Dimension-Independent Convex-Cell Based HPC: Representation Scheme and Implementation Issues *

Valerio Pascucci[§] *Vincenzo Ferrucci*[‡] *Alberto Paoluzzi*[‡§]

[‡]Dip. di Disc. Scientifiche: Sez. Informatica, Terza Università
Via Segre 2, 00146 Roma, Italy

[§]Dip. di Informatica e Sistemistica, Università "La Sapienza"
Via Buonarroti 12, 00185 Roma, Italy

Abstract

This paper is devoted to the discussion of some of the design issues which concern the representation scheme underlying the geometrical nucleus of the PLASM design language. A central choice in the representation of geometry within the language was to store the minimum possible amount of topology. E.g., the boundary of a polyhedron in $\Re^3$, which is a set of 3D polygons, is stored without adjacencies. Thus, each such boundary polygon is considered as an elementary polyhedron within its affine hull, and represented as a solid 2D decomposition in convex cells. This representation allows to easily perform some operations that are not usually available in a geometric modeler, as well as to represent both manifolds and non-manifolds in a uniform manner.

In particular, each geometrical object is represented as a multilevel hierarchical structure (Hierarchical Polyhedral Complex, HPC) that collects a set of elementary polyhedra, affinely mapped from their local coordinate systems and grouped as a whole. Two representation are provided for the same geometry: a *weak* and a *complete* representation. In a weak representation the set of elementary polyhedra collected in a hierarchical structure constitutes a covering of the required pointset. Conversely, in a complete representation the elementary polyhedra are assumed to be pairwise disjoint. The validity check of weak geometrical data (*polyhedral sequences*) can therefore be performed at a syntactical level, whether the test over complete geometrical data (*polyhedral complexes*) is much harder. A conversion function is given that allows to transform a weak polyhedral sequence into a complete polyhedral complex, by removing the intersections between polyhedra. For each operator (defined both on sequences and complexes), the conversion function establishes a homomorfism between the spaces of weak and complete representations. Each operator can so be implemented within the domain that seems more appropriate, performing possible conversions only when really necessary.

*This work was partially supported from Italian Research Council, with contract n. 91.03226.64, within the "PF Edilizia" Project.

Solid Modeling '95, Salt Lake City, Utah USA

1 Introduction

The main novelty that the representation discussed in this paper exhibits with respect to standard current solid modeling technology is the decision of storing as little topology as possible. We noted in fact that the (very simple) adjacency data structures used in our previous prototype modelers [7, 3] are quite unused when a solid model is rendered by some graphics server, and even when the model is used to compute the integral properties [2] of the object. Hence, following the old, good CSG approach [12], we decided to associate to any model an acyclic multigraph which stores the hierarchical object structure, i.e. both the operations and the sub-components which generate any object component. We believe that such an information is much more interesting for the designer than the availability of the complete adjacency relations in some non-manifold subset of the boundary. In a few words, with respect to the representation of topology, geometry and structure, the representation scheme proposed here can be seen exactly half-way between solid modeling and contemporary standard graphics. This representation scheme is being extensively used within the geometry engine of the functional design language PLASM [8, 6].

Another choice, again inspired from CSG, consists in

the use of implicit linear inequalities for the representation of convex cells. The authors believe this choice has strong implications on the efficiency and coverage of the proposed scheme. Several basic operations, e.g. the extrusion of cells and the product of complexes (see [1]) become natural and efficient. A similar influence is expected on the usual Booleans, a point which is subject of on-going research. Last but not least, this approach can be also extended to curved cells described as the intersection of implicit nonlinear inequalities.

Some ideas that we discuss in the present context present similarity with those proposed by Rossignac and Requicha in [14], where the *Constructive Non-Regularized Geometry* representation, a broad generalization of CSG schemata [12], is introduced and fundamental algorithms for a very large class of solid and non-solid objects are discussed (see also Takala [15] and [10]). This class includes decompositions of objects with dishomogeneous material properties, incomplete boundaries and mixed dimensionality. The CNRG scheme retains the CSG merits, and in particular the mapping of the model to a generating expression and the independence from both the specific set of primitive shapes used and their representation. Primitive shapes are here replaced by polyhedra of any dimension, defined in their minimal affine hull, and possibly mapped in a higher dimensional embedding space. Motivations for a dimension-independent approach to solid modeling are widely discussed in [13] and [5]. Motivations for a functional approach to geometric programming and modeling are given in [6]. A short review of the literature on design languages can be found in [8]. Some design choices peculiar to the representation described in the following are based on the solid modeling experience gained at "La Sapienza" in the development of the 3D polyhedral modeler *Minerva* [7, 2] and the multidimensional modeler $Simple^n_X$ [5], currently extended with the HPC representation and used as the geometry engine of the PLASM language.

The paper is structured as follows. In Section 2, some entities (convex cells, d-complexes, d-polyhedra, embeddings, affine mappings and polyhedral complexes) are defined which allow sets of polyhedra to be represented and manipulated as a whole. In Section 3 the complete and weak representations schemata are introduced and discussed, along with their interrelationship. Section 4 illustrates some ideas on the implementation of operators versus their algebraic properties. Our belief is that they may provide the system designer with important guidelines for his implementational choices. Finally, some extensions of the proposed scheme are introduced in the conclusion Section, concerning inhomogeneous dimensionality, partially open and curved cells. The solution outlined for the first two points is already being experimented in PLASM.

2 Background

In this Section are given the mathematical concepts needed to describe the domain of the HPC representation scheme, i.e. the set of abstract mathematical objects which will be represented into the scheme.

2.1 Hulls, Cells and Mappings

We recall here some basic mathematical concepts. Among the others, definitions are given for the affine and convex hull of a set of points, and the notions of convex cell, of its faces and of d-complex. Also, the embedding and affine transformations are discussed, as well as their coordinate representation.

Affine and Convex Hull The ambient space considered in the following is the d-dimensional Euclidean space E^d represented by the vector space $\Re^d$. Given a finite set of points $V \subset E^d$ and a corresponding set of reals, the following "hulls" of V are defined:

Definition 1 *Let a finite set of points* $V = \{v_1, v_2, \ldots, v_h\} \subset E^d$ *and real numbers* $\lambda_1, \lambda_2, \ldots, \lambda_h$ *be given. Then:*

a). The affine hull *of* V *is the set of points*

$$\operatorname{aff} V = \{x = \sum_i \lambda_i v_i \mid v_i \in V, \sum_i \lambda_i = 1\}.$$

b). The convex hull *of* V *is the set of points*

$$\operatorname{conv} V = \{y = \sum_i \lambda_i v_i \mid v_i \in V, \sum_i \lambda_i = 1, \lambda_i \geq 0\}.$$

Points $x \in \operatorname{aff} V$, $y \in \operatorname{conv} V$ in E^d are said to be affinely and convexely dependent on the points in V, respectively. A set of points is called affinely (convexely) independent if none of its points belongs to the affine (convex) hull of the others. The dimension of a hull of V is the maximum number (decreased by one) of its affinely independent points. If V is a set of independent points, then $\dim \operatorname{aff} V = |V| - 1$.

Cells, Complexes and Polyhedra A linear, convex, compact and full-dimensional subset of E^d is called *convex cell*. A convex cell can be defined in two equivalent ways, either as convex combination of points or as intersection of affine halfspaces:

Definition 2 *A* convex cell *c of dimension d, or d-cell, in E^d is equivalently defined as:*

a). The convex hull of $h \geq d+1$ convexely independent points V, the vertices of the cell, $d+1$ of which are affinely independent.

b). *The bounded subset of E^d whose elements satisfy a system of $l \geq d+1$ nonsingular and nonredundant inequalities.*

We recall that, given a system of linear inequalities, an inequality $a_{i,0} + \sum_j a_{i,j}x_j \leq 0$ is *singular* if it can be substituted by the corresponding equation $a_{i,0} + \sum_j a_{i,j}x_j = 0$ with no change in the solution set of the system. An inequality $a_{i,0} + \sum_j a_{i,j}x_j \leq 0$ is *redundant* if it can be eliminated with no change in the solution set of the system. An inequality $a_{i,0} + \sum_j a_{i,j}x_j \leq 0$ is called *valid* with respect to a convex cell c, if $c \subset \{x |\ a_{i,0} + \sum_j a_{i,j}x_j \leq 0\}$.

The two descriptions of a convex cell c given in Definition 2 are easily seen to be equivalent. They both define a convex set, in the first case as a convex hull of vertices, in the second as the finite intersection of convex sets. Moreover, the defined set has the same dimension of the embedding space, in the first case because it contains $d+1$ affinely independent points, in the second because the defining inequalities are nonsingular. Strictly speaking, the additional constraints requiring convex independence of the vertices and nonredundancy of the defining inequalities are not necessary to correctly define the cell. Rather, they have been introduced with the purpose of imposing a "minimal" description of the cell, where no redundant elements appear.

Definition 3 *Let $a_0 + \sum_j a_j x_j \leq 0$ be a valid inequality with respect to the cell c. The set $f = \{x \in c |\ a_0 \sum_j a_j x_j = 0\}$ is a k-face of c if* $\dim \text{aff}\, f = k$.

A maximum order face of the cell c is called facet. The set of all k-faces of c, of any order k, is denoted by $\mathrm{F}(c)$. A collection of convex cells is a d-complex if it satisfies some restrictions:

Definition 4 *A d-complex $\mathcal{K}$ in E^d is a finite collection of d-cells such that for each pair of distinct cells $c_i, c_j \in \mathcal{K}$, either $c_i \cap c_j = \emptyset$, or $c_i \cap c_j \in \mathrm{F}(c_i) \cap \mathrm{F}(c_j)$.*

According to the definition above, a d-complex is a set of convex cells in E^d which are either disjoint or intersect along their k-faces. Notice that a d-complex consists only of full-dimensional cells, and therefore differs from the notion of cell complex of algebraic topology, where a complex contains also lower-dimensional entities. The computation of lower dimensional cells of a d-complex can be accomplished, when explicitly required, through the k-skeleton extraction operator [9].

The support space of a d-complex is the set union of the pointsets corresponding to its cells. The definition of polyhedron follows:

Definition 5 *A d-polyhedron $P \subset E^d$ is the support space of some d-complex $\mathcal{K}$.*

Hence, a d-polyhedron may be nonmanifold, nonconvex and/or unconnected, but is required to be homogeneously d-dimensional. The interior of the polyhedron P is denoted by $\overset{\circ}{P}$. The boundary ∂P of a polyhedron P is the set of points of E^d which belong neither to the interior nor to the exterior of P. In this context, a polyhedron is always associated to a d-complex, and therefore to a set of disjoint convex cells which meet only along their faces. For each facet of a convex cell the equation of the corresponding affine hull, the *facet hyperplane*, is obtained by restricting one of the defining inequalities. This facet may either be a subset of the polyhedron boundary, or separate two adjacent cells. In the latter case it belongs to the polyhedron interior.

The representation of the i^{th} facet of a cell c is given by the row vector $(a_{i,0}, a_{i,1}, \ldots, a_{i,d})$, called *facet covector*, whose elements are the coefficients of the facet inequality:

$$a_{i,0} + a_{i,1}x_1 + \ldots + a_{i,d}x_d \leq 0.$$

Notice that facets of a cell are associated to vectors in $\Re^{d+1}$, while vertices are associated to vectors in $\Re^d$. This asymmetry can be eliminated by using the representation of points in E^d by means of homogeneous coordinates [11]. Hence, we assume to represent vertices as column vectors and facet covectors as row vectors. Then, for a cell c with vertex vectors $\{v_1, v_2, \ldots, v_h\}$ and facet covectors $\{f_1, f_2, \ldots, f_l\}$, the following inequalities are satisfied:

$$f_i v_j \leq 0, \qquad i = 1, \ldots, l, \quad j = 1, \ldots, h. \quad (1)$$

This condition clearly shows the symmetry between facets and vertices in the definition of a cell. Choosing some arbitrary point x as the independent variable in (1) instead of the v_j's, the inequalities reduce to the system of l inequalities defining the cell pointset (see Definition 2b). From the latter, all internal and boundary points of the cell can be obtained. The set of vertices is given by the feasible solutions which are obtained by restricting to equations d rows of the system, such that the corresponding face covectors f_i are linearly independent, in all possible ways.

Analogously, an arbitrary covector w can be chosen as the independent variable instead of the f_i's. Thereby the system (1) is reduced to h inequalities which define, in the dual space, the convex set of all covectors representing valid inequalities for c. The facet hyperplanes of the cell coincide with the set of feasible covectors w which satisfy the restriction to equations of d inequalities such that the corresponding v_j's are linearly independent (affinely independent in non-homogeneous coordinates).

Embedding and Affine Mapping Polyhedra and d-cells have been defined as sets which have the same

dimension as their ambient space. To provide the capability of representing lower-dimensional sets embedded in a higher dimensional space in arbitrary affine subspaces, the embedding and affine mapping are defined as follows:

Definition 6 Embedding mapping *is a linear transformation* $E^d \rightarrow E^n$, $d \leq n$, *such that* $(x_1, \ldots, x_d) \mapsto (x_1, \ldots, x_d, 0, \ldots, 0)$.

Definition 7 Affine mapping *is a linear invertible transformation which maps points* $x \in E^n$ *to points* $y \in E^n$ *such that the homogeneous coordinates of* y *are a linear combination of the homogeneous coordinates of* x.

The embedding and affine mapping of a convex cell are easily computed. The embedding transformation is equivalently computed either (a) by pre-multiplying vertex vectors v by a rectangular matrix containing a block corresponding to the $(d+1) \times (d+1)$ identity matrix I and a $(n-d) \times (d+1)$ null matrix 0, or (b) by post-multiplying face covectors for the transpose of the same matrix:

$$\begin{bmatrix} I \\ 0 \end{bmatrix} v, \qquad f\,[I\ 0]\,.$$

The affine mapping is computed either by pre-multiplying vertex vectors by a matrix T or by post-multiplying face covectors by T', where both T and T' are square invertible $(n+1) \times (n+1)$ matrices. In fact, it can be seen that T' is the inverse of T. At this purpose, let assume that f_i^* and v_j^* are the images of f_i and v_j, respectively, under the affine mapping, and observe that the inequalities (1) must still hold after the mapping. We obtain:

$$f_i^* v_j^* = f_i T' T v_j = f_i v_i,$$

and therefore:

$$T'T = I \Rightarrow T' = T^{-1}.$$

The application of a linear transformation T to a d-polyhedron P, denoted by $T(P)$, is defined as the application of T to each d-cell of the complex underlying P. The transformation T may correspond to an embedding, an affine mapping, or to their composition.

2.2 Hierarchical Polyhedral Complex (HPC)

Polyhedral instances and complexes allow to construct hierarchical sets of polyhedra, and are inductively defined as follows. The basic definition is the one of 0-order polyhedral instance. Then, the j-order polyhedral complex (for $j \geq 1$) is defined by means of polyhedral instances of order less than j. Finally, the definition of k-order polyhedral instance is given using k-order polyhedral complexes.

Definition 8 *Polyhedral instances and polyhedral complexes are defined as follows:*

a). A 0-order polyhedral instance *is the set of points of* E^n *defined by the application of an embedding transformation and an affine map to a polyhedron* P.

b). A j-order polyhedral complex, *for* $j \geq 1$, *is a finite non-empty collection of polyhedral instances of order less than* j, *where at least one* $(j-1)$*-order instance appears and such that they intersect only along their boundaries.*

c). A k-order polyhedral instance *is the set of points of* E^n *defined by the application of an embedding transformation and an affine map to a* k*-order polyhedral complex.*

A polyhedral instance of some order k, for $k \geq 0$, is denoted by I. A polyhedral complex of some order j, for $j \geq 1$, is denoted by C.

The concepts defined above introduce a multilevel hierarchical structure which can be visualized as an oriented acyclic multigraph (see Figure 1). To avoid duplication of information, each complex or polyhedron can be referred any number of times within different instances. According to Definition 8, a polyhedral instance is not necessarily a full-dimensional set within its embedding space. When this is the case, the interior of a polyhedral instance I is, strictly speaking, empty. However, we regard in this case the relative interior and relative boundary as the interior and boundary of I, respectively:

$$I = T(P) \Rightarrow \begin{cases} \partial I = T(\partial P) \\ \mathring{I} = T(\mathring{P}) \end{cases}$$

Notice that at the level of the hierarchical polyhedral complex no topological information is stored, which implies e.g. that topologically different objects may correspond to the same structure. Nonetheless, such objects can be easily distinguished by considering the associated geometrical information, which is carried by the affine and embedding transformations.

3 Complete and Weak Representation

The set of *abstract mathematical models* considered in the scheme is denoted by $\mathcal{M} = \mathcal{P} \cup \mathcal{I} \cup \mathcal{C} \cup \mathcal{T}$, where $\mathcal{P} = \{P\}$, $\mathcal{I} = \{I\}$, $\mathcal{C} = \{C\}$ and $\mathcal{T} = \{T\}$ denote

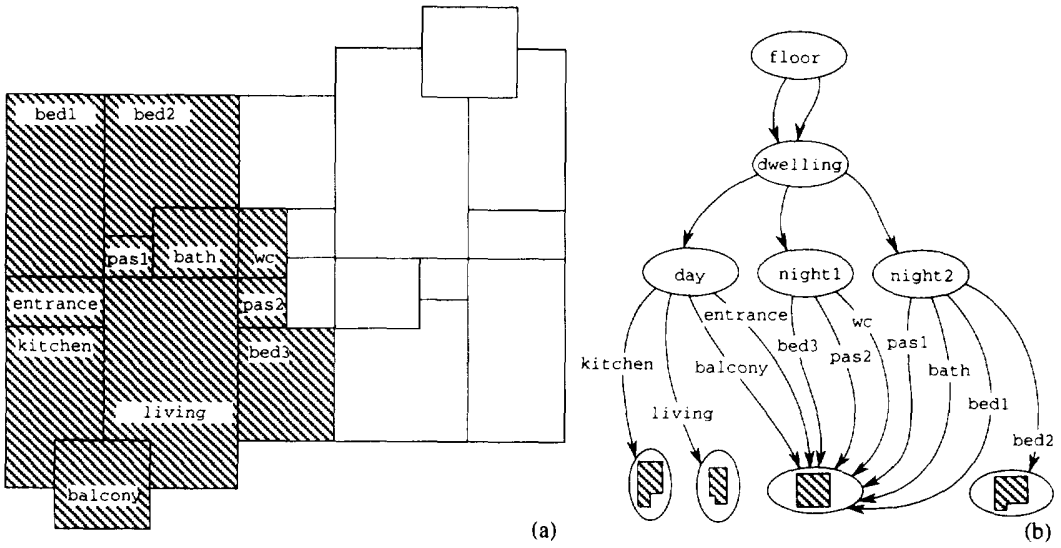


Figure 1: A polyhedral complex (plan of a house), with its multilevel hierarchical structure visualized as an oriented acyclic multigraph.

the sets of all polyhedra, polyhedral instances, polyhedral complexes and affine maps. The *representation* of a polyhedron, instance, complex and affine map is denoted by $\langle P \rangle \in \mathcal{R}$, $\langle I \rangle \in \mathcal{R}$, $\langle C \rangle \in \mathcal{R}$ and $\langle T \rangle \in \mathcal{R}$, where $\mathcal{R}$ is the set of all valid representations.

3.1 Decompositions with Convex Cells

The representation $\langle P \rangle$ of a polyhedron P is here presented, together with a short discussion of the main aspects that characterize and motivate its actual definition as a decomposition with convex cells, including some issues that concern the robustness.

The representation of a polyhedron immediately follows from its definition as support space of a d-complex. Therefore it is composed by a set of convex cells, each represented by its vertices and facets. The internal topological structure of a polyhedron is given by the adjacency information associated to each cell facet. Each cell facet is in fact coupled to the identifier of the adjacent cell in the d-complex, where null adjacency symbols are related to the facets belonging to the polyhedron boundary.

The whole cell description is given symbolically, whether a set of numeric data are separately stored in the polyhedron representation. If such numeric data are the coordinates of the cell vertices, we have a *vertex-based* representation. If the numeric data stored are the covectors of the cell facets, we have a *facet-based* representation. The two kinds of representation differ consequently for the symbolic description of the convex cells. In the first case each vertex is coupled with a vector identifier, whether each facet is implicitly described by the set of vertices incident to the cell. Symmetrically, in the facet-based representation, the facets are coupled to symbolic references to covectors in the polyhedron, whether each vertex in a cell is implicitly represented by the set of all incident facets.

The wide use of symbolic data in the polyhedron representation allow to set up its geometrical and topological structure when firstly defined. When the polyhedron is successively manipulated, some inconsistency may occur between numeric and symbolic data. In this case the latter are regarded as more reliable, as more stable than numeric information. Consider, e.g., an affine transformation like a simple rotation applied to the vertices of a square embedded in E^3. The transformed vertices may return perturbed such that they are no more exactly coplanar. Hence they define a (thin) tetrahedron instead of an embedded square (see Figure 2a). In the HPC representation this is not a problem since any square is locally defined as a full dimensional object in E^2, possibly embedded in E^3. Therefore, it cannot be assumed as intrinsically 3-dimensional.

Similarly, the four vertices of a squared facet of a cube may occur not to be exactly coplanar, such that the cube facets increase their number by one (see Figure 2b). The set of adjacent cells changes and, even worse, two adjacent cells may intersect each other, so violating the definition of d-complex (see Figure 2c). The symbolic data in the HPC representation allow to solve such problems, since the four vertices are considered coplanar if (and only if) their identifiers are all members of the implicit description of some facet.

Using a facet-based representation we can deal with similar problems. For example the top vertex of a

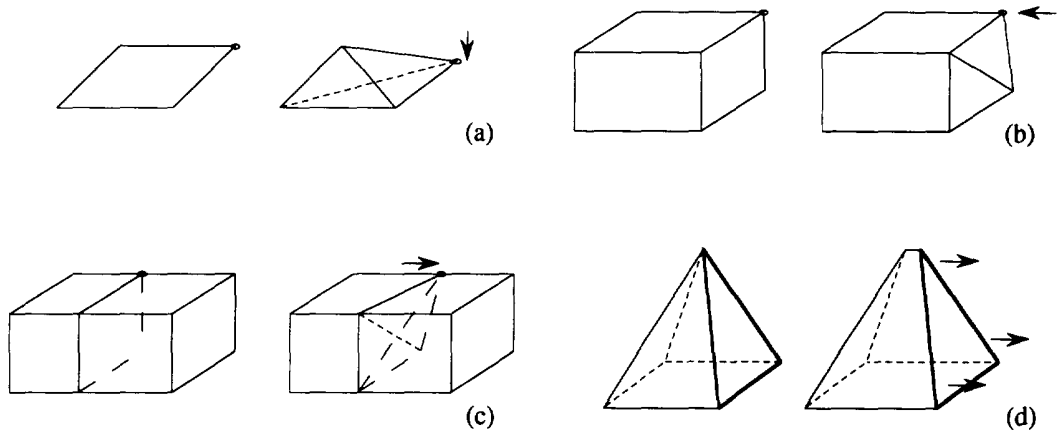


Figure 2: Four possible geometrical inconsistency problems, following from little modification of numerical values in a polyhedron representation.

square-based pyramid is numerically computed as the unique solution of a system of four equations (the equations of facet hyperplanes incident on the vertex). This system may become incompatible for little modifications of one facet covector: the four facet covectors so define two (close) vertices instead of one (see Figure 2d). The symbolic data in the HPC representation allow to detect such an error, since the implicit description of the vertex forces the four facets to intersect each other in a single point.

We can say that the the constraint of using only full dimensional polyhedra, symbolically described as much as possible, allow to arrange a robust representation. Numeric computations are considered less reliable than symbolic ones, hence they are avoided as much as possible. This also increases the efficiency of some algorithms on polyhedra, which take advantage from the reach symbolic description of the cells. E.g., the membership check of a vertex to some cell facet, in a facet-based representation, is simply performed by a set inclusion test of the facet symbol in the vertex implicit description. In a vertex-based representation the same check is a set inclusion test of the vertex identifier in the facet implicit description.

3.2 Complete Representation

The function $\mathbf{r} : \mathcal{R} \rightarrow \mathcal{M}$ maps valid representations to the corresponding abstract models. An element from the domain $\mathcal{R}$ is mapped to an element of the range $\mathcal{M}$ as follows, where parentheses are used for ordered sequences and curly brackets for non-ordered sets. According to Requicha's terminology [12], the inverse function $\mathbf{r}^{-1} : \mathcal{M} \rightarrow \mathcal{R}$ is a representation scheme from the set of abstract mathematical models $\mathcal{M}$ to the set of representations $\mathcal{R}$.

$$
\begin{array}{lcl}
\langle P \rangle & \mapsto & \mathbf{r}\langle P \rangle = P; \\
\langle T \rangle & \mapsto & \mathbf{r}\langle T \rangle = T; \\
\langle I \rangle = (\langle T \rangle, \langle P \rangle) & \mapsto & \mathbf{r}(\langle T \rangle, \langle P \rangle) = T(P) = I; \\
\langle I \rangle = (\langle T \rangle, \langle C \rangle) & \mapsto & \mathbf{r}(\langle T \rangle, \langle C \rangle) = T(C) = I; \\
\langle C \rangle = (\langle I \rangle_1, \ldots, \langle I \rangle_m) & \mapsto & \mathbf{r}(\langle I \rangle_1, \ldots, \langle I \rangle_m) = \\
 & & \{\mathbf{r}\langle I \rangle_1, \ldots, \mathbf{r}\langle I \rangle_m\} = \\
 & & \{I_1, \ldots, I_m\} = C.
\end{array}
$$

The various geometric entities which are involved in the definition of the $\mathbf{r}^{-1}$ function are depicted in Figure 3 along with the corresponding placement within a hierarchy of representational domains. At the first level of the representation convex cells are stored as sets of vertices and face covectors. Next, polyhedra are maintained as collections of cells along with sets of vertices (facets). More abstract elements, i.e. polyhedral instances and complexes, are represented only by (possibly multiple) references to elementary components.

Hence, a complex can be modeled as a directed acyclic multigraph with a single source node (the root complex). Every internal node is associated to a component complex, and arcs departing from the relative node correspond to the polyhedral instances of the complex. Terminal nodes are associated to elementary polyhedra (see Figure 1). Notice how different instances can refer to the same polyhedron (complex), to which different affine maps are applied, thereby avoiding unnecessary duplication of information, a technique commonly found in standard graphics systems [4].

The progressive enlargement of the domains of abstract mathematical models corresponds to a similar increase of the representational power. Starting from the set of full-dimensional convex d-cells and going through polyhedra and instances, the HPC scheme captures polyhedral complexes which may correspond to

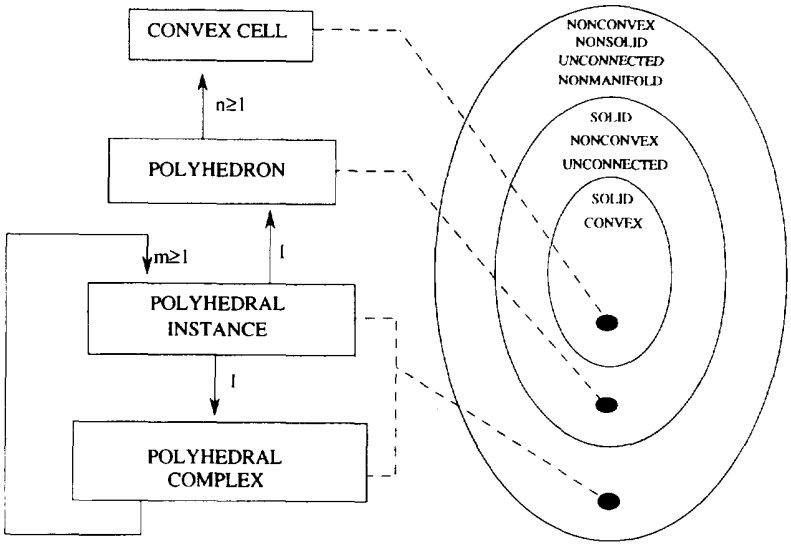


Figure 3: The correspondence between different classes of models captured by the $\mathbf{r}^{-1}$ scheme and the relative domains. Numbers next to the arrows indicate the cardinality of relationships.

nonsolid, nonconvex, unconnected and nonmanifold objects.

At the present time, dimensionally inhomogeneous objects fall outside of the domain of the HPC scheme. This restriction is not inherent to the technique that we propose, which can be readily extended to describe nonregular pointsets, but was rather motivated by the wish to quickly obtain a working software prototype, avoiding for the moment the issues connected to the implementation of non-regularized Booleans. We plan to devote further study to this point and remove the limitation within the future research effort, which will lead to the capability of describing more general object topology, according to the guidelines of [14].

3.3 Weak Representation

The "weak" representation scheme $\mathbf{wr}^{-1}$ is introduced in this Section. This scheme is based on an extension of the concept of polyhedral complex. At this purpose, we define the *polyhedral sequence*, which is derived by weakening the definition of polyhedral complex. Every polyhedral sequence can be associated to a valid complex according to the $\mathbf{r}^{-1}$ scheme by application of the *progressive difference* operator defined in the following. The weak representation scheme is both computationally more efficient (the evaluation of progressive differences is deferred as long as possible) and a more accurate mirroring of the semantics of the designing process.

3.3.1 Extended Polyhedral Instances and Sequences

We move to the larger domain which is needed to provide further flexibility to the object description by defining the *extended polyhedral instances* and the *polyhedral sequences*. In their definition the same reference mechanism is used that already appeared in the definition of instances and complexes.

Definition 9 *An* extended polyhedral instance $\tilde{I}$ *is the set of points of* E^n *defined by the application of an embedding transformation and an affine map either to a polyhedron* P*, or to a polyhedral complex* C*, or to a polyhedral sequence* S *(see next definition) in* E^d*, with* $d \leq n$.

Definition 10 *A* polyhedral sequence S *is an ordered set of* m *extended polyhedral instances* $(\tilde{I}_1, \tilde{I}_2, \ldots, \tilde{I}_m)$*, all embedded in the same space* E^n.

A polyhedral sequence is therefore similar to a complex, the main difference being that the first is an *ordered* set of *extended* polyhedral instances, and not a non-ordered set of polyhedral instances subject to a constraint of pairwise disjointness. Extending the notation, S denotes a sequence, $\mathcal{S}$ the set of polyhedral sequences, $\tilde{I}$ an extended polyhedral instance, $\tilde{\mathcal{I}}$ the set of extended polyhedral instances, and finally $\mathcal{M}^* = \mathcal{M} \cup \tilde{\mathcal{I}} \cup \mathcal{S}$ the extended set of all (extended) abstract mathematical models.

The representation of a sequence is denoted by $\langle S \rangle$, the representation of an extended instance by $\langle \tilde{I} \rangle$, and $\mathcal{R}^*$ is the set of all extended representations, including

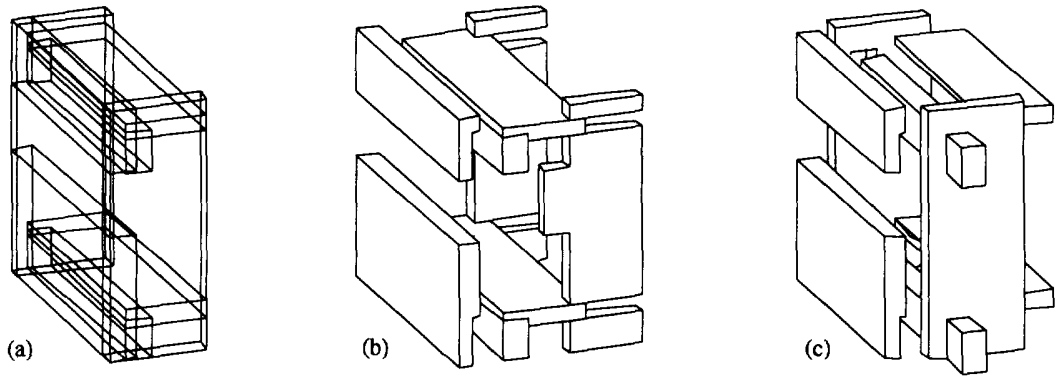


Figure 4: Two polyhedral sequences may differ only for the ordering of the composing polyhedral instances. The evaluation of progressive difference on different sequences may generate different polyhedral complexes.

polyhedral sequences and extended instances. To distinguish between $\mathcal{R}$ and $\mathcal{R}^*$, we use the terms *valid* or *complete* representations for the first and *extended or weak* representations for the second. The mapping between the set of extended representations and the set of extended models is given by the function

$$\mathtt{r}^* : \mathcal{R}^* \to \mathcal{M}^*,$$

which is obtained by extending $\mathtt{r}$ to the set of weak representations. Hence, we define $\mathtt{r}^* = \mathtt{r}$ in $\mathcal{R}$, whether $\mathtt{r}^*(\langle \tilde{I} \rangle) = \tilde{I}$ and $\mathtt{r}^*(\langle S \rangle) = S$.

3.3.2 Progressive Difference

Let "$-$" denote the regularized difference operator, defined as the closure of the interior of the set difference of the arguments [16]. The *progressive difference* function $\mathtt{D}$ is defined as follows, where $\mathcal{M}^* = \mathcal{M} \cup \tilde{\mathcal{I}} \cup \mathcal{S}$

$$\mathtt{D} : \mathcal{M}^* \to \mathcal{M}$$

is defined as follows:

$$\begin{array}{lll} m \in \mathcal{M} & \mapsto & \mathtt{D}m = m \\ \tilde{I} = T(S) & \mapsto & \mathtt{D}T(S) = T(\mathtt{D}S) \\ S = (\tilde{I}_1, \ldots, \tilde{I}_m) & \mapsto & \mathtt{D}(\tilde{I}_1, \ldots, \tilde{I}_m) = \{I_1, \ldots, I_m\} = C \\ & & I_1 = \mathtt{D}\tilde{I}_1, \\ & & I_i = \mathtt{D}(\tilde{I}_i - \tilde{I}_{i-1} - \ldots - \tilde{I}_1), \end{array}$$

for $i = 2, \ldots, m$.
From the evaluation rules given above, we see that the function $\mathtt{D}$ is the identity on $\mathcal{M}$. The evaluation of $\mathtt{D}$ on the extended instance $\tilde{I}$, mapping T of the sequence S, returns the (non extended) instance I, application of the same mapping T to the complex $C = \mathtt{D}S$.

When evaluated on a polyhedral sequence, $\mathtt{D}$ removes the intersections between the component instances. A polyhedral sequence is an ordered set, so that the ordering translates into a precedence rule while executing differences. According to such a rule, points which belong to more than one instance are considered as belonging to the one with the lowest index. Hence, every instance is subtracted from all instances which appear later in the sequence, according to the denotation of "progressive difference". An example of the application of the $\mathtt{D}$ function is shown in Figure 4 (see also [8]). Notice that the application of a progressive difference to two polyhedral sequences which differ only in the order of their instances may produce two different polyhedral complexes.

Shifting our attention from the set of models to the set of representations, analogously we define the $\mathtt{D}^*$ function:

$$\mathtt{D}^* : \mathcal{R}^* \to \mathcal{R}.$$

Its evaluation requires the algorithmic computation of the progressive difference on the representations of the arguments. Hence, it is by definition:

$$\mathtt{D}(\mathtt{r}^*(x)) \equiv \mathtt{r}(\mathtt{D}^*(x)), \qquad x \in \mathcal{R}^*.$$

The $\mathtt{D}^*$ function acts therefore on the representations in the same way in which $\mathtt{D}$ acts on models. The progressive difference is a key operator in the HPC scheme, because it allows for the description of geometries by using sequences instead of complexes.

3.3.3 Weak Representation Scheme

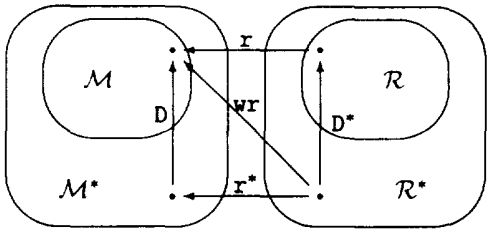
The "weak representation scheme" is introduced here, where elements of $\mathcal{R}^*$ can be used to directly describe polyhedral complexes. We define:

$$\mathtt{wr} : \mathcal{R}^* \to \mathcal{M},$$

where

$$\mathtt{wr} \doteq \mathtt{D} \circ \mathtt{r}^* \equiv \mathtt{r} \circ \mathtt{D}^*,$$

and where $\circ$ denotes functional composition. The $\mathtt{wr}^{-1}$ representation scheme deserves the designation *weak* because it allows for the representation of polyhedral complexes with elements of $\mathcal{R}^*$. Thus, the $\mathtt{wr}^{-1}$ scheme can be used in place of $\mathtt{r}^{-1}$, because the two mappings share the same domain $\mathcal{M}$. Summarizing, the domain and range of the r, $\mathtt{r}^*$, wr, D and $\mathtt{D}^*$ functions, which define the $\mathtt{r}^{-1}$, $\mathtt{r}^{*-1}$ and $\mathtt{wr}^{-1}$ representation schemata, are:



When using $\mathtt{wr}^{-1}$ in place of $\mathtt{r}^{-1}$, the set of available representations is enlarged from $\mathcal{R}$ to $\mathcal{R}^*$, thereby introducing a strong relaxation to the geometric constraints of the scheme. Every time that a valid representations according to the $\mathtt{r}^{-1}$ scheme must be derived, it is required the evaluation of the $\mathtt{D}^*$ function on the corresponding weak representation. Hence, we say that a valid representation is obtained by *evaluating* a weak representation.

The introduction of the weak representation scheme enriches the set of representations corresponding to the same abstract model. Such extension is due to the use of progressive differences, which can be traced back in turn to a CSG-like method. In particular, such an approach has been undertaken within a framework based on a decompositive approach.

4 Morphisms and Operators

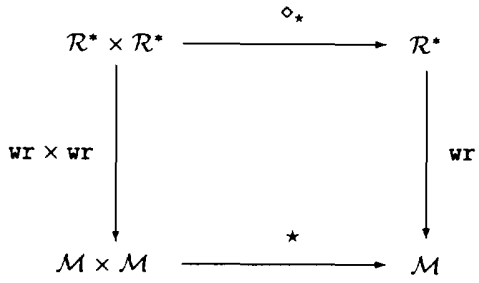
Every operator, denoted generically as $\star$, is assumed to be defined on the space of models to ensure that its semantics does not depend on the particular representation scheme which is used. Assuming a binary $\star$, we have:

$$\star : \mathcal{M} \times \mathcal{M} \to \mathcal{M}.$$

For every $\star$ operator defined on the models, a corresponding $\diamond_\star$ algorithm must be defined which specifies its action on the representations. The correctness of such algorithm is guaranteed by the condition:

$$\mathtt{wr}(x \diamond_\star y) = \mathtt{wr}(x) \star \mathtt{wr}(y), \qquad x, y \in \mathcal{R}^*.$$

Therefore, the wr function must be an homomorphism from $(\mathcal{R}^*, \diamond_\star)$ to $(\mathcal{M}, \star)$. The wr homomorphism can be visualized as in the following commutative diagram ($\mathtt{wr} \times \mathtt{wr}$ is the product function between product spaces):



The specification of the $\diamond_\star$ algorithm may differ according to the choice of its domain and range. Remembering that $\mathcal{R} \subset \mathcal{R}^*$ and that $\mathtt{D}^*(\mathcal{R}^*) = \mathcal{R}$, we may define four classes of algorithms, whose elements $\diamond^1_\star$, $\diamond^2_\star$, $\diamond^3_\star$ and $\diamond^4_\star$ are characterized by the following signatures:

$$\begin{aligned} \diamond^1_\star &: \mathcal{R} \times \mathcal{R} \to \mathcal{R}^* \\ \diamond^2_\star &: \mathcal{R} \times \mathcal{R} \to \mathcal{R} \\ \diamond^3_\star &: \mathcal{R}^* \times \mathcal{R}^* \to \mathcal{R}^* \\ \diamond^4_\star &: \mathcal{R}^* \times \mathcal{R}^* \to \mathcal{R} \end{aligned}$$

It is not necessary to implement the same operator in four different algorithmic flavors. The relationships existing between the four algorithm classes allow to implement only one version, and use it, when necessary, in place of the others. Hence, the choice of the algorithm which is effectively implemented is not imposed by the representation scheme but left to the system designer, who should be guided by both algorithmic and implementational considerations.

We recall that $\mathcal{R} \subset \mathcal{R}^*$, i.e. that every valid representation is also a weak representation. It follows that "valid" input data can be fed to algorithms originally designed and coded for "weak" input data. Similarly, an algorithm which returns a valid representation as output can substitute another one which is supposed to generate weak output data. In general, the use of $\diamond^4_\star$ in place of $\diamond^1_\star$, $\diamond^2_\star$ and $\diamond^3_\star$ is always formally correct. From a practical viewpoint, the extensive use of this property may be undesirable because of efficiency lack, due to the fact that geometric properties of valid representations may be not exploited within algorithms designed for weak data.

The use of the $\mathtt{D}^*$ function to convert a weak representation to a valid one, symmetrically allows to substitute the operator $\diamond^4_\star$ with an expression involving in turn only one of the other three. The following functional equations describe such transformations:

$$\begin{aligned} \diamond^4_\star &= \mathtt{D}^* \circ \diamond^1_\star \circ (\mathtt{D}^* \times \mathtt{D}^*) \\ \diamond^4_\star &= \diamond^2_\star \circ (\mathtt{D}^* \times \mathtt{D}^*) \\ \diamond^4_\star &= \mathtt{D}^* \circ \diamond^3_\star \end{aligned}$$

Summarizing, it is possible both to use $\diamond^4_\star$ in place of $\diamond^1_\star$, $\diamond^2_\star$ and $\diamond^3_\star$, as well to operate the inverse substitu-

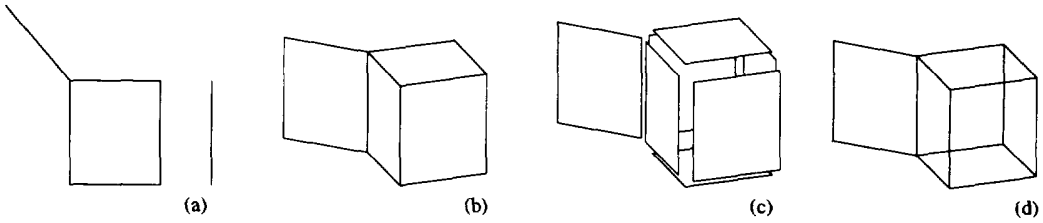


Figure 5: The product of complexes in (a) generates a polyhedral complex (b) with non-uniform intrinsic dimension. In (c) and (d) the (exploded) 2-skeleton and the 1-skeleton are shown, respectively.

tion. Thus, anyone among the four possible versions of $\diamond_*$ may be implemented to virtually guarantee the availability of all four. The possibility of choosing the more convenient implementation of any operator among the possible four provides a further degree of freedom in the design and implementation of a system based on weak representations.

An efficient implementation of class 4 is particularly convenient because it allows the substitution of the other three and eliminates the need of computing the D^* function. Nonetheless, implementations of class 3 are preferable in general, because their cascade application is possible (they have the same domain and range) while exploiting the advantages of the weak representation. The use of available algorithms developed in the framework of decompositive schemata leads to class 2 implementations, in which a valid representation is used for the operands as well as for the result.

The choice of the algorithm is also bound to the geometric meaning of the operator which is being defined, e.g. it is advisable to avoid implementations which are basically equivalent to a cascade application of the progressive difference and of the specific operation requested, since these steps should be maintained separate.

The same considerations apply to unary operators, with the possibility to define four different algorithmic versions for each operator defined on models, which differ for the input and output data representation. Each of the four implementations can potentially substitute the others, possibly using the D^* function to perform the needed data conversion.

5 Conclusions and Further Extensions

The characteristics of the HPC scheme presented in this paper are deliberately limited to the features that have been widely tested and for which a stable formalization has been established. Further enlargements to the scheme are currently experimented or planned. Three are the extensions we consider necessary to achieve the broader representational domain: (1) nonuniform intrinsic dimensionality of models, (2) redefinition of the convex cell as a partially open set, (3) use of curved facets. As stated in Section 3.2 the point (1) is encompassed by the current definition of the scheme, and supported in the implementation by the product of polyhedra and k-skeleton extraction operators (see Figure 5), described in details in [9]. Anyway, standard Booleans need more research and experimentation. Currently, we are able to compute only the intersection of polyhedral complexes.

About extension (2), we are testing a simple but effective method. A tag is associated to each face of a cell, which specifies if it is open or not, i.e. if its inequality must be considered a strict inequality. In Figure 6 a building model is shown obtained as the 2-skeleton of a set of simple blocks; the balconys differ from the internal spaces as they are defined as blocks which are open (strict inequality) on their top facet.

Extension (3) is necessary to represent curved objects, which are not currently supported. Anyway it is always possible to have good approximations, since any curved object can be approximated by using piecewise linear maps on simplicial decompositions (see Figure 7). This approach seems to be sufficient in most real cases and retains the advantages of simple and efficient linear mathematics. Anyway it is not enough in general and we plan to explore extensions for non-linear facet support.

Acknowledgments

The authors wish to thank the anonymous reviewers for many useful comments and suggestions.

References

[1] Bernardini, F., Ferrucci, V., Paoluzzi, A., and Pascucci, V. Product operator on cell complexes. In *Solid Modeling '93, Second ACM/IEEE Symposium on Solid Modeling and Applications*

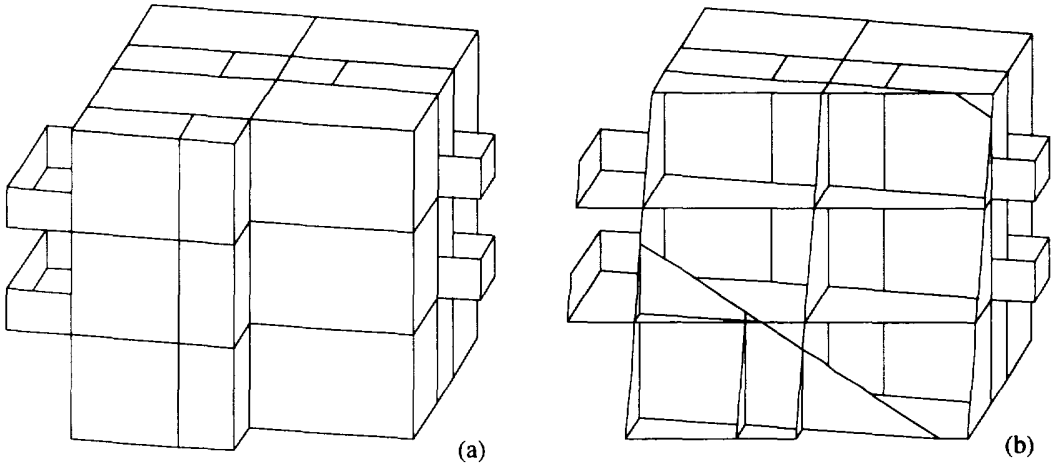


Figure 6: A simple building model, where balconies are obtained as (the 2-skeleton of) partially open cells. (a) dimetric view, (b) dimetric cutaway view.

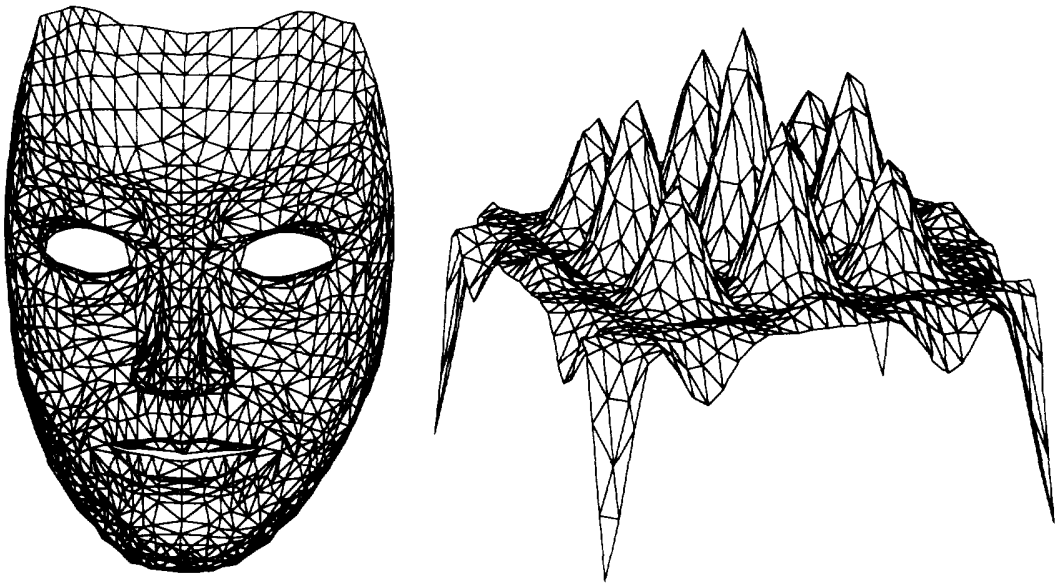


Figure 7: Two polyhedral approximation of curved objects.

(Montreal, CA, 1993), J. Turner, J. Rossignac, and G. Allen, Eds., ACM Press, pp. 43–52.

[2] CATTANI, C., AND PAOLUZZI, A. Boundary integration over linear polyhedra. *Computer Aided Design 22*, 2 (Mar. 1990), 130–135.

[3] FERRUCCI, V., AND PAOLUZZI, A. Extrusion and boundary evaluation for multidimensional polyhedra. *Computer Aided Design 23*, 1 (Feb. 1991), 40–50.

[4] HOWARD, T., HEWITT, W., HUBBOLD, R., AND WYRWAS, K. K. *A Practical Introduction to PHIGS and PHIGS PLUS*. Addison Wesley, Reading, MA, 1991.

[5] PAOLUZZI, A., BERNARDINI, F., CATTANI, C., AND FERRUCCI, V. Dimension–independent modeling with simplicial complexes. *ACM Transactions on Graphics 12*, 1 (Jan. 1993), 56–102.

[6] PAOLUZZI, A., PASCUCCI, V., AND VICENTINO, M. Geometric programming: A programming approach to geometric design. *ACM Transactions on Graphics*, accepted for publication, 1995.

[7] PAOLUZZI, A., RAMELLA, M., AND SANTARELLI, A. Boolean algebra over linear polyhedra. *Computer Aided Design 21*, 8 (Oct. 1989), 474–484.

[8] PAOLUZZI, A., AND SANSONI, C. Programming language for solid variational geometry. *Computer Aided Design 24*, 7 (1992), 349–366.

[9] PASCUCCI, V., FERRUCCI, V., AND PAOLUZZI, A. Dimension-independent convex-cell based HPC: Skeletons and product. To appear as Tech. Rep., Dip. di Informatica e Sistemistica, Università "La Sapienza", 1995.

[10] RAPPOPORT, A. A scheme for single instancing representation in hierarchical assembly graphs. In *Proceedings Modeling in Computer Graphics: Methods and Applications* (Genova, Italy, June 1993), Springer Verlag, pp. 213–223.

[11] REES, E. G. *Notes on Geometry*. Springer Verlag, Berlin, 1988.

[12] REQUICHA, A. A. G. Representations for rigid solids: Theory, methods and systems. *ACM Computing Surveys 12*, 4 (Dec. 1980), 437–464.

[13] REQUICHA, A. A. G., AND ROSSIGNAC, J. R. Solid modeling and beyond. *IEEE Computer Graphics and Applications 12*, 9 (Sept. 1992), 31–44.

[14] ROSSIGNAC, J. R., AND REQUICHA, A. A. G. Constructive non-regularized geometry. *Computer Aided Design 23*, 1 (Feb. 1991), 21–32.

[15] TAKALA, T. A taxonomy of geometric and topological models. In *Proceedings Eurographics Workshop on Mathematics and Computer Graphics* (S. Margherita, Genova, Italy, Oct. 1991).

[16] TILOVE, R., AND REQUICHA, A. A. G. Closure of Boolean operations on geometric entities. *Computer Aided Design 12*, 5 (Sept. 1980), 219–220.