged triangle representation is an example of a vertex based boundary model that explicitly and uniquely represents triangular faces and vertex elements. Edges are represented implicitly, as defined by adjacent vertices. The topological structure is only partially represented. The basic representational element is the triangular face. Each face has two 3-tuples representing the adjacent faces and bounding vertices. A shell is composed of a (ordered) set of faces. A polyhedron is canonically composed of a maximal number of shells, such that nonmanifold polyhedra are represented as manifolds with local nonmanifoldness. The face polygons are triangulated after each (Boolean) operation. As such, the number of triangles is minimized, and a canonical triangulation based on the ordered boundary vertices is possible. The geometry is represented in the vertices. Faces contain exterior/interior information. This allows for the representation of unbounded solids (infinite volume) with bounded (finite) surface. The triangulation only allows for linear polyhedra or linearized approximations of polyhedra with curved surfaces.

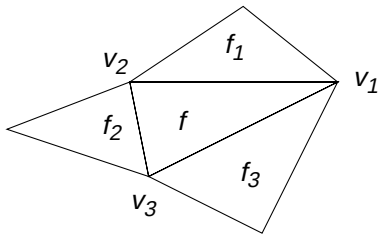

Figure 4 Winged triangle data structure.

Table 4 Winged triangle representation.

Representation	Vertex based boundary representation.
Elements	*Polyhedron*, *shell*, *face*, *edge* and *vertex*. All, except *edges*, are explicit and unique. *Edges* are implicitly defined. The topology is attached to *faces*, and the geometry to *vertices.*
Topology	*Polyhedron*: an (ordered) list of *shells.* *Shell*: list of *faces* and *vertices.* *Face*: three (ordered) bounding *vertices.* three (ordered) adjacent *faces.*
Geometry	*Face*: an exterior pointing normal vector. *Vertex*: 3D coordinates.
Scope	*Regular*: only regular 3D *polyhedra.* *Pseudo manifold*: (multishell) manifold representation of non-manifold *polyhedra.* *Unbounded*: infinite volumes with finite surface. *Linear polyhedra* or linearized representations of curved surfaces.
Remarks	(Possibly) *canonical*: *Polyhedron* has maximal number of *shells.* Canonical triangulation of *face* polygons based on ordered *vertices.* Ordered lists of *shells*, *faces* and *vertices.*

The winged triangle representation can be represented abstractly as:

$$\mathbf{R}_{\mathbf{winged\_triangle}} = \langle(\mathbf{R}_{\mathbf{wt\_polyhedron}})\rangle$$
$$\mathbf{R}_{\mathbf{wt\_polyhedron}} = \langle(\mathbf{R}_{\mathbf{wt\_shell}})\rangle$$
$$\mathbf{R}_{\mathbf{wt\_shell}} = \langle(\mathbf{R}_{\mathbf{wt\_face}}), (\mathbf{R}_{\mathbf{vert}})\rangle$$
$$\mathbf{R}_{\mathbf{wt\_face}} = \langle V_{\text{normal}}, (\mathbf{R}_{\mathbf{vert}})^3, (\mathbf{R}_{\mathbf{wt\_face}})^3\rangle$$

The winged triangle representation consists of multiple shells, each made up of a list of faces. Each face is a sequence of three vertices. The $\mathbf{R}_{\mathbf{wt\_face}}$ may have derivations that include defining the polygonal boundaries of coplanar adjacent triangular faces. These face polygons may be matched to identify opposite matching edge pairs, which may be defined with an additional record and point to the matching pair. The resulting boundaries may define faces that have multiple loops bounding them. Loops adjacent to the same face may be matched and put in the same face record. These allow the derivation of the half edge data structure:

$$\mathbf{R}_{\mathbf{wt\_shell}} = \langle(\mathbf{R}_{\mathbf{wt\_face}}), (\mathbf{R}_{\mathbf{vert}})\rangle$$
$$\rightarrow \langle((\mathbf{R}_{\mathbf{vert}})^3)\rangle \rightarrow \langle(\mathbf{R}_{\mathbf{he\_face}}), (\mathbf{R}_{\mathbf{he\_edge}}), (\mathbf{R}_{\mathbf{vert}})\rangle \rightarrow \mathbf{R}_{\mathbf{he\_solid}}$$

Therefore, Paoluzzi's winged triangle representation subsumes Mäntylä's half edge representation. (The derivations are presented in three steps for illustrative simplicity only.)

6.4 Krishnamurti and Stouffs' maximal element representation

The maximal element representation is an example of a polygon based boundary model. It is a canonical representation in which the topology is mostly implicitly represented. The basic representational element is the face in the representation of a solid or volume segment, an edge in the representation of a face or plane segment and a vertex in the representation of an edge or line segment. That is, solids, faces, edges and vertices are recursively defined by their boundary of a lower dimensionality and all elements can be part of a shape without necessarily being part of the boundary of a shape. That is, nonregular shapes can be represented; yet, the operations on shapes are always regular within their dimensionality. These operations of sum, difference, intersection and symmetric difference define a generalized Boolean algebra (or Boolean ring) on shapes. A general solid consists of one or more disjoint (maximal) volume segments that share no faces but may share edges and/or vertices. Thus, solids may be nonmanifold. The boundary of a volume segment is composed of one outer shell and zero, one or more inner shells, constituting holes in the solid. Each shell is composed of a set of (maximal) plane segments, canonically ordered according to the face equation. Similarly, each plane segment has a single outer boundary and zero, one or more inner boundaries of (maximal line segments). A line segment is defined by its two endpoints. The geometry is represented in the vertices and partially duplicated in the edge and face equations.

The maximal element representation can be described as follows:

$$\mathbf{R}_{\mathbf{maximal\_element}} = \langle(\mathbf{R}_{\mathbf{me\_segment}})\rangle$$
$$\mathbf{R}_{\mathbf{me\_segment}} = \langle\mathbf{R}_{\mathbf{me\_volume}} \mid \mathbf{R}_{\mathbf{me\_plane}} \mid \mathbf{R}_{\mathbf{me\_line}} \mid \mathbf{R}_{\mathbf{me\_point}}\rangle$$
$$\mathbf{R}_{\mathbf{me\_volume}} = \langle(\mathbf{R}_{\mathbf{me\_shell}})\rangle$$
$$\mathbf{R}_{\mathbf{me\_shell}} = \langle(\mathbf{R}_{\mathbf{me\_plane}})\rangle$$
$$\mathbf{R}_{\mathbf{me\_plane}} = \langle E_{\text{plane}}, (\mathbf{R}_{\mathbf{me\_bound}})\rangle$$
$$\mathbf{R}_{\mathbf{me\_bound}} = \langle(\mathbf{R}_{\mathbf{me\_line}})\rangle$$
$$\mathbf{R}_{\mathbf{me\_line}} = \langle E_{\text{line}}, (\mathbf{R}_{\mathbf{me\_point}})^2\rangle$$
$$\mathbf{R}_{\mathbf{me\_point}} = \langle\mathbf{R}_{\mathbf{vert}}\rangle$$

The algebraic approach can be extended to nongeometric and nongraphical information. The maximal element representation is fundamental to shape grammars. Different grammar formalisms exist that include nongeometric information such as color and line thicknesses. nongraphical information such as labels or other types of weights can also be included. A simple example is shown below:

$$\mathbf{R}_{\mathbf{me\_point}} = \langle\mathbf{R}_{\mathbf{vert}}, [\mathbf{R}_{\mathbf{weight}}]\rangle$$

Table 5 Maximal element representation.

Representation	Polygon based boundary representation.
Elements	*Volume segment, shell, plane segment, boundary, line segment* and *point.* All are explicit, all are unique, except *line segment* and *point.* The topology is explicit only top down; the geometry is attached to *points* (and *line* and *plane segments*).
Topology	*Volume segment*: ordered list of *shells.* *Shell*: ordered list of *plane segments.* *Plane segment*: ordered list of *boundaries.* *Boundary*: ordered list of *line segments.* *Line segment*: two bounding *points.*
Geometry	*Plane segment*: equation of a plane. *Line segment*: equation of a line. *Point*: 3D coordinates.
Scope	*Nonregular*: mixed dimensional shapes as arrangements of *volume, plane, line segments* and *points.* Intrinsically nonmanifold. *Holes*: multiple *shells* (*boundaries*) for a single *volume* (*plane*) *segment*, of which one outer and all other inner.
Remarks	*Canonical*: Maximal *volume, plane* and *line segments*. Ordered lists of elements.

7 SUBSUMPTION LATTICE

The four representations taken together can be diagrammed by their subsumption ordering relations, as shown in Figure 5. The representations are classified abstractly, by the conditions each is able to represent. It is instructive to note that the subsumption lattice defined here differs from inheritance lattices used in object oriented languages and databases: entities with more attributes are higher in the lattice than those without the attributes.

The four representations, namely the winged edge, half edge, winged triangle and maximal element representations are respectively denoted by $\mathbf{R_{we}}$, $\mathbf{R_{he}}$, $\mathbf{R_{wt}}$, and $\mathbf{R_{me}}$. The ordering is specified by the rules in Table 6.

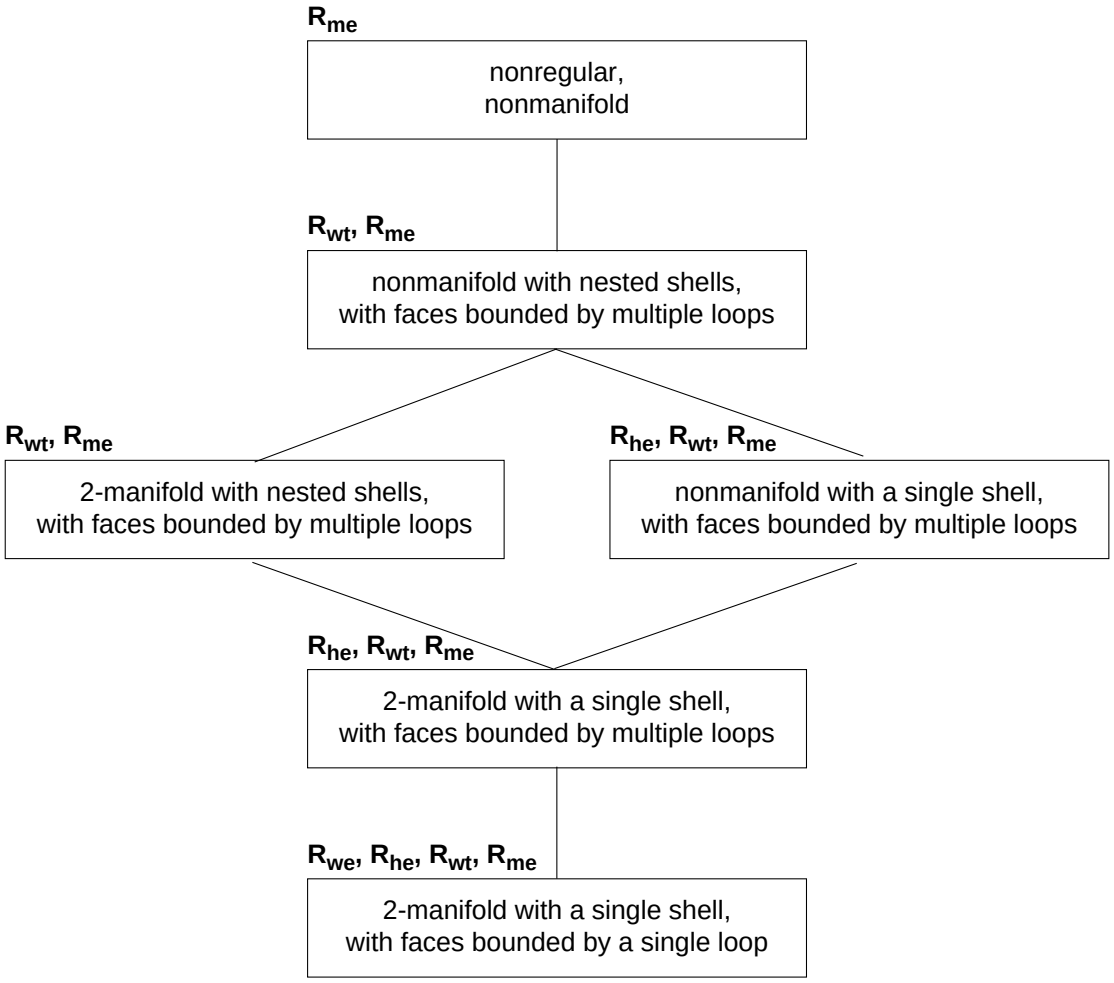

Figure 5 The subsumption lattice for the four solid modeling representations.

Table 6 Subsumption relations between the four representations.

Subsumption	*Rules*	*Explanation*
$\mathbf{R_{wt} \leq R_{me}}$	rule 1.6	$\mathbf{R_{me}}$ includes nonregular geometries.
$\mathbf{R_{he} \leq R_{wt}}$	rules 2.2 and 2.3	$\mathbf{R_{wt}}$ allows for nested shells.
$\mathbf{R_{we} \leq R_{he}}$	rules 2.2 and 2.3	$\mathbf{R_{he}}$ includes nonmanifold solids and allows multiple loops on faces.

As can be seen, the maximal element representation subsumes all the others, meaning that it can represent all individuals of the other representations. The winged edge representation is the most limited and cannot represent many conditions found in the other representations. It is subsumed by all three representations. The winged triangle representation subsumes the half edge representation: they both allow for nonmanifold solids and for faces with holes, but the winged triangle representation also allows for nested shells, whereas a half edge solid is composed of a single shell. The reason for this is that the half edge representation does not have a shell element. However, the representation can easily be extended to include a shell element. The resulting, extended, half edge representation is then equivalent to the winged triangle representation.

8 DISCUSSION

Exact translations can always be made from one representation to another that subsumes it. That is, a representation lower in the lattice can always translate its data into one above it, either directly or based on the transitivity relation. Notice that under restricted conditions, translations may be made from a more general representation to one that it subsumes. For example, a maximal shape that carries only a regular solid may be translated, with no loss of information, to the half edge; if it also is a 2-manifold and has only a single shell and faces with single loops, to the winged edge. Predicate checking for these conditions can ascertain for a set of objects to be translated, which ones can and cannot be translated without information loss. Only exact translations can be treated in this way. Thus, the approximation of a B-spline surface with a facetted (polygonal) surface cannot be directly dealt with using the concepts presented.

The rules of derivation in a representation are domain specific. In the arc example, rules of geometry define the derivations that can be applied. Those derivations are the core of a translation process. The extension of a representation **R** is those derivations that are needed to generate another equivalent representation **R'**. The process of translation then becomes the process of generating the base representation for representation **R'** by regrouping the attributes in both the base and the derived sets of attributes in **R**.

The ordering of representations allows their rigorous comparison at the semantic level and provides a basis for rigorous translation. As different applications are developed for use throughout the building life cycle, the application of such comparisons will be necessary, if translations are to be made automatically and robustly.

Note

The work by Charles Eastman was supported by the National Science Foundation, grant number IRI 9319982. This author benefitted from discussion with George Stiny.

9 REFERENCES

Autodesk (1992) Drawing Interchange and File Formats, a chapter in *AutoCAD Release 12 Customization Manual,* Autodesk, Sausalito, California.

Bloor, M.S. (1991) STEP-standard for the exchange of product model data in *Standards and Practices in Electronic Data Interchange,* The Institution of Electrical Engineers (IEE), IEE Colloquium, 2/1-3, London.

Baumgart, B.G. (1975) A Polyhedron Representation for Computer Vision in *National Computer Conference 1975*, AFIPS Press, Montvale, N.J, 589-596.

Braid, I.C., R.C. Hillyard and Stroud, I. A. (1980) Stepwise Construction of Polyhedra in Geometric Modeling in *Mathematical Methods in Computer Graphics and Design (*ed. K. W. Brodlie) Academic Press, London.

Cardelli, L. and Wegner, P. (1985) On Understanding Types, Data Abstraction and Polymorphism. *Computing Surveys*, **17**, 471-523.

Eastman, C. (1993) Life Cycle Requirements for Building Product Models. *International Conference on Information Technology in Construction,* Singapore, August 1993.

Eastman, C. M. (1994) Talk on Product Modeling. Presented at Engineering Design Research Center, Carnegie Mellon University.

Foley, J.D., Van Dam, A., Feiner, S. and Hughes, J. (1992) *Computer Graphics: Principles and Practice*. 2nd edition. Addison-Wesley, Reading, Massachusetts.

Krishnamurti, R. (1992) The Maximal Representation of a Shape. *Environment and Planning B: Planning and Design,* **19**, 267-288.

Mäntylä, M. (1988) *An Introduction to Solid Modeling*. Computer Science Press, Rockville, Maryland.

Paoluzzi, A., Ramella, M. and Santarelli, A. (1989) Boolean Algebra over Linear Polyhedra. *Computer Aided Design*, **21**, 474-484.

Pooch, U. and Neider, A. (1973) A Survey of Indexing Techniques for Sparse Matrices. *Computing Surveys*, **5**, 109-133.

Smith, B., Rinaudot, G. R., Reed, K. A. and Wright, T. (1988) *Initial Graphics Exchange Specification (IGES), Version 4.0*. SAE/SP-88/767, Society of Automotive Engineers, Warrendale, Pennsylvania.

Stone, H.S. (1973) *Discrete Mathematical Structures and Their Applications*. Science Research Associates, Chicago.

Stouffs, R. (1994) *The Algebra of Shapes*. Ph.D. Dissertation. Department of Architecture, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Stouffs, R. and Krishnamurti, R. (forthcoming) Sorts: an algebraic approach to representations. manuscript. Architecture and CAAD, Swiss Federal Institute of Technology, Zurich and Department of Architecture, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Yamaguchi, F. and Tokieda, T. (1985) Bridge Edge and Triangular Approach in Solid Modeling in *Frontiers in Computer Graphics* (ed. T. L. Kunii) Springer-Verlag, Tokyo, 44-65.

Rudi Stouffs is a Researcher at the Chair for Architecture and CAAD, Swiss Federal Institute of Technology, where he focuses on issues of data loss in collaborative and multidisciplinary design environments. Dr. Stouffs is currently developing an abstract representational model that allows for comparing data representations and managing data integrity in data exchange and communication. He received his Ph.D. in Architecture from Carnegie Mellon University, in 1994. Subsequently, he was an Assistant Professor in the Department of Architecture at CMU where he taught both graduate core courses in the area of computational design as well as a freshman computer modeling course, and advised graduate students.

Ramesh Krishnamurti is Professor in the Department of Architecture at Carnegie Mellon University. Previously he was at Bolt, Beranek and Newman, University of Edinburgh and the Open University. His research interest is in computational design with emphasis on the formal and algorithmic aspects of generative construction. His activities [past and present] has a multidisciplinary flavor and include spatial grammars, spatial algorithms, geometrical modeling, architectural analysis, knowledge based design systems, integration of natural language and graphics, user interfaces, computer simulation, graphical programming environments, and war games. He presently teaches courses on computer and geometrical modelling, shape grammars, and design support systems. He is Regional Editor (Americas) of *Building and Environment* and serves on the editorial board of *Languages of Design.*

Charles Eastman is Professor in the Colleges of Architecture and Computer Science at Georgia Institute of Technology, Atlanta. Professor Eastman's research interests are in engineering databases, geometric modeling and design theory. Over twenty years, he has developed a number of CAD systems and engineering databases, both as research prototypes and commercially. Previously, he was Director of the CAD-Graphics Lab at Carnegie Mellon University and the Director of the Center for Design and Computation at UCLA. He currently holds editorial positions on four journals: *Research in Engineering Design*, *Automation in Construction*, *Computer-Aided Design*, and the electronic journal *Information Technology in Construction.*