RC 14340 (#64201) 1/18/89
Computer Science 32 pages

Research Report

SGC: A Dimension-Independent Model for Pointsets with Internal Structures and Incomplete Boundaries

Jarek R. Rossignac and Michael A. O'Connor

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, N.Y. 10598

Published as

"SGC: A Dimension-Independent Model for Pointsets with Internal Structures and Incomplete Boundaries", J. Rossignac and M. O'Connor, in Geometric Modeling for Product Engineering, Eds. M. Wosny, J. Turner, K. Preiss, North-Holland, pp. 145-180, Proceedings of the IFIP Workshop on CAD/CAM, 1989.

Published in Geometric Modeling for Product Engineering,
North-Holland, Rensselaerville, NY, pp.145-180
September, 1989

SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries

Jarek R. Rossignac
Michael A. O'Connor

IBM Research Division
P.O. Box 218
Yorktown Heights, New York 10598

Abstract

Geometric modelling is central to many CAD/CAM applications. Different applications deal with different geometric entities: **solids** are used for modelling mechanical parts; **lower dimensional objects,** such as faces and curves, may represent construction aides or regions of contacts between solids; **interior faces** are used to decompose solids into finite elements or into subsets exhibiting different physical properties; objects with **cracks** or **sets without boundary** are needed for modelling valid domains for certain differential equations; **higher dimensional objects** are the basis for path planning in configuration space and for modelling scalar or vector fields. Communication between such diverse application programs is made difficult because of incompatible definitions of valid objects and of different data structures used to represent them. To alleviate many integration problems, we propose a new modelling tool called ***Selective Geometric Complex*** (abbreviated SGC) which provides a common framework for representing objects of mixed dimensionality possibly having internal structures and incomplete boundaries. SGCs are composed of finite collections of mutually disjoint ***cells,*** which are open connected subsets of n-dimensional manifolds and generalize the concepts of edges, faces, and vertices used in most solid modelers. The connectivity between such cells is captured in a very simple incidence graph, whose links indicate "boundary-of" relations between cells. By choosing which cells of an object are "active" one can associate various pointsets with a single collection of cells. These pointsets need not be homogeneous in dimension, nor even be closed or bounded. Although most geometric manipulations necessary to support SGCs (at least in three dimensions) are available in many existing geometric modellers, data structures and high-level operations provided with these modellers are not designed to represent and process such complex objects. Therefore, to support useful operations on SGCs, we have decomposed Boolean and other set-theoretic operations (closure, interior, boundary) into combinations of three fundamental steps for which we have developed dimension-independent algorithms: a ***subdivision*** step, which makes two objects "compatible" by subdividing the cells of each object at their intersections with cells of the other object, a ***selection*** step, which defines "active" cells, and a ***simplification*** step, which, by deleting or merging certain cells, reduces the complexity of an object's representation without changing the represented pointset and without destroying useful structural information. Furthermore, combinations of these steps may produce a variety of special-purpose operations, whose effect is controlled by simple predicates, or filters, for cell selection.

1.0 INTRODUCTION

Research in Computer Aided Geometric Modelling originated in the 1960's and has concentrated on the interactive design and display of curves and surfaces [1], and later of solid part models [2]. Geometric modelling has been successfully applied in mechanical design, computer graphics, numerically controlled machining, and finite element mesh generation and analysis. It is expected to play an increasingly important role in the automation of design and analysis activities for manufactured parts, and in the generation of plans for manufacturing processes [3].

Because different applications focus on different aspects of the represented geometry and so require different processing capabilities, each application software utilizes its own, problem specific, design front-end and geometric representation that offers the best compromise between storage requirements and processing performance. For example:

- CSG or boundary representations of solids are used for the design of mechanical parts,
- subsets of curved surfaces are used to represent sculptured portions of car bodies,
- lines and circles are used to represent axes and construction aides for principal features of mechanical parts,
- structured lattices of points are used to represent decompositions of solids into finite element meshes,
- graphs describing assemblies of solids are used to represent solids composed of different materials such as the layers in silicon devices,
- representations of pointsets that are not closed are implicitly used for crack analysis and integral properties calculations,
- lower dimensional sets are used for analyzing tolerances or for representing regions of contact between solids,
- higher dimensional sets are used for path planning in configuration space or for modelling scalar or vector fields.

Often these representations are not compatible and cannot in general be derived from one another. They may not even be complete representations of a real part, but only reflect some specific aspects. The inherent communication problem remains a serious impediment to the integration of application packages into automatic design and analysis tools to provide higher functionality. Consequently, application systems are often enhanced individually to provide functions already available elsewhere. These enhancements can require the development of more complete representations (for example, to support the simulation of plans for numerically controlled machining, surface models were replaced by solid models) and certainly lead to costly redevelopments or at least re-implementations, of already existent fundamental algorithms.

To address the problem of integrating several applications into a common framework and thus share the core tools for user-input, geometric calculations, and graphics, we propose a new unified and complete representational scheme that captures collections of geometric objects of possibly different nature and dimensionality into a single model which explicitly reflects the geometric interrelations between these objects. Our scheme captures the aspects of a model that pertain to geometry and makes information about the geometry, connectivity, and incidence of the individual geometric components explicitly available. We represent geometric objects as finite collections of mutually disjoint (relatively open) ***cells*** in some n-dimensional manifold. In $\mathbf{R}^3$, these cells are connected open subsets of the three-dimensional space, of two dimensional surfaces, or of one dimensional curves, or simply points. The connectivity (often referred to as "topology") of such objects is captured in a graph whose links indicate incidence relations between cells and their boundaries. We call such a representation a **geometric complex.**

Geometric complexes can accommodate lower-dimensional cells that are not bounding any full dimensional cell, and thus are suitable for representing pointsets (often called "non-manifolds") that

are not homogeneous in dimension, as for example solids combined with dangling faces and edges in $\mathbf{R}^3$. Furthermore, geometric complexes can distinguish between a pointset and its internal (topological) structure by storing internal faces, or the location of some particular line or axis inside a solid.

Each cell c is an open connected submanifold of some supporting manifold M. The point-set represented by c is the interior of a region of M delimited by its boundary, ∂c, defined here as the set of points of the closure of c that are not in c. It is represented implicitly by a reference to M and to ∂c, which in turn is represented as a collection of mutually disjoint lower-dimensional cells (except when c is a vertex or has no boundary). For example, the representation of a face c may be composed of a surface M that contains c and of the collection of edges and vertices that bound c. The union of the pointsets of c and bc is closed. In any geometric complex, the boundary of each cell must also be represented as a collection of cells, and thus the pointset spanned by all cells is closed.

By "selecting" which cells of an object are **active** one can associate various pointsets with the same geometric complex. For example, one can declare that the interior of the face is **active** and that the bounding edges and vertices are **not active**. The pointset associated with such a geometric complex would then be limited to the interior of the face, although the edges and vertices are part of the structure of the geometric complex. The resulting representation is called a **Selective Geometric Complex** and is obtained by attaching to each cell of a geometric complex a list of attributes, one of which indicates whether the cell is active or not. The **active** attribute permits the representation of **non-closed pointsets**, such as a spherical ball from which one has extracted the central point, a solid cube with a missing face, or simply the interior of a disk in $\mathbf{R}^3$.

Other cell attributes may be used to associate information with individual cells and to provide the user or the application program with explicit information about the topological relations between cells. For example, a pointset composed of a set of lines in $\mathbf{R}^2$ is represented as a geometric complex composed of zero and one-dimensional cells, the open line segments and the intersection points. If the lines are named, one can associate with each cell a provenance attribute indicating the list of the original lines on which the cell lies. To find if there are points where more than two lines intersect, it is then sufficient to extract cells for which the provenance attribute has more than two entries.

A structure for representing complex geometric objects is useful in computer aided design and analysis, only if algorithms are available that support easy construction and interrogation of the data structure. Low level construction steps, like for example Euler operators [4,5,6,7], can be important development tools, but do not provide acceptable design interfaces. Higher level operations that support shape composition or that closely model the effect of manufacturing operations are needed [8]. In this paper, we propose a formalism for the various operations on SGCs that correspond to the usual Boolean set theoretic operations (union, intersection, difference) and also to more specific operations such as closure and interior which may be combined with Boolean operators to define regularized Boolean operators [9].

Although in geometric modelling the regularized operators are usually and justly stressed, the non-regularized operations can by themselves be important. For example, to compute which faces of a cube lying on a table are exposed to corrosion, one can produce a model of the set theoretic difference between a cube in $\mathbf{R}^3$ and the table on which it lies. That model has: one active 3D cell (the interior of the cube), six 2D cells (the interior of the faces) of which only five are active, twelve 1D cells (the interior of the edges) of which only eight are active, and eight 0D cells (the vertices) of which only four are active. The exposed faces of the cube may be simply obtained by extracting its active two-dimensional cells. Such a filter may easily be defined by the user as a high-level operation. Using the dimension, the connectivity, or other attributes of each cell, such as the history of creation [10], different cells may be selected as active by applying simple logical predicates. Such filters can be used to extract particular geometric features, such as dangling or interior faces or to produce regularized sets.

The Boolean and other set-theoretic operations, as well as many other special purpose operations, can be decomposed into three fundamental steps:

Subdivision,	which makes two objects "compatible" by subdividing their cells;
Selection,	which defines which cells are active;
Simplification,	which, by deleting or merging certain cells, reduces the complexity of an object without changing the represented point set nor destroying important structural information.

Data structures and set operations that are available in popular solid modellers [11,12] are not designed to represent and process non-regular pointsets with complex internal structures, and thus must be extended. However, the three steps mentioned above can be implemented using slightly modified versions of geometric algorithms that are already supported in many modelling systems, at least in three dimensions.

The data and the high-level algorithms for SGCs are defined recursively and thus are not limited by dimension, provided that geometric calculations (such as the intersection and decomposition of manifolds) are supported [13]. Consequently, SGCs may be used to model, for example, a six-dimensional pointset of possible positions in a robot's configuration space and their internal structure and cell attributes may be used to distinguish singular configurations.

The paper is organized as follows. Section 2 summarizes relevant aspects of current solid modelling technology and discusses some of its advantages and limitations. Section 3 introduces the fundamental concepts, defines SGCs, shows how they can be used to manipulate geometric objects of the types defined above, and briefly compares SGCs with other mathematical decomposition schemes for representing dimensionally non-homogeneous objects. Section 4 presents a simple but general data structure for SGCs and discusses possible variations in the light of similar efforts independently carried out elsewhere. Section 5 briefly presents the essential high-level operations for manipulating and merging SGCs and for extracting from SGCs the topological, structural, and geometric information commonly requested by application programs.

The objective of this paper is to introduce the theoretical foundations upon which the SGC structure and algorithms are based, to present the generality and the practical applications of SGCs, and to provide some guidelines for implementing the data structures and algorithms that support SGCs using geometric tools already available in many solid modellers. Thus, while recognizing its importance, we will sacrifice technical detail for the sake of clarity of exposition, with the plan to return to consideration of the more delicate or complicated points in other reports. In particular, due to space constraints, only the most important aspects and the overall organization of the algorithms are discussed; descriptions of the detailed algorithms that merge and simplify SGCs and low-level operations that add, subtract, and merge cells in SGCs is deferred to a separate article.

2.0 SOLID MODELLING FOUNDATIONS AND LIMITATIONS

Solid Modelling systems are essential for automating the design of manufactured parts and of manufacturing processes [14]. The interactive design, simulation and analysis tools they provide significantly reduce the cost of design activities by shortening the design cycle, improving the quality of the final products, and increasing the efficiency of manufacturing processes [15, 16, 17, 18, 19, 20, 21]. The impact of a solid modeller is heavily dependent on its coverage (the type of shapes that can be modelled), on its ease of use (facilities for expressing and modifying shape), and on its ability to efficiently perform the usual analysis functions.

Shape specification techniques supported by commercially available geometric modelling system generally fall into one of the two following categories: surface modelling and solid modelling. At-

tempts to combine both design schemes in the same system have been slow to emerge [22, Jare84., 23], partly because of a lack of representation schemes for objects of mixed dimensionality, and also because operations that combine such compound geometric objects have not so far been properly implemented nor even well defined.

Surface modellers allow users to define geometric entities of dimension two or less in a constructive manner, by defining them parametrically, or in terms of their boundary, or as intersections of other entities. For example, a B-spline patch is defined in parametric form; a 2D polygonal face in $\mathbf{R}^3$ may be defined by its bounding edges, which, could have been previously defined in terms of vertices; and a curve may be defined as the intersection of two surfaces. Surface modellers support composite geometric objects that combine possibly overlapping pointsets of different dimensions, but do not support explicit representation of 3D solids. Even though a collection of faces may implicitly represent a solid, the lack of algorithms for computing interferences between such solids or for modelling the effect of manufacturing operations on them restrict the use of surface modellers to graphics and surface machining.

More recent systems, called **solid modellers** [8] have been developed that provide an informationally complete representation for solid objects and support high-level operations on such models. They have a larger scope of potential applications including interference detection [24], mass property calculation [25], and automatic generation of plans for numerically controlled machining of parts [18]. Pointsets represented in current solid modellers are restricted to be "valid" solids in $\mathbf{R}^3$.

2.1 Valid solid models

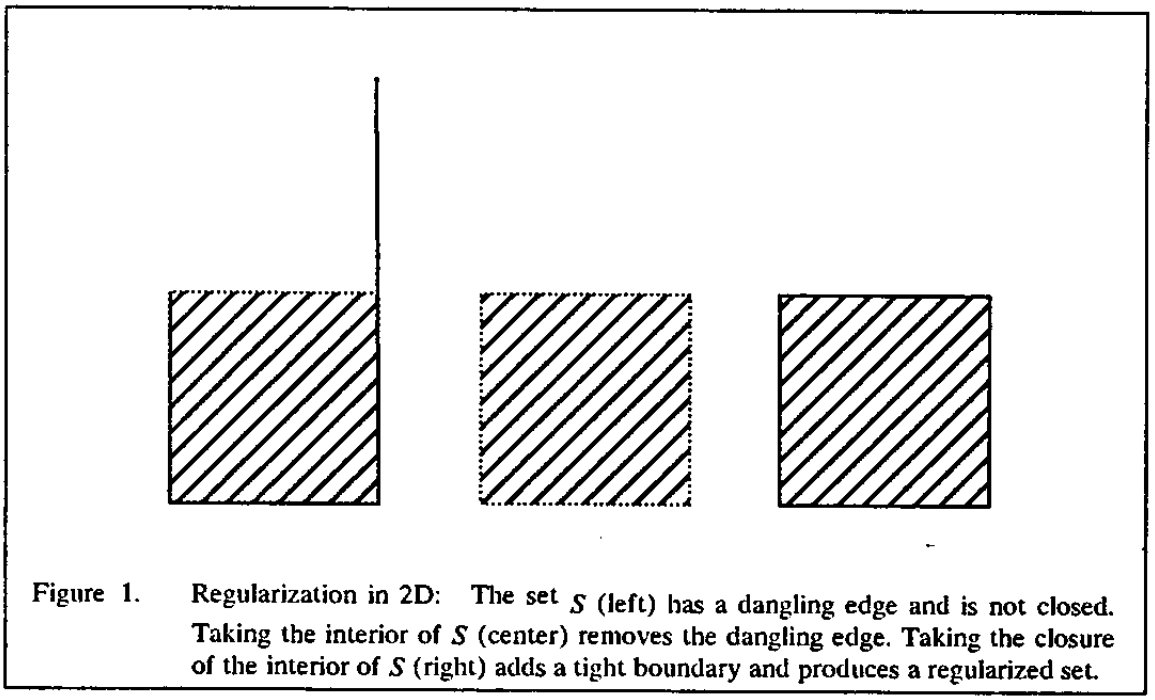

Figure 1. Regularization in 2D: The set S (left) has a dangling edge and is not closed. Taking the interior of S (center) removes the dangling edge. Taking the closure of the interior of S (right) adds a tight boundary and produces a regularized set.

A pointset is considered a "valid" solid if it is a closed, bounded, and homogeneously three-dimensional set [9]. Homogeneously three-dimensional sets are collections of 3D volumes that have no dangling edges, faces, or vertices. Because they are closed, valid solids include their boundaries, and thus have no lower-dimensional holes or cracks. Dimensional homogeneity is mathematically expressed by saying that the pointset is **regular**, i.e., that it is equal to the topological closure of its interior in $\mathbf{R}^3$ [26] (see Figure 1 for a 2D example).

In addition, for computational purposes, it is often required that the pointset be bounded, that it have finitely many disjoint parts, and that its boundary be "well behaved", that is, be composed of finitely many simple parts [26]. The class of semi-algebraic sets fulfill these computational requirements. They can be expressed as Boolean combinations of sets defined by algebraic inequalities, that is, of regions of space where a polynomial function of the three space coordinates has non-positive value. For example, a ball of radius R centered at the origin is a half-space defined by the algebraic inequality: $x^2 + y^2 + z^2 - R^2 \leq 0$. Consequently, the class of **bounded, regular, semi-algebraic sets** has often been associated with valid solid models and has been successfully used in solid modelling [27].

2.2 Boundary representations

A solid can be described in terms of its boundary by listing the types, positions, and dimensions of its boundary components. The boundary of a valid 3D solid is composed of a finite number of faces that are two dimensional subsets the surfaces of types supported by the modeller. Each individual modeller supports a fixed family of surfaces. Depending on the particular modeller, this family is typically composed of some subset of planes, quadrics, tori, and even rational polynomial parametric surface. A face is delimited from the rest of its supporting surface by its edges (one-dimensional curve subsets). Since trimmed curved faces are represented implicitly by their supporting surface and by their bounding edges, and since an edge bounds several faces, a boundary representation is often based on a graph whose nodes correspond to faces, edges, and vertices, and whose links specify face/edge and edge/vertex incidences [28, 29]. An edge may be represented explicitly in parametric form, or implicitly by its two endpoints and, if necessary, its supporting curve. A solid modeller based on boundary representation, then, is an implementation of data structures and algorithms which create and manipulate boundary representations.

2.3 Constructive Solid Geometry

Solids may be defined from "valid" boundaries constructed in surface modellers. Unfortunately, testing the validity of a boundary is expensive and complex. It involves geometric tests (for example verifying that the interiors of two different faces do not intersect) and topological tests (for example making sure that the faces are connected to form a series of closed shells that each separate space into two or more parts and that there are no dangling faces, edges, and vertices). Construction techniques that guarantee the validity of the graph of "boundary of" relations have been developed [28, 5], but directly creating or manipulating a boundary description is in general too tedious for humans, who prefer to design solids in a hierarchical and incremental manner (i.e., by combining or altering previously defined subsolids). Many mechanical parts can be precisely specified through Boolean set theoretic operations on simple primitive solids [30]. For example, combining two parts into a single solid can be modelled mathematically as a set theoretic union. Further shape modifications that produce specific geometric features (such as slots, holes, bosses...) [31] can often be formulated in terms of set theoretic unions and differences between the part to be modified and simple feature volumes to be added or subtracted [10]. A solid model may also be used to represent and analyze the geometric effects of certain manufacturing operations (for example material removal through milling and drilling, or material deposition in silicon devices [20]). In conclusion, solids that can represent a large family of parts and tools or can characterize the effect of a large class of manufacturing operations may be conveniently specified by combining subsolids or primitive volumes through set theoretic Boolean operations. Such specifications correspond to set-theoretic Boolean expressions, which can be parsed and easily and efficiently stored in a computer as a (binary) tree whose leaves represent primitive shapes (typically blocks, cylinders, spheres, cones or extrusions of 2D regions bounded by line segments and circular arcs) and whose nodes correspond to Boolean operations and potentially represent elaborate pointsets.

The Boolean **union** of two valid solids always produces a valid solid. The set theoretic **difference** between two valid solids may, however, produce a set that does not contain its entire boundary, and thus is not closed, as shown in Figure 2. Similarly, the set theoretic **intersection** of two valid solids may

produce a set that has dangling edges or faces, and so is not homogeneous in dimension, as shown in Figure 3. Consequently, Boolean combinations of valid solids do not necessarily produce valid solids. To guarantee that the output of one Boolean operation may be used as input to another operation, solid modellers use a modified version of Boolean operations, called regularized Booleans [32]. These perform the standard Boolean operations followed by a regularization operation which returns the closure of the interior of the resulting set, and thus eliminates all dangling faces, edges, and vertices, while including the entire boundary of the full dimensional part of the pointset. Such a representation for valid solids as regularized Boolean combinations of simple primitives is called a Constructive Solid Geometry representation, abbreviated CSG. The semantic distinction between regularized and non-regularized Booleans has strong practical implications on the algorithms that compute the boundaries or evaluate the properties of solids defined in CSG [33].

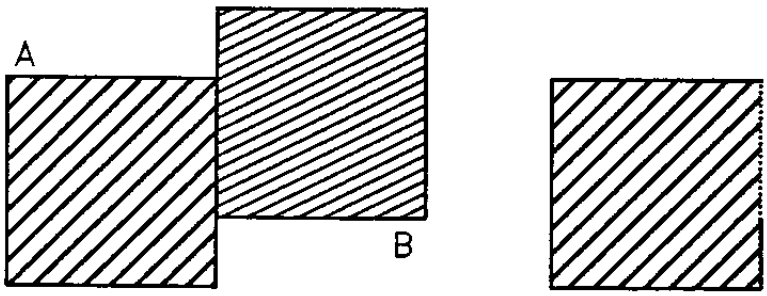

Figure 2. Non-closed set: The Boolean difference A – B of the two closed sets (left) is the non-closed set (right) obtained by subtracting from A a portion of its boundary.

Although many application algorithm operate directly on "unevaluated" CSG representations [25, 34], CSG representations provide very little explicit information about the associated solid and often must be "evaluated", or converted, into equivalent boundary representations. For example, deciding whether the solid defined by an elaborate CSG representation is connected or not, whether it has holes or handles, or even whether it is empty, are computationally expensive tasks [35, 24], which in general involve computing elements of the boundary [36].

2.4 Boundary evaluation

A complete CSG-to-Boundary conversion process, often called *boundary evaluation*, [37] may be organized in several ways [36]: either incrementally, by constructing the boundaries of the left and right operands before a Boolean operation is performed, or simultaneously, by considering the entire CSG representation and producing the boundary elements of the resulting solid all at once. Both schemes require calculating [38] and classifying [39, 29] a large number of geometric intersections.

Boundary evaluation—whether done directly from CSG or by combining boundary representations pairwise (as in some solid modellers based on boundary representations)—is based on the observation that the boundary of a Boolean combination, S, of two solids, A and B, is contained in the union of the boundaries of these solids. The bounding faces of S may thus be obtained by trimming [36] the faces of A and B to portions that lie on S and then merging such portions into "valid" faces of S [40].

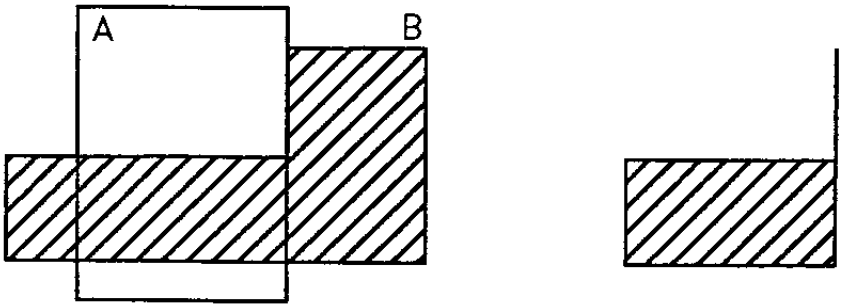

Figure 3. Non-homogeneous set: The Boolean intersection $A \cap B$ of the two homogeneously 2-D sets (left) is the non-homogeneous set (right) composed of a solid 2-D area and of a 1-D dangling edge.

Since faces are implicitly defined in terms of their edges, most boundary evaluation algorithms first compute tentative edges as intersections of all faces of A with all faces of B. These edges partition the faces into open subsets such that each face-subset is either entirely on the boundary of S or does not intersect it. The status, or classification, for a face-subset F of a face of A can often be simply derived by classifying F with respect to B, i.e., by determining whether F lies inside B or not. Furthermore, the status of F can often be propagated through certain face subsets of A that are incident to F without need for classifying them. When a face of A and a face of B overlap, the status of the common subsets of these faces must lie on the boundary resulting from a regularized Boolean operation between A and B, and thus depends not only on the nature of the Boolean operations, but also on the relative position (called face-neighborhood) of A and B with respect to the common face [39].

In practice, to eliminate the expense of constructing and then simplifying a partitioning for each face, most boundary evaluation algorithms partition and classify edges. To produce results consistent with the semantics of regularized operations, edge classification involves the evaluation of edge-neighborhoods [33].

Boundary evaluation algorithms that correctly perform regularized Boolean operations evaluate face- or edge-neighborhoods in order to eliminate from the boundary representation all internal or dangling faces and edges, i.e., all faces and edges that are not incident to both the interior of the resulting solid and the interior of its complement.

2.5 Extensions to solid modelling

Applications of geometric modelling to manufacturing automation require modelling non-solid geometric entities that describe the interaction of solids. For instance, the two-dimensional contact region between two solids models of electrodes could be of great interest for the analysis of some electromagnetic properties; the area of the contact region might determine the current flow. Such a region can easily be obtained as the set theoretic intersection of the two blocks, but cannot be obtained by regularized Boolean operations in three dimensions.

One might choose to consider the two blocks as two parts of the same solid and use that composite solid as input to further Boolean operations, such as drilling a hole through it. The separation between

the two electrodes cannot be represented as an internal face of the composite solid, because the solid would not be valid and because classification algorithms for regularized Boolean operations were not designed to handle such cases. Typically, they would eliminate the internal face, since it does not lie on the boundary of the resulting solid. The loss of information about the separation face can be avoided by considering the composite solid as an assembly and by distributing all further Boolean operations over the assembly operation [41]. However, such an approach results in the duplication of the common face-subset of the two electrodes and in the loss of geometric adjacency information. A relation linking the two electrodes and specifying whether they are touching or not could be established, but its status would depend on the positions of the electrodes and of the hole. To avoid these complications and support solids with internal separations, Arbab proposes to use s-sets, defined as unions of finite numbers of disjoint open-regular pointsets (pointsets that are equal to the interior of their topological closure). The set of such pointsets is closed under the standard intersection, but not under the standard union, since s-sets are restricted to be composed of connected cells whose boundaries are two-dimensional manifolds [41]. The set of open-regular pointsets is closed under Arbab's version of regularized Booleans, which return the interior of the closure of the results of standard Boolean operations, but destroy all internal boundaries.

We believe that a useful modeller should be able to represent and process the following geometric objects:

1. regularized 3D solids, which are necessary for many manufacturing applications,
2. non-closed solids, which are useful to study functions that are defined in the interior of a region but not on its boundary,
3. mixed dimensional collections of sets, which combine solids with "dangling" faces, and lines, and are useful in applications to VLSI analysis,
4. sets with missing interior points which might be used to model cracks or point loads,
5. sets with interior boundary points, such as the face separating the block from the table in our previous example, which are important for modelling solids composed of different parts,
6. sets of more that three dimensions, which are useful, among others, for configuration space analysis,
7. sets on any manifold, such as the clearance region of a robot, modelled on a sphere in terms of joint angle coordinates,

and both regularized and standard Boolean set operations on them, as well as other topological, structural, or user defined operations.

In the following sections we discuss a new framework, that of SGCs, and high level algorithms, which in a dimensionally independent way address all of these issues. When restricted to R^3, SGCs generalize Arbab's s-sets by providing support for pointsets with dangling internal and external boundary elements, while maintaining closure under all standard and regularized Boolean operations.

3.0 DEFINITION OF SELECTIVE GEOMETRIC COMPLEXES

This section introduces some fundamental concepts upon which SGCs are based.

3.1 Geometric elements in solid modelling

Although a face of a solid in R^3 is often thought of as a closed set, most algorithms used in solid modelling process face singularities (such as cusps, apices, and self-crossings) [42, 43] and bounding edges in a different manner than the regular (non-singular) interior subsets of the face. Similarly, a curve that has a singular self-crossing point is typically split for further processing into two or more edges that do not contain any singularity in their interior. As a consequence, singularities and bounding elements of a face or edge are often represented in a modeller and treated by the internal algorithms as separate lower dimensional elements, and not simply included in the face or edge. We

decided to exploit this separation in our work, and thus must formally define the fundamental geometric entities.

In practice often only semi-algebraic sets are considered in solid modelling, so that all faces, edges, and vertices of valid solids that may be represented in current modellers can be considered **connected full dimensional subsets of real algebraic varieties in R^3**. (A **real algebraic variety in R^n** is the locus of common real zeros of a finite set of real polynomials in **n** variables.) For simplicity we shall use the term "variety" to denote real algebraic varieties.

In this paper all varieties will be subsets of R^n and will be topologized with the relative topology with respect to R^n. A variety is always closed in R^n. A variety V, which is a subset of another variety W, is said to be a sub-variety of W, and if V is a proper subset of W then it is a proper sub-variety of W. By definition, intersections of varieties are also varieties, but it is also true that any finite union of varieties is a variety. A variety is called **reducible** when it can be expressed as the union of proper sub-varieties, and is called **irreducible** otherwise. Every variety can be uniquely decomposed into a union of irreducible varieties.

Varieties often contain points where certain "smoothness" properties vanish. Such points may require special attention in algorithms that process varieties. Their precise characterization will be described in detail in a separate report. Intuitively, the **singular points**, S, of a variety V include cusps, self crossings, and isolated lower-dimensional pieces of V. The set of singular points of a variety form a proper subvariety and so is closed. If V is an irreducible variety in R^n, then the regular points, R, of V can be defined as V − S. R is a smooth, non-empty, imbedded submanifold of R^n and has only finitely many **connected** components. We call each of these components an **extent** of V. Since S is closed, R and each of its **extents** is a relatively **open** subset of V. The dimension of a variety V in R^n, denoted V.**dimension**, is defined to be the dimension of R. Similarly, the singular points, S, of V can be decomposed as a finite union of connected, relatively open subsets of **extents** of lower dimensional varieties. Thus V itself can be decomposed as a disjoint union of a finite set M of connected smooth manifolds. In fact, M can be chosen such that for any manifold *M* in M, the boundary of *M* equals the union of some subset of M. Such a decomposition is called a manifold decomposition of V[1] . For example, in R^3, all points on a sphere are regular, and so the sphere coincides with its one extent, while a cone has one singular point, the apex, and decomposes into three extents: its apex and the two faces separated by this apex. When only simple (linear or quadratic) equations are used, the decomposition can always be found and is implicitly used in several modelling systems [50, 51, 43, 52, 53].

3.2 Cells

We propose a scheme for representing multi-dimensional geometric objects that is based on a fundamental entity called *cell*, which we define to be a **connected open subset of an extent.** By this definition, each face, edge, and vertex of 3D valid solids typically supported in current modellers is a cell. To each cell **c** we associate the unique irreducible variety, denoted **c.variety**, and the unique extent, denoted **c.extent**, that contain it as a relatively open subset. A cell, then, is defined and will be represented in terms of its **extent** and bounding cells[2].

Although, in some current modellers, the highest dimensional cells need not be connected, they are represented by lower-dimensional cells that are connected sets. For example, it is difficult to conceive of a polyhedral 3D modeller, where two or more disconnected line segments are considered a single edge. For uniformity in dimension, we enforce this requirement on all our cells. This choice leads to

---

[1] See [44] for the elementary properties of real algebraic varieties. The theory of stratifications [45, 46, 47] provides a geometric decomposition of a variety, which is more than sufficient for our purposes. A definition of imbedded submanifolds may be found in [48, 49].

[2] Parametric surface patches or 3-dimensional patches may be defined explicitly in terms of mappings, but trimmed subsets of them are often represented implicitly.

some complications during the subdivision process, but greatly simplifies the specification of the topological relation between cells.

3.3 Geometric complexes

Cells may share common boundaries, and therefore, to avoid duplications and provide a representation which carries incidence relations between a cell and its boundary, we shall use a structure, called a **geometric complex**, which for simplicity, we often will call a **complex**.

The topological boundary of a cell c, defined as the difference between the closure of c and c itself, will be denoted ∂c.

Definition 1: A **geometric complex** K is a finite collection of cells c_j, with $j \in J$, such that:

1. $i,j \in J$ and $i \neq j \Rightarrow c_i \cap c_j = o$
2. $\forall c \in K,\ \exists I \subset J \ni \partial c = \bigcup_{i \in I} c_i$
3. $\forall b \in c.\mathbf{boundary},\ (b \subset c.\mathbf{extent})$ or $(b \cap c.\mathbf{extent} = o)$,

where $\cap$ and $\cup$ denote the intersection and union of the pointsets that correspond to the appropriate cells.

It follows from this definition that the union of all the cells of a complex always represents a closed pointset. The real plane minus the integral lattice points is a valid cell, but has infinitely many connected components in its boundary, so that the finiteness of a complex implies that this cell may never be part of a complex. Thus membership in a complex imposes constraints on a cell.

Two operators on the cells of a complex will be essential in the sequel. For any cell c in a complex K, c.**boundary**(K), for brevity written c.**boundary**, denotes the collection of all cells c_i of K such that $c_i \subset \partial c$. The **star** operator maps a cell c to the pre-image of c under the **boundary** operator. Specifically, c.**star**(K), or c.**star** for short, is defined as the set of all cells of K that contain c in their boundary, that is, $\forall b \in K$, $b \in c.\mathbf{boundary} \iff c \in b.\mathbf{star}$[3]. Both operators **star** and **boundary** return collections of cells. For any cell c in the complex K, c.**boundary** is a valid complex of K, but c.**star** is not.

Because both the **star** and **boundary** operators define transitive relations between their operand and the collections they return ($v \in e.\mathbf{boundary}$ and $e \in f.\mathbf{boundary} \Rightarrow v \in f.\mathbf{boundary}$), storing all such relations in a representation of a complex may be redundant. Nevertheless, for simplicity of exposition we shall assume in our algorithms that all **boundary** and **star** relations are directly accessible[4].

The dimension of a cell c is denoted c.**dimension** and corresponds to the dimension of c.**extent**. The k-skeleton of a complex K is denoted K.**skeleton**(k) and refers to the collection of all cells of K that are of dimension less or equal to k. Given any collection X of cells in a complex K, it is easy to see that the collection $X \cup X.\mathbf{boundary}$ is a valid complex. It follows that K.**skeleton**(k) is a valid complex. Let K.**cells**(k) be the set of all cells of dimension k in K[5].

Two complexes, A and B, are said to be **equal** if they have the same collection of cells, i.e. for any cell a in A, there is a cell b in B such that $a = b$ and vice versa. However, two complexes with equal pointsets need not be equal.

---

3 The notation "$c \in X$" stands for "the cell c is part of the collection X of cells".

4 This pertains to implementation decisions. For example, should a 2D face contain direct references to all its bounding vertices, or should the vertices that bound its edges be accessed through the edges?

5 Note that K.**skeleton**(k) $\neq$ K.**cells**(k) + K.**cells**(k).**boundary**, because K.**skeleton**(k) may contain lower dimensional cells that are not bounding any cells of dimension k.

Two complexes A and B are **compatible**, if $\forall a \in A, \forall b \in B, a \cap b \neq o \Rightarrow a = b$. A complex A is a **refinement** of a complex B, if each cell of B can be written as a union of cells of A, i.e, if the pointsets of A and B are equal and if $\forall a \in A, \exists b \in B, \ni a \subset b$. Our definition implies that any complex is a refinement of itself. If A is a refinement of B and A is not equal to B, then A is said to be a **proper refinement** of B.

3.4 Neighborhood

It has been recognized [54,42] that even in standard modelling systems knowledge of the **extent** and **boundary** of a cell is not always sufficient to define the cell. For example, two complementary arcs of a circle share the same **extent** (the circle) and the same **boundary** (their two common bounding vertices). Sometimes (implicit) global considerations can be used to provide enough additional information to complete the definition. For example, the boundaries of "valid", and thus bounded, solids can be used to define three dimensional regions in space in terms of collections of faces. However, in general, to complete the definitions of cells and to ease algorithmic development, the concept of **neighborhood** of a lower-dimensional cell in a higher-dimensional one is usually employed [33]. Thus, in our example of the circular arcs, an arbitrary choice of orientation of the circle is made (often derived from an explicit parametrization), and then a vertex is said to bound and arc on the right (respectively left) if it follows (respectively precedes) the arc along the chosen orientation. Similarly, an edge and a face it bounds can be arbitrarily assigned orientations to make sense of the edge bounding the face on the right (or on the left). In these sections, we generalize these notions to geometric complexes. We will restrict the use of neighborhood information to the case where the bounding element is of dimension one less than the cell it bounds. In these situations neighborhoods are simplest to define and process, and they suffice for the algorithms that follow.

A cell **b** of dimension k is said to be a **regular boundary cell** of a cell **c** in **b.star** if **c** is of dimension k+1 and if **b** is contained in **c.extent**. If **b** is a regular boundary cell of **c**, then the **neighborhood** of **b** with respect to **c**, is a property of the pair (b,c), and can take one of the following three values: **left**, **right**, or **full**. For convenience of notation, it will be defined as a function, denoted **b.neighborhood(c)**, associated with **b** and having parameter **c**. **b.neighborhood(c)** describes the topological relation between **b** and **c** in **c.extent**. If **b** does not belong to the **c.extent**, then **b.neighborhood(c)** is not defined, and **b** is called a **singular boundary cell** of **c**.

b.neighborhood(c)=full indicates that **b** is not in the boundary of the topological closure of **c**, in other words, that $b \cup c$ may be considered as a valid cell. We shall refer to such situations by saying that **b** is an "interior boundary" cell of **c**. Alternatively, one may say that **b** is an interior boundary cell of **c** if **b** is not contained in **c** but is contained in the interior of the closure of **c**. On the other hand, **b.neighborhood(c)** $\neq$ **full** indicates that **b** is in the boundary of the topological closure of **c**, relative to the topology of **c.extent**. We refer to such situations by saying that **b** is a "exterior boundary" cell of **c**. Alternatively, one may say that **b** is an exterior boundary cell of **c** if **b** is not contained in **c** and is not contained in the interior of the closure of **c**.

When the **neighborhood** of **b** with respect to **c** is not **full**, it must be **left** or **right**, and denotes the "side" on which **c** is located with respect to **b** [42]. To define "side", we assume that there is a function D, that is continuous throughout **b** and which to each point P of **b** associates a direction D(P) in **c.extent**, such that D(P) is orthogonal to **b** at P. The existence of such a function imposes constraints on the orientability of **b** and **c**[6]. In terms of this function D, we define **b.neighborhood(c) = left** to mean that for each P in **b** there is a curve C, beginning at P and totally contained in $c \cup P$, whose tangent direction at P is coincident with D(P). Similarly, **b.neighborhood(c) = right** means that for each P in **b** there is a curve C, beginning at P and totally contained in $c \cup P$, whose tangent direction at P is coincident with -D(P).

[6] A precise formulation and discussion of these constraints is deferred to a separate report.

Strictly in terms of the goal of providing sufficient information to be assured that a cell is well defined, much less is needed in addition to the **boundary** and **extent** of a cell than **neighborhood** information for each regular boundary cell. In fact, sometimes no more information is necessary at all. For example, two points in a line or two disjoint circles in a plane are the boundary of a unique cell in the line or plane. In general, the **boundary** of a cell can be seen to be the boundary of at most two cells in an **extent**, and these two cells are disjoint. Thus any means for distinguishing between these two cells (for example the **neighborhood** of one exterior boundary cell, the recognition of an internal boundary cell, or the knowledge that some point is in the cell) is enough to provide a well founded definition.

Even if the **neighborhoods** of all regular boundary cells are required for algorithmic efficiency, they can often be derived from partial **neighborhood** information or from global considerations. For example, if the **neighborhood** of a boundary cell **b** of a cell **c** is known, this information can be propagated to the connected component of **c.boundary** in **c.extent** that contains **b**. For "valid"[7] solids in R^n more can be said. Let F be a finite set of mutually disjoint, well defined (in some manner), bounded (n-1)-dimensional face-cells that form the boundary of a valid solid S in R^n. The complement of S is composed of finitely many connected disjoint regions of R^n. Since valid solids are bounded, there can only be one unbounded region, and that region must lie in the complement of S. Consequently, the other regions may be classified as belonging to the represented solid or not by simply alternating this classification each time a boundary face is traversed. This property has been used in certain algorithms to classify points against solids represented in terms of their boundaries by generating a half-line originating at the tested point and counting the parity of the number of times the half-line intersects the boundary (the point is out if that number is even, and in other wise). This half-ray approach can also be used to establish the neighborhood of a face **b** with respect to a solid **c** at some interior point P of **b**. For that, one simply constructs an open half-line, L, orthogonal to **b** at P and leaving **b** in a direction consistent with D, as defined above[8]. If L intersects the boundary of **c** an even number of times, then **b.neighborhood**(**c**) is equal to **left**, and otherwise to **right**.

In practice, the repeated propagation of partial **neighborhood** data can be costly, while the ray-casting technique is difficult to extend to the full generality of SGCs. The existence of internal boundaries makes the count of inclusion-changing intersections problematical, and the existence of unbounded components makes the simple parity check inapplicable. Thus it seems more economical to store all available **neighborhood** information for reuse. It is reasonable to assume that the **neighborhood** information for the primitive shapes supported by the system are known completely, as is the case in current modellers. Moreover, the algorithms for manipulating SGCs operate on and return SGCs with complete **neighborhood** information. Thus, for simplicity, we assume that **bneighborhood**(**c**) is always directly available for all cells **c** with regular boundary **b**.

3.5 Selective geometric complex

As we have seen, the union of the pointsets of the cells of a complex is always a closed set. To provide the capability for representing non-closed objects and controlling the structural decomposition of the represented pointset, we introduce the notion of a **Selective Geometric Complex**, abbreviated SGC.

A Selective Geometric Complex, O, is composed of a complex, denoted O.**complex** and of an extendible set of attributes attached to each cell (or group of cells).

---

[7] A "valid" solid in R^n is a closed, bounded, homogeneously n-dimensional set with "nice" boundary [26]..

[8] Since **c.dimension** = **b.dimension** + 1, there are always two possible choices for the direction of D at each point of **b**, and in fact the choice at a single point determines D at all points of **b**. We assume, for simplicity, that this choice can be derived consistently from a description of **c.extent** and **b.extent**.

In particular, the **active** attribute specifies whether the pointset defined by the cell should be included in the pointset defined by the SGC. Consequently, the pointset associated with an SGC, O, is denoted O.**pointset** and is defined as the union of all pointsets of the cells c of O.**complex** such that c.**active**=TRUE. Thus, the **active** attributes enables an SGC to represent a non-closed pointset (i.e., a pointset with cracks or with an incomplete external boundary).

Various attributes, other than **active**, may be defined incrementally by the user, and can play important roles for manipulating and interrogating SGCs, as will be demonstrated later. For example, a structural attribute can be used to preserve during merging and simplification operations the internal structure of a pointset, such as the faces of a finite element decomposition.

3.6 Comparison of SGCs with other complexes

In the foregoing, we have stressed the similarity of cells of a complex to the vertices, edges, and faces currently used in geometric modelling in order to present cells as natural generalizations of the standard boundary elements. While we hope that the formal structure of complexes as a collection of mutually disjoint cells with compatible boundaries composed of unions of other cells and even the **boundary** and **star** operators will not seem altogether dissimilar to the usual data structures for geometric modelling, they in fact follow much more closely the practice in common complexes used in algebraic topology. In fact, the definition of a complex, the two major operators (**boundary** and **star**), and even the terminology would be almost identical if we were considering simplicial or CW complexes [55] rather than geometric complexes. The differences lie almost exclusively in the concept of what constitutes the fundamental entity: the cell. That the concept of a cell would differ is far from surprising, in as much as the cells of simplicial and CW complexes were developed to attack topological questions, whereas the cells developed here have been tailored to the needs of geometric modelling.

Each cell of a simplicial complex is homeomorphic to an n-simplex (an n-dimensional triangle) for some n, while each cell of a CW complex is homeomorphic to an n-ball for some n. Homeomorphisms allow great simplification of structure. For example, any valid, n-dimensional, homotopically simple solid with connected interior and arbitrarily many boundary elements can be represented as a single n-simplex or as a CW complex composed of three cells: a point in the solid's boundary, the rest of the boundary, and the interior. But, if the purpose is geometric modelling, either of these representations is unacceptable, for the very reason of its simplicity. All the boundary element information has been lost, swallowed up in the interior of faces. Morover, the homeomorphisms in general lead far outside the realm of maps that can be represented, evaluated, or manipulated within a computer. If we make the reasonable restriction that boundary data become explicitly available and that the maps remain within the scope of the varieties supported in a modeller, the situation changes. First, the representations become more complex. For example, a cube in R^3 has a minimal representation as a simplicial complex with 5 3-simplices, 16 faces, and 22 edges; a line must be decomposed into infinitely many line segments to represent it as a CW complex. Moreover, arbitrary choices become unavoidable, so that there is no chance to expect uniqueness in representation. In contrast, the cells of SGCs have been designed to be the representation requiring minimum fragmentation consistent with explicit presence of boundary data and with connectedness of cells. Thus an SGC representation of the cube has 1 volume, 6 faces, 12 edges, and 8 vertices; and the line is a single one dimensional cell. Later we describe algorithms for obtaining a unique SGC representation for pointsets.

4.0 DATA STRUCTURE

In this section, we describe a possible organization of a representation scheme for SGCs. We discuss what information should be available to provide a complete and practical representation, but we do not discuss techniques for minimizing storage and access time. Specifically, for convenience, we tolerate a certain amount of redundancy in the graph representation of SGCs by assuming several data paths from one cell to cells in its **boundary**. We also treat the references from a cell to its **boundary** and to its **star** as unordered sets, although they can be implemented using ordered linked lists, hash tables for efficiency, or any other convenient data structures. We also implicitly assume that all cells of a given dimension may be accessed directly, which can be achieved using a variety of standard data structures. Furthermore, to improve the efficiency of algorithms that merge SGCs by eliminating the redundancy of certain geometric computations, cells should also be grouped in terms of the hierarchy of varieties and sub-varieties to which they belong. Consequently, cells of SGCs can be accessed in three ways:

1. using **boundary** and **star** links between cells,
2. using sets of references to all cells of a given dimension,
3. using sets of references to all cells in a given **variety** or **extent**.

Only the first graph structure will be discussed in this paper.

4.1 Incidence graph for complexes

The structure of a geometric complex extends the vertex/edge/face graph [28], commonly used for modelling boundaries of solids, in two ways: (1) to arbitrary high dimensions, and (2) to non-homogeneous structures. Diverse aspects of extensions of the vertex/edge/face graph have been independently addressed in the CAD/CAM research community [54, 56, 57]. The closest to our work is Kevin Weiler's radial edge structure [57, 58], which also can be used to represent "non-manifolds", i.e., objects with dangling and interior edges and faces. Weiler's structure is—to the best of our knowledge—limited to three dimensions and currently restricted to linear cells (polyhedra, polygons, line segments, and points), although an extension to curved surfaces is currently being studied. Weiler's representation has been engineered to improve the incidence graph traversal time by organizing edges into loops around faces and ordering faces around their common edges. Such explicit ordering information could be added on top of the SGC representation scheme, but efficiently organizing this information for non-linear structures (where for example an edge belonging to the external boundary of a face bounds it on two or more sides) in higher dimension [59] is still a research issue. Therefore, for uniformity across dimensions and also to simplify our merging algorithms, we assume a very simple data structure, in which there is no spatial ordering of the **star** or **boundary** links.

To each cell **c** of an SGC we associate the following information:

c.extent, which is a reference to the **extent** in which the cell **c** lives, and thus indirectly to the **variety** of this **extent**.

c.dimension, which is the dimension of **c** (and of **c.extent**).

c.boundary, which is an unordered list of references to cells of the same SGC that are bounding **c**. It is convenient, but not essential, to group these references by the dimension of the sets they refer to.

c.star, which is an unordered list of references to cells of the same SGC bounded by **c**. In addition, whenever for a cell **b** in **c.star**, **c.neighborhood(b)** is defined, then the value of **c.neighborhood(b)** is attached[9] to the reference for **b**. It is also convenient, but not essential, to group these references by the dimension of the sets they refer to.

[9] **c.neighborhood(b)** could also be attached to references from **b** to **c** in **b.boundary(c)**.

c.active, which is an attribute whose value indicates whether the pointset of c should be considered as part of the pointset of the SGC.

c.Attributes, which is a collection (list) of name/value pairs defined by users or by application programs.

To illustrate the incidence graphs, let us consider two simple examples of SGCs in two dimensions. The object in Figure 4 is composed of two faces, seven edges, and six vertices. Arrows on the edges indicate some arbitrary orientation of their supporting **extents**. The common **extent** of the faces (the plane of this paper) is also oriented. Edges E1 and E3 bound no faces, while edge E2 bounds two faces, F1 and F2, of the same **extent**. Vertices V1, V3, and V4 each bound four edges, while vertices V2 and V5 bound only one edge each. The vertex V6 bounds no edge, but instead is an internal vertex of face F2 (probably used to distinguish the center of the outer circle). The incidence graph of for the geometric complex of the object in Figure 4 is shown in Figure 5 . The dotted lines, which indicate the bidirectional **boundary/star** incidence between faces and vertices, are redundant except for the link between F1 and V6.
For this particular geometric complex, explicit neighborhood information is redundant, since it could be automatically derived from the **boundary/star** links in the incidence graph and from the geometric description of the **extent** of each cell.

Considering only local information, E4 cannot be distinguished from E5 since they have the same **boundary** and the same **extent**. Similarly, E6 cannot be distinguished from E7. On the other hand, edges E1, E2, and E3 have all different boundaries, and thus can be distinguished. The two circular **extents** and the three edges, E1, E2, and E3 divide the plane into four regions, two of which are F1 and F2. The geometric location of F1, F2, E4, E5, E6, and E7 can be obtained as follows:

1. F1 is the region containing V6.
2. F2 is the region bounded by E2 and not containing V6.
3. E6 is the portion of the large circle that bounds F1.
4. E7 is the portion of the large circle that bounds F2.
5. E4 is the portion of the small circle that bounds F1.
6. E5 is the portion of the small circle that bounds F2.

From these definitions and the orientations of the **extents**, the face/edge and edge/vertex **neighborhoods** can be derived if necessary. We shall nevertheless assume that **neighborhood** information is readily available when well defined, and our merging and simplifying algorithms maintain that information.

The graph for Figure 5 can be easily implemented using sets, symbolized by the brackets "<" and ">" in the printout below. (We have omitted the **active** and other attributes.) The **boundary** and **star** sets are grouped into subsets of constant dimension, symbolized by interior brackets. The **neighborhood** information is encoded together with the elements of **star** using L, R, and F symbols and grouping parentheses. As pointed out earlier, **neighborhood** could instead be associated with **boundary** links, or with both **star** and **boundary** links.

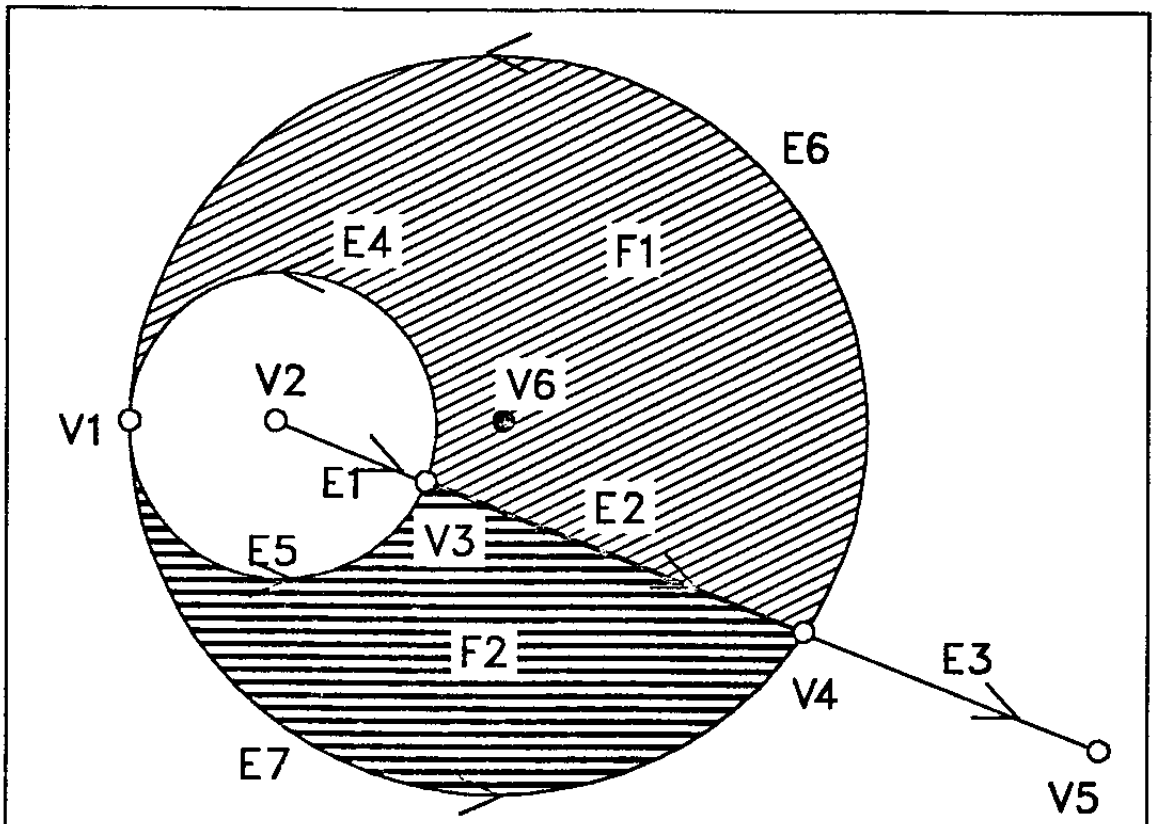

Figure 4. A complex: – represented by 2 faces, 7 edges, and 6 vertices.

```
FACES
  F1: ext=plane, bdry=<<E2,E6,E4>>,<V1,V3,V4>>
  F2: ext=plane, bdry=<<E7,E2,E5,>,<V1,V4,V3,V6>>
EDGES
  E1: ext=line, bdry=<<V2,V3>>
  E2: ext=line, bdry=<<V3,V4>>, star=<(F1,L),(F2,R)>
  E3: ext=line, bdry=<<V4,V5>>
  E4: ext=cir1, bdry=<<V3,V1>>, star=<(F1,R)>
  E5: ext=cir1, bdry=<<V1,V3>>, star=<(F2,R)>
  E6: ext=cir2, bdry=<<V4,V1>>, star=<(F1,L)>
  E7: ext=cir2, bdry=<<V1,V4>>, star=<(F2,L)>
VERTICES
  V1: ext=pt1, star=<<(E4,L),(E5,R),(E6,L),(E7,R)>,<F1,F2>>
  V2: ext=pt2, star=<(E1,R)>
  V3: ext=pt3, star=<<(E1,R),(E2,L),(E4,L),(E5,R)>,<F1,F2>>
  V4: ext=pt4, star=<<(E2,R),(E3,L),(E6,L),(E7,R)>,<F1,F2>>
  V5: ext=pt5, star=<(E3,R)>
  V6: ext=pt6, star=<<>,<F2>>
```

Different configuration may be found in the geometric complex of Figure 6, which is composed of one face, five edges, and four vertices. The edge E6 is on the left and on the right of vertex V4 and therefore V4.**neighborhood**(E6)=**full**. Similarly, E2.**neighborhood**(F1)=**full**. On the other hand, vertex V6 is not connected to any edge or face, and thus its **star** is empty. Here, however, no global geometric information may be used to distinguish E4 from E5, since they have the same **extent**, **boundary**, and **star**, and thus **neighborhoods** are needed.

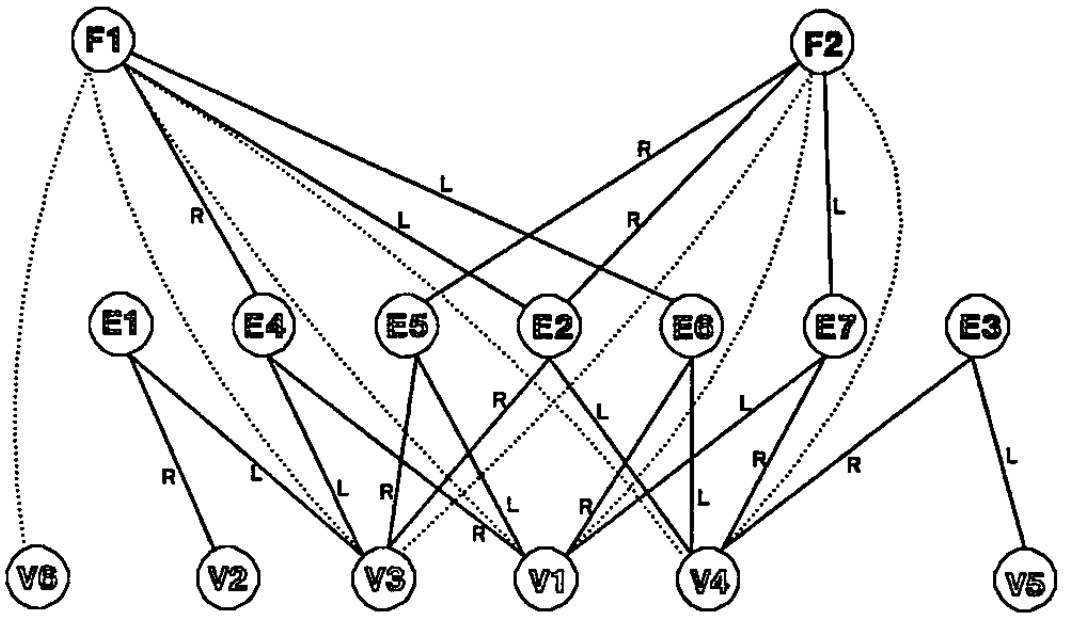

Figure 5. The graph of a complex: solid lines define **boundary/star** links from one dimension to the next. L and R annotations on these lines indicate LEFT and RIGHT neighborhood relations. Dashed lines indicate **boundary-star** links that jump more than one dimension and do not carry any neighborhood information.

The printout for the incidence graph of Figure 6 is presented below:

```
FACES
  F1: ext=plane, bdry=<<E2,E6,E4,E5>,<V3,V4,V5>>
EDGES
  E2: ext=line, bdry=<V3,V4>, star=<(F1,F)>
  E3: ext=line, bdry=<V4>
  E4: ext=circ1, bdry=<V3,V5>, star=<(F1,R)>
  E5: ext=circ1, bdry=<V3,V5>, star=<(F1,R)>
  E6: ext=circ2, bdry=<V4>, star=<(F1,L)>
VERTICES
  V3: ext=pt3, star=<<(E2,R),(E4,R),(E5,L)>,<F1>>
  V4: ext=pt4, star=<<(E2,L),(E3,F)>,<F1>>
  V3: ext=pt5, star=<<(E4,L),(E5,R)>,<F1>>
  V6: ext=pt6
```

4.2 Geometric representation of extents and varieties

We have assumed that to each cell is associated an **extent**. The graph structure and algorithms for SGCs are independent of the way in which an **extent** is represented, provided that its intersections with other **extents** (or rather the intersections of the associated varieties) can be computed and that relative **neighborhoods** can be derived from the orientations of the **extents**. Thus the choice of representation for **extents** and varieties will be dictated by the desired geometric coverage, the concern for the reliability of geometric calculations, and the tools available for computing intersections of varieties and for decomposing them into **extents**.

For example, if varieties corresponding to systems of linear equations are the only varieties supported in an SGC-based modelling system, then these equations are sufficient to **implicitly** represent the varieties and also the **extents**, since each **extent** is equal to its **variety**. Certain conventions can be

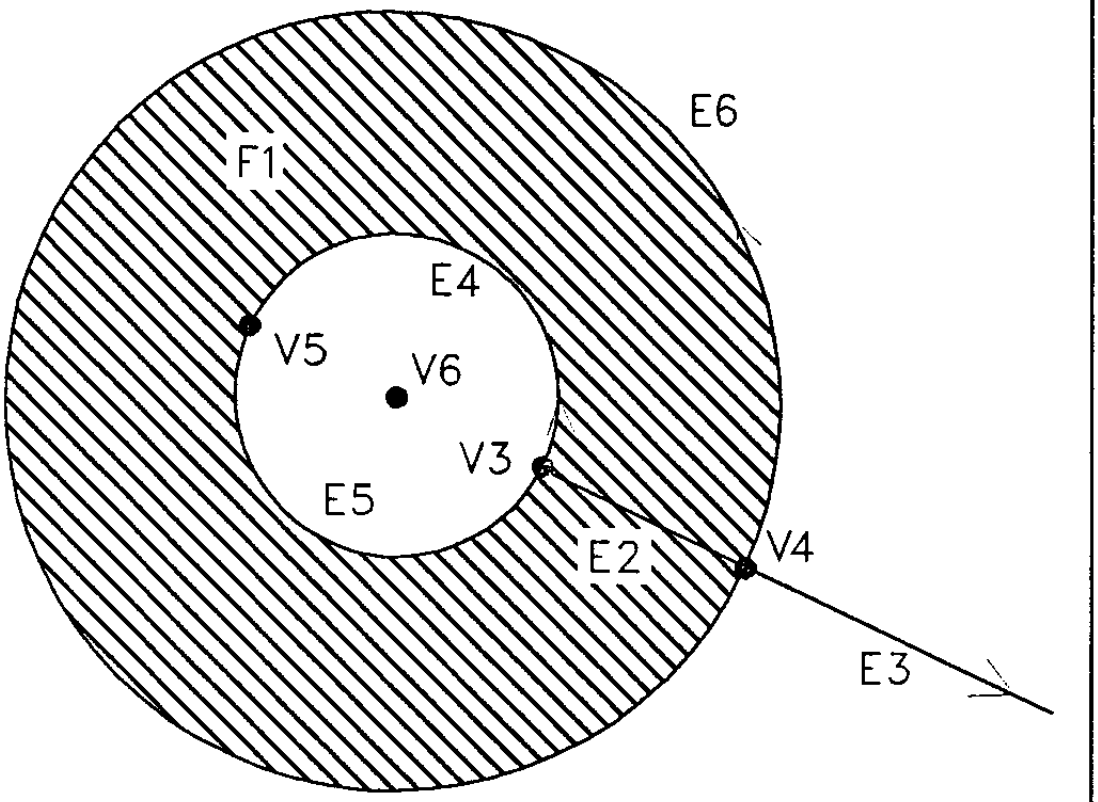

Figure 6. A complex: represented by 1 face, 5 edges (one of which is unbounded), and 4 vertices.

adopted to represent the orientation of each **extent**. On the other hand, **parametric** representations may prove more efficient for intersection calculations (for example, a line in $\mathbf{R}^3$ may be represented by a point and a direction) provided that these parametric representations are exact (for example extended precision rational coefficients or that the consequences of approximating them (for example with floating point coefficients) are acceptable.

For higher degree varieties one needs to store sufficient information to distinguish between the different **extents** of a **variety**. For example, the intersection of two cylinders may be composed of two (disjoint) curves that, by definition, have the same implicit representation. Thus, to distinguish between the two **extents**, the implicit representation can be augmented with an inequality that corresponds to some half-space that contains one curve, but is disjoint from the other. On the other hand, if a separate parametric representation is available for each curve, it can be used advantageously [60]. Unfortunately, such parametric representations can rarely be expressed by polynomials with rational coefficients, so that calculation and processing of their intersections with other surfaces may be difficult. An approach to this problem proposed in [61] uses symbol manipulation techniques to compute exact answers to the geometric queries on the relative position of simple curved surfaces in $\mathbf{R}^3$.

Many varieties can be conveniently represented **explicitly** by a type (for example circle or cylinder) and a few parameters (for example, the values of the radii of the circle or the cylinder). Such explicit representations are convenient, but typically require special case treatment by algorithms that processes the variety (for example, for displaying or computing its intersections with other varieties) [51].

All three representations (implicit, parametric, and explicit) are available for many of the varieties used in existing modellers (for example natural quadrics). Typically, the explicit representation is used for designers input and storage. The parametric representation of curves is important for display purposes. The implicit representation of surfaces is often used for computing intersections of these surfaces with parametrically represented curves [60, 50].

5.0 HIGH-LEVEL ALGORITHMS

Topological operations (closure, interior, boundary), Boolean operations (union, intersection, and difference) and their regularized counterpart, and in fact many other structural or special purpose operations (k-skeleton) are extremely useful for creating, combining, and interrogating geometric models of parts and manufacturing processes. The semantics of these operations can be precisely defined for SGCs and SGCs form a closed set under them. While developing algorithms for manipulating SGCs, it became clear that most of the topological, Boolean, and structural operations may be simply constructed as a sequence that involves one or more of the following primitive operations on SGCs:

subdivision, which makes the geometric complexes of two SGCs *compatible* with each other by refining them, i.e., by subdividing their cells,

selection, which selects cells of one or more compatible geometric complexes and sets their **active** attributes according to some selection criteria, and

simplification, which by deleting or merging certain cells produces a simpler SGC that represents the same pointset while still maintaining desired internal structures.

5.1 Compatible subdivision of geometric complexes

Given two geometric complexes A and B, the goal of the subdivision procedure is to compute two objects A' and B' such that:

- A' is a refinement of A,
- B' is a refinement of B, and
- A' and B' are compatible.

The compatibility is achieved by splitting cells of A and of B at their intersections while maintaining a valid structure for both complexes. For example, let **a** be a cell of the complex A and **b** a cell of the complex B. Sometimes the intersection $\mathbf{c} = \mathbf{a} \cap \mathbf{b}$ cannot be represented as a single cell, but it can always be represented as a collection of disjoint cells $\mathbf{c}_j$. Adding the $\mathbf{c}_j$'s to the boundary of **a** (which may involve splitting **a**) replaces **a** in A by a collection of disjoint cells $\mathbf{a}_i$ (of same or lower dimension) such that $\mathbf{a} = \cup \mathbf{a}_i$. The attributes of **a** are inherited by all $\mathbf{a}_i$. A similar replacement is performed for the cell **b** in B.

The subdivision algorithm is composed of an outside loop which starts with dimension 0 and goes up to the highest dimension of both geometric complexes. For each value k of this dimension, our algorithm first makes the k-cells of A compatible with the (k-1)-skeleton of B, then symmetrically makes the k-cells of B compatible with the (k-1)-skeleton of A. (Each one of these operations involves, in turn, another loop that traverses all cells of one of the geometric complexes by increasing dimension.) Finally, cells of dimension k in both A and B are subdivided at their common intersections. At each stage, both A and B are refined, but remain valid SGCs, which is the principal advantage of this bottom-up approach and greatly simplifies the algorithm.

```
subdivide: SUBR(A,B)
  FOR k=0 TO MAX(A.dimension,B.dimension) DO
    BEGIN
    subdivide A with B.cells(k-1);
    subdivide B with A.cells(k-1);
    subdivide A and B with the intersections of their k-cells;
    END;
  RETURN(<A,B>);
  END;
```

Essentially our algorithms add to A and B the intersection of their cells. The bottom-up approach ensures that when a new intersection cell is added, its boundary is already in place. For example, let a cell **a** of A partly overlap a cell **b** of B of the same dimension m. When the dimension k in the above algorithm reaches m, **a** is first split by the boundary of **b**, then **b** is split by the boundary of **a**. The third step in the loop has only to recognize which portion of **a** and **b** overlap. This approach is similar to the Euler operators [28]. The details of this algorithm and the discussion of its correctness will be presented in a separate report.

The action of the subdivision algorithms is illustrated in Figure 7, where two SGCs A (top left) and B (top right) are made compatible by the the following subdivision steps: Second line: a 0D cell (vertex) of A that intersects a 1D cell (edge) of B is added to the structure of B. Note that there was no 0D cells in B and therefore A is unchanged. Third line: the two points of intersections between 1D cells of A and of 1D cells of B are computed and added to both A and B. Note that these additions disconnect the other 1D cells. At this point the 1-skeletons of A and B are compatible, and A and B are valid objects. Bottom line: subdividing 2D cells of A with 0D cells of B and vice versa has no effect. However, subdividing the 2D cell of A with 1D cells of B results in the addition to the structure of A of two arcs of B, which disconnect the 2D cell of A into three 2D cells and two 1D cells. Note that the boundaries (vertices) of the two arcs added to A are already in place. Similarly, subdividing the 2D cell of B with 1D cells of A also refines B. Finally, there is no lower dimensional intersection of 2D cells of A with 2D cells of B.

5.2 Selection of cells

The selection operation may be applied to a single SGC, in which case it may simply change the **active** attributes of cells that satisfy certain conditions. Selection may also be applied to several compatible SGCs. In that case, selection first constructs a single valid SGC from the appropriate cells of all the input SGCs. A new incidence graphs is constructed by merging relevant parts of the incidence graphs of the input SGCs in an incremental fashion. Finally, the **active** attributes of cells of the new SGC are set.

This merging operation may be performed in various ways. The simplest, although not necessarily the most efficient, approach considers two SGCs, A and B, at a time and "inserts" in C (a copy of A) all cells of B that are not already cells of A. The incidence graph obtained in this manner represents a valid geometric complex. Each cell, **c**, of C corresponds to a single cell **a** of A and/or a single cell **b** of B. To each attribute T of **c** (including the **active** attribute) is temporarily associated a pair of values. The first of these values corresponds to the value of T for **a** in A or is set to NIL if there was no cell in A corresponding to **c** or if **a** did not have a T attribute. The second is derived in a similar manner with respect to **b**. These pairs of "inherited" attributes may be used to store the history (provenance) of each cell and also to select cells based on the role they played in A and B. This history may be used to produce pointsets that correspond to Boolean expressions involving A and B, which in turn may be obtained by merging other SGCs. Note that all Boolean combinations of a collections of mutually compatible SGCs can be obtained without any geometric calculation by simply evaluating Boolean expressions stored in the history of each cell [62].

To determine whether a cell **b** of B is a cell of A it suffices to classify any interior point of **b** against all cells of A: if a cell, **a**, of A is found that contains the point, then **b** is equal to **a** (since the complexes

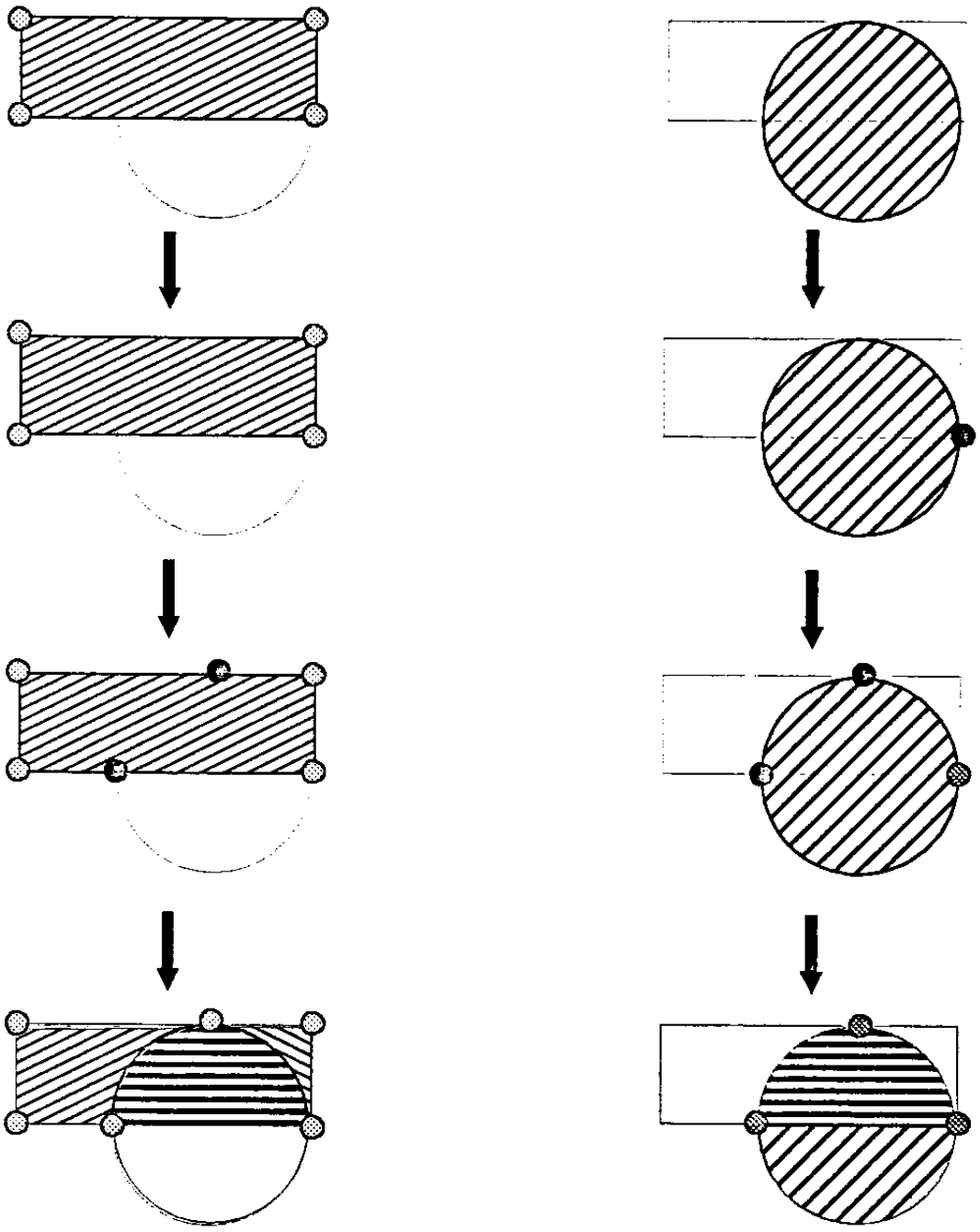

Figure 7. Subdividing two objects: Two objects A (top left) and B (top right) are made compatible by subdivision. For reference, B is shown on the left column and A on the right column using thin lines.

of A and B are compatible), otherwise $\mathfrak{b}$ is not a cell of A[10]. On the other hand, the subdivision algorithms involves identifying identical cells in both A and B, and this information can be preserved

[10] This simple test is sufficient because cells are required to be connected.

by setting "same-as" relations between cells of different SGC, or by representing these duplicate cells only once as nodes shared by the incidence graphs of both SGCs.

To simplify cell insertion, the merging algorithms simply adds cells of B to C, and also proceeds bottom-up, starting with cells of the lowest dimension in B, so that, at each stage, C is a valid SGC and that the boundary of any cell to be inserted is already in place. The insertion of a cell **b** into C given **b.boundary** in C involves:

1. adding **b** in the lists of the cells of the appropriate dimension in C,
2. adding **b.extent** to the list of extents in C (if it is not already there) and adding **b** to the list of cells associated with **b.extent**,
3. setting the bi-directional links of **b.boundary**,
4. setting, when appropriate, the **neighborhoods** for these links, and
5. inserting a reference to **b** in the **star** and **boundary** lists of all cells in **b.boundary**.

Cell selection computes the value of the **active** attribute for each cell **c**, according to Boolean expressions, or predicates, that may involve:

- the dimension of **c**,
- the two values of **c.active** inherited from cells of A and B that contain **c**,
- the presence of cells in **c.boundary** or in **c.star**,
- the inherited values of other attributes, as well as,
- various geometric or topological properties.

The following examples illustrate the power of selection. The Boolean union, A ∪ B, of two SGCs may be obtained by "activating" (setting to TRUE the **active** attributes of) all cells that were "active" (whose **active** attribute was TRUE) in A or in B. The difference, A − B, may be obtained by activating cells that were active in A but were not active or not contained in B. The closure of an SGC is obtained by activating all cells whose **star** contains an active cell. (This is efficiently done by visiting all cells of the SGC and by activating the **boundary** of each active cell. A marker may be used to prevent processing the same cells more than once.) A regularized difference may be obtained by performing the selection for "difference" followed by the selection for "closure".

5.3 Simplification

Given an SGC, A, the objective of simplification is to find an SGC, A', that represents the same pointset and has the same structural information, but has a geometric complex with as few cells as possible. (The complex of A is a refinement of the complex of A'.)

We introduce three primitive operations which perform valid and effective simplifications on particular cells while preserving the validity of the complex:

DROP	removes an inactive cells,
JOIN	removes an external boundary cell between two cells of the same extends and thus merges these two cells into one,
INCORPORATE	removes a interior boundary cell by merging it with a unique cell that it bounds.

A simplification step can only take place if it

- preserves the validity of the geometric complex,
- preserves the represented pointset, and
- does not produce a loss of structural information.

Simplifying operations that would alter the represented pointset are easily avoided by preventing:

- DROP operations of active cells,
- JOIN operations when all three cells are not active, and

- INCORPORATE operations when both cells are not active.

Structural geometric information that has to be preserved can be indicated by a **structure** attribute attached to each cell. For example, if the **structure** values of a cell, **a**, differs from the **structure** value of an interior boundary cell **b** in **a**, then **b** would not be incorporated into **a**, even if both **a** and **b** are active. Similarly, if the **structure** attribute of an inactive cell is set, the cell will not be dropped during simplification.

Preventing simplification operations that would destroy the validity of the geometric complex is very important and requires more detailed consideration of the incidence graph and of the associated **extent**s. We present here a simplified version of the necessary tests. A proof of their validity can be obtained by considering operations that are dual to these simplifying operations, that is, operations that insert cells into geometric complexes. It will be presented in a separate report.

- A cell **c** can only be dropped if its **star** is empty, i.e., if it is not used to bound another cell.
- An interior boundary cell, **b**, can only be incorporated into a cell **c**, if **b** lies in **c.extent** and if **c.star** = **b.star** + **c** (that is, if adding to **b.star** a reference to **c** produces the same set of references as **c.star**), i.e., if **b** is not the intersection of **a** with some other cell.
- A k-cell, **b**, can be used to join two (k+1)-cells, **a** and **c**, if **b.extent** is contained in the common **extent** of **a** and **c**, if **b** belongs to both **a.boundary** and **b.boundary**, and if **a.star** = **c.star** = **b.star** + **a** + **c**.

Figure 8 illustrates two situations where simplification would result in invalid geometric complexes.

Figure 8. Invalid simplifications: In the left figure a vertex is shared as a common boundary by two pairs of collinear edges, but cannot be joined with any of the the pairs. In the right figure, an edge bounding two coplanar faces cannot be joined with them because it is the contact between these faces and another cylindrical face.

The simplification process operates in a top-down manner, starting with cells of highest dimensions and finishing with points. For each dimension, k, first the DROP operations are performed for all k-cells that can be dropped without producing an invalid geometric complex, without altering the represented pointset, and without losing any structural information. Then, all k-cells that can be used to join two (k+1)-cells are merged with them. Finally, all k-cells that can be incorporated into a cell of their **star** are incorporated.

The following algorithm summarizes this approach:

```
simplify: SUBR(A)
  FOR k=A.dimension TO 0 DO
    BEGIN
    FOREACH c IN A.cells(k) DO
      IF can_drop(c) THEN drop(c);
    FOREACH c IN A.cells(k) DO
      FOREACH a1 IN c.star.cells(k+1) DO
        FOREACH a2 IN c.star.cells(k+1) DO
          IF a1≠a2 AND can_join(c,a1,a2) THEN join(c,a1,a2);
    FOREACH c IN A.cells(k) DO
      FOREACH a IN c.star.cells(k+1) DO
          IF can_incorporate(c,a) THEN incorporate(c,a);
    END;
  END;
```

The FOREACH loops may be made more efficient by using the hierarchical organization of cells in terms of the set inclusions of their **extents** and varieties, but these speed-ups are not discussed here.

Figure 9 illustrates a sequence of simplification steps in $\mathbf{R}^2$.

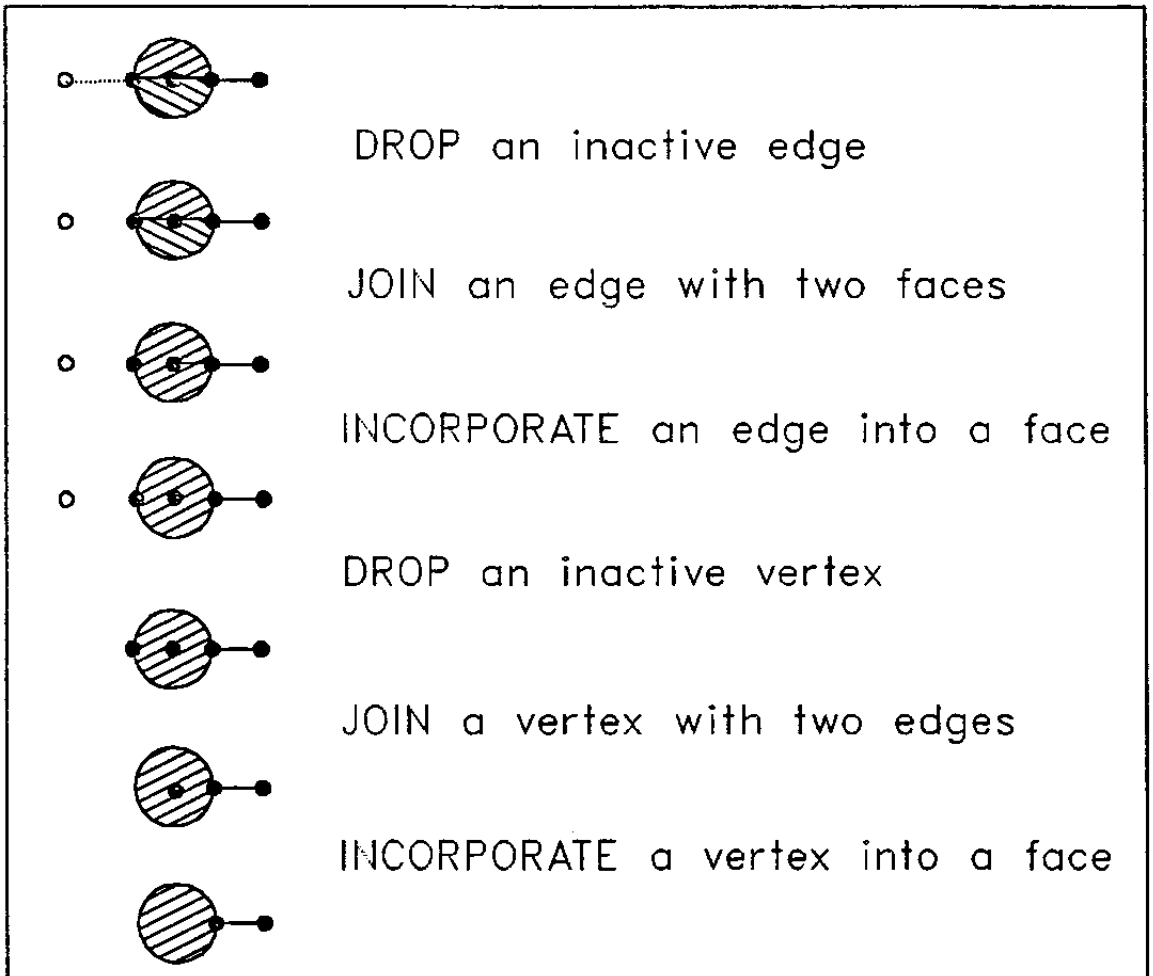

Figure 9. Simplification sequence: The original SGC is composed of 11 active cells (2 faces, 5 edges, and 4 vertices) and of 2 inactive cells (1 edge and 1 vertex). After simplification, the same pointset is composed of only 5 active cells (1 face, 2 edges, and 2 vertices).

We define an SGC, A, to be **simple** if there exists no SGC A' such that A is a proper refinement of A'. Let A.**simplified** be the result of applying the above simplification to A. It can be shown that A.**simplified** is simple, i.e., that one pass of our simplification algorithm, which tests each cell only once, produces an SGC that cannot be further simplified. Furthermore, if two SGCs, A and B, represent the same pointsets, then A.**simplified** and B.**simplified** are identical. Thus, simplification provides a unique SGC representation of pointsets (always an important issue in geometric modelling).

5.4 Applications

Figure 10 illustrates the use of various operations to transform a particular SGC.

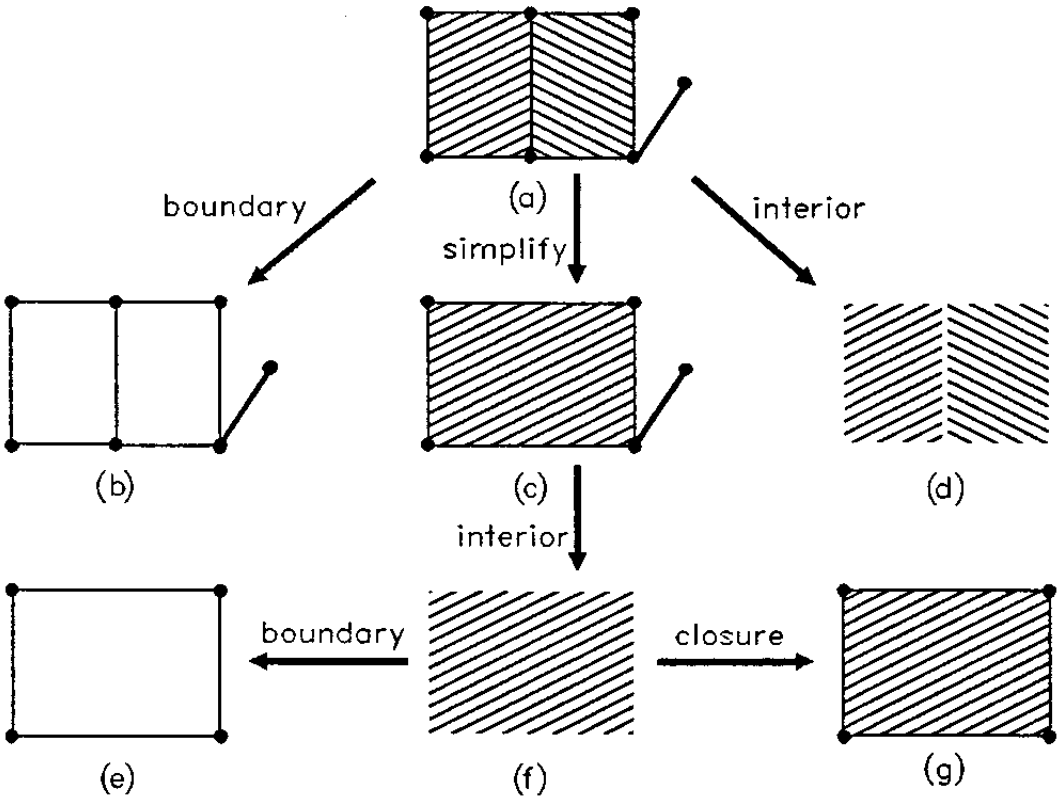

Figure 10. Structural operations: A SGC (a) is transformed by several sequences of operations. Only active cells are shown.

From an SGC, A, defined in $\mathbf{R}^2$ and shown in Figure 10(a), we derive several other SGCs:

1. A.**boundary**, defined as the K.**skeleton**(1) is shown in Figure 10(b).
2. A.**simplified**, the result of the simplification algorithm, obtained by joining two faces and two pairs of edges, is shown in Figure 10(c).
3. A.**interior**, the interior of A, defined as an SGC in which only cells of dimension n in $\mathbf{R}^n$ are active, is shown in Figure 10(d). It is simply obtained as an SGC representation of K.**cells**(2). Note that inactive lower-dimensional cells are not shown.
4. K.**simplified**.**interior**, shown in Figure 10(f), is the topological interior of the pointset represented by A. It may be theoretically derived from (d) by Arbab's regularization, but in practice

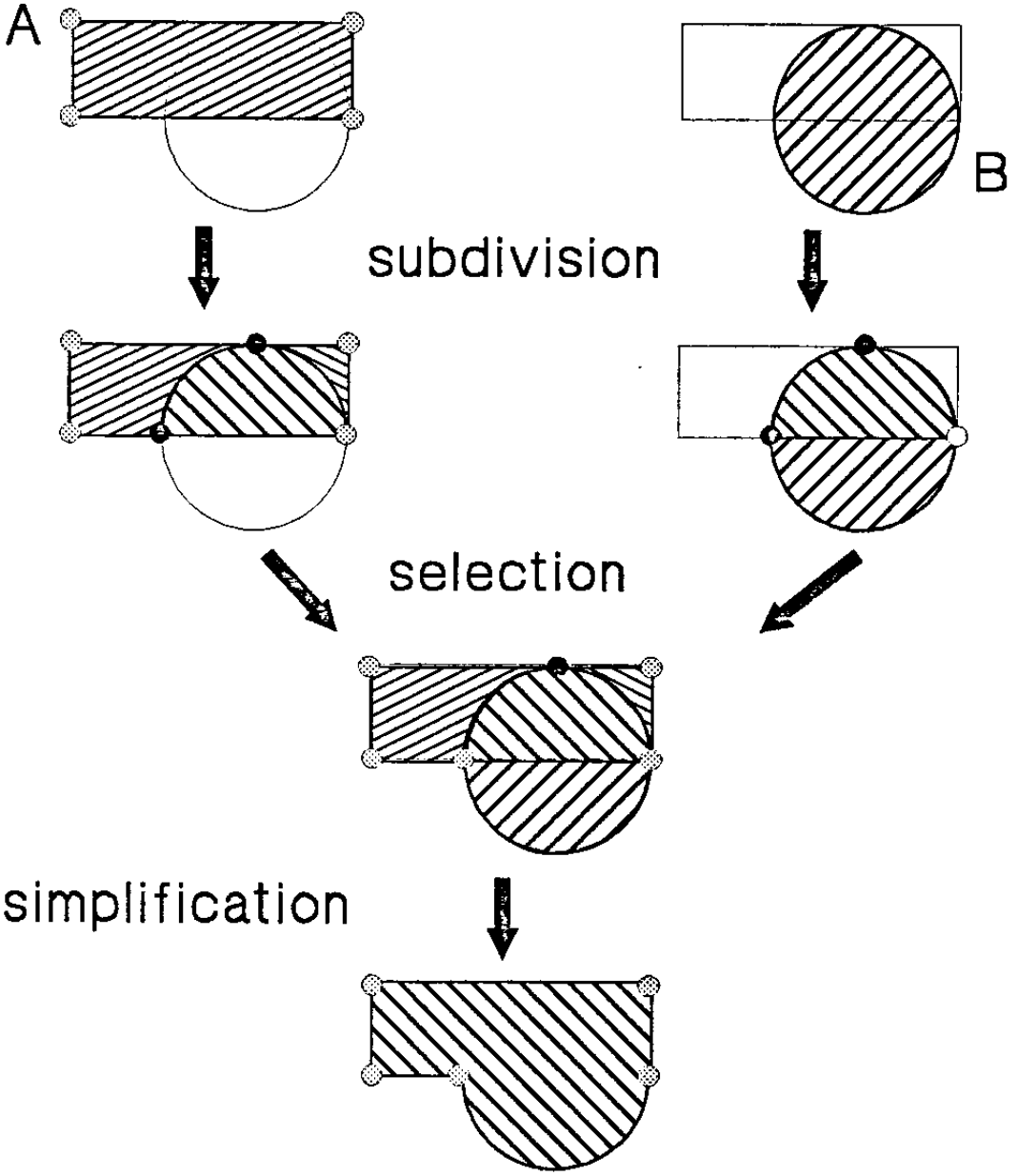

Figure 11. Regularized Boolean operation: The two SGCs, A and B, are first made compatible through subdivision, then their cells are grouped into a common SGC by a selection, and finally, the simplification step is used to reduce the complexity of the representation.

that derivation with SGCs would involve the a closure step, followed by simplification and interior.

5. The regularized version of A is shown in Figure 10(g) and is derived from (f) through a closure operation.

6. The boundary of the regularized version of A is shown in Figure 10(e) and is derived from (f) through a boundary operation.

Figure 11 illustrates the use of the subdivision, selection, and simplification steps for a regularized Boolean operation on two SGCs. Note that after the selection step of Figure 11, the correct pointset for A ∪ B is obtained. The simplification is used to remove boundary cells that are interior to A ∪ B and performs the cleaning step of merging operations used for boundary evaluation in CSG. The simplification step is essential for obtaining the outside boundary.

6.0 CONCLUSION

Geometric modelling is central to many CAD/CAM applications. Different applications are based on different representations and use different design interfaces, which makes communication between applications very difficult. We propose a unified representation scheme, called SGC, that could be shared by many applications. SGCs can represent a large class of pointsets, including open pointsets, non-manifolds, and sets with cracks. SGCs can also capture the internal structure of such pointsets ,such as a finite element mesh decomposition. Furthermore, attributes attached to individual geometric elements may be used to capture the provenance, or design history, of these elements, to mark elements that convey important structural information, or to efficiently interrogate the relative position of certain constituents without additional geometric calculations.

We provide precise mathematical definitions that characterize valid SGCs and describe a family of algorithms that transform and combine SGCs to produce the effect of topological, Boolean, and structural operations on pointsets. The set of SGCs is closed under all these operations, which guarantees that the result of any of these operations can be used as input to other operations. Both the representation scheme and the algorithms have been developed independently of dimension, and thus can be used for objects of dimension higher than three. On the other hand, the geometric algorithms used for evaluating boundaries of solid models defined in CSG as Boolean combinations of solid primitives could be used with minor modifications to support the geometric operations on SGC, at least in R^3.

7.0 ACKNOWLEDGEMENTS

We are indebted to V. Srinivasan for supporting and fostering this work, to M. Wesley and A. Sharma for believing in its potential applications to the three-dimensional simulation and analysis of the effects of VLSI manufacturing processes, and to numerous colleagues and friends for showing so much interest in our work and for urging us to disseminate it.

8.0 REFERENCES

[1] I.D. Faux and M.J. Pratt, *Computational Geometry for Design and Manufacture*. Chichester, U.K.: Ellis Horwood, 1979.

[2] A.A.G. Requicha and H.B. Voelcker, "Solid modelling: A historical summary and contemporary assessment", *IEEE Computer Graphics and Applications*, vol. 2, no. 2, pp. 9-24, March 1982.

[3] A.A.G. Requicha and H.B. Voelcker, "Solid modelling: Current status and research directions", *IEEE Computer Graphics and Applications*, vol. 3, no. 7, pp. 25-37, October 1983.

[4] C.M. Eastman and K. Weiler, "Geometric modelling using the Euler operators", *Proc. First Annual Conf. on Computer Graphics in CAD/CAM Systems*, MIT Cambridge, MA, pp. 248-259, April 9-11, 1979.

[5] I.C. Braid, "Superficial blends in geometric modelling", C.A.D. Group Document No. 105, Univ. of Cambridge, February 1980.

[6] M. Mantyla and R. Sulonen, "GWB: a solid modeller with Euler operators", *IEEE Computer Graphics and Applications*, vol. 2, no. 7, pp. 17-31, September 1982.

[7] M. Mantyla, "A note on the modeling space of Euler operators", *Computer Vision, Graphics, and Image Processing*, vol. 26, pp. 45-60, April 1984.

[8] A.A.G. Requicha and H.B. Voelcker, "Constructive solid geometry", Tech. Memo. No. 25, Production Automation Project, Univ. of Rochester, November 1977. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, New York 14853.)

[9] A.A.G. Requicha, "Mathematical models of rigid solid objects", Tech. Memo. No. 28, Production Automation Project, Univ. of Rochester, November 1977. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, New York 14853.)

[10] J.R. Rossignac, P. Borrel, and L.R. Nackman, "Interactive design with sequences of parameterized transformations", *Proceedings of the Second Eurographics Workshop on Intelligent CAD Systems: Implementation Issues,* April 11-15, 1988, Veldhoven, The Netherlands, pp. 97-127. Also available as IBM Research Report RC 13740, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY, May 1988.

[11] I.C. Braid, "The synthesis of solids bound by many faces", *Communications of the ACM*, vol. 18, no. 4, pp. 209-216, April 1978.

[12] B.G. Baumgart, &qu.Winged edge polyhedron representation", AIM-79, report STAN-CS-320, Stanford University, Stanford, CA. 1972.

[13] H. Edelsbruner, J. O'Rourke, R. Seidel, "Constructing arrangements of lines and hyperplanes with applications", *Proc. 24th Symp. of Foundations of Computer Science,* 1983.

[14] J.W. Boyse and J.E. Glichrist, "GMSolid: Interactive modelling for design and analysis of solids", *IEEE Computer Graphics and Applications*, vol. 2, no. 2, pp. 27-40, March 1982.

[15] A. Parkinson, "The use of solid models in BUILD as a database for NC machining", *Proc. Prolamat 1985,* Paris, France, pp. 293-299, June 11-13, 1985.

[16] W.A. Hunt and H.B. Voelcker, "An exploratory study of automatic verification of programs for numerically controlled machine tools", Tech. Memo. No. 34, Production Automation Project, Univ. of Rochester, January 1982. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, New York 14853.)

[17] M.A. Wesley, T. Lozano-Perez, L.I. Lieberman, M.A. Lavin, and D.D.Grossman, "A geometric modelling system for automated mechanical assembly", *IBM Journal of Research and Development*, vol. 24, no. 1, pp. 64-74, January 1980.

[18] U.A. Sungurtekin and H.B. Voelcker, "Graphical simulation and automatic verification of NC machining programs", *Proc. IEEE International Conference on Robots and Automation*, vol. 1, pp. 156-165, April 1986.

[19] R.N. Wolfe, M.A. Wesley, J.C. Kyle, F. Gracer, W.J. Fitzgerald, "Solid Modelling for Production Design" *IBM Journal of Research and Development*, vol. 31, no. 3, pp. 277-295, May 1987.

[20] G.M. Koppelman and M.A. Wesley, "OYSTER: A study of integrated circuits as three-dimensional structures" *IBM Journal of Research and Development*, vol. 27, no. 2, pp. 149-163, March 1983.

[21] S.A. Cameron, "A study of the clash detection problem in robotics", *International Conference on Robotics and Automation*, St. Louis, pp. 488-493, 1985.

[22] A.P. Rockwood, "Introducing sculptured surfaces into a geometric modeler", in M. S. Pickett and J. W. Boyse, Eds., *Solid Modeling by Computers*. New York: Plenum Press, pp. 237-258, 1984.

[23] R.F. Sarraga and W.C. Waters, "Free-form surfaces in GMSOLID: Goals and issues", in M. S. Pickett and J. W. Boyse, Eds., *Solid Modeling by Computers*. New York: Plenum Press, pp. 187-209, 1984.

[24] R.B. Tilove, "A null-object detection algorithm for Constructive Solid Geometry", *COMM. ACM*, vol. 27, no. 7, pp. 684-694, July 1984.

[25] Y.T. Lee and A.A.G. Requicha, "Algorithms for computing the volume and other integral properties of solids: I - Known methods and open issues, II - A family of algorithms based on representation conversion and cellular approximation ", *Comm. ACM*, vol. 25, no. 9, September 1982.

[26] A.A.G. Requicha and R.B. Tilove, "Mathematical foundation of constructive solid geometry: General topology of closed regular sets", Tech. Memo. No. 27a, Production Automation Project, Univ. of Rochester, June 1978. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, New York 14853.)

[27] C.M. Brown, "PADL-2: A technical summary", *IEEE Computer Graphics and Applications*, vol. 2, no. 2, pp. 69-84, March 1982.

[28] K.J. Weiler, "Edge-based data structure for solid modelling in curved surface environments", *IEEE Computer Graphics and Applications*, vol. 5, no. 1, pp. 21-40, January 1985.

[29] M. Mantyla, "Boolean Operations of 2-Manifolds through Vertex Neighborhood Classification", *ACM Transactions on Graphics*, vol. 5, no. 1, pp. 1-29, January 1986.

[30] N.M. Samuel, A.A.G. Requicha and S.A. Elkind, "Methodology and results of an industrial part survey", Tech. Memo. No. 21, Production Automation Project, Univ. of Rochester, July 1976.

[31] I.C. Braid, *Designing with volumes*, Cantab Press, Cambridge, England, 1974.

[32] R.B. Tilove and A.A.G. Requicha, "Closure of Boolean operations on geometric entities", *Computer Aided Design*, vol. 12, no. 5, pp. 219-220, September 1980.

[33] A.A.G. Requicha and H.B. Voelcker, "Boolean Operations in Solid Modelling: Boundary Evaluation and Merging Algorithms", *Proceedings of the IEEE*, Vol. 73, No 1, January 1985.

[34] S.D. Roth, "Ray casting for modeling solids", *Computer Graphics and Image Processing*, vol. 18, no. 2, pp. 109-144, February 1982.

[35] P. Hanrahan, "An introduction to relational topology", *Proc. CAD'84*, Brighton, U.K., pp. 461-469, April 3-5, 1984.

[36] J.R. Rossignac and H.B.Voelcker, "Active Zones in CSG for Accelerating Boundary Evaluation, Redundancy Elimination, Interference Detection, and Shading Algorithms", *ACM Transactions on Graphics*, vol. 8, no. 1, January 1989.

[37] A.A.G. Requicha and J. Vandenbrande, "Automatic process planning and part programming", Institute for Robotics and Intelligent Systems, Report IRIS 217, University of Southern California, Los Angeles. April 1987.

[38] A.P. Morgan and R.F. Sarraga "A method for computing three surface intersection points in GMSOLID", Report GMR-3964, General Motors Research Labs., Warren, MI, February 1982.

[39] R.B. Tilove, "Set membership classification: A unified approach to geometric intersection problems", *IEEE Trans. on Computers*, vol. C-29, no. 10, pp. 874-883, October 1980.

[40] C. E. Silva, "Alternative definitions of faces in boundary representations of solid objects" Tech. Memo. No. 36, Production Automation Project, Univ. of Rochester, November 1981.

[41] F. Arbab, "Set models and Boolean operations for solids and assemblies", Technical Report CS-88-52, Computer Science Department, University of Southern California, Los Angeles, California, 90089-0782, July 1985.

[42] C. Hoffmann and J. Hopcroft, "Geometric ambiguities in boundary representations", TR 86-725, Computer Science Dept., Cornell Univ., Ithaca, NY, January 1986.

[43] C. Hoffmann, "Applying algebraic geometry to surfaces intersection evaluation", TR CSD-TR-772, Computer Science Dept., Purdue Univ., West Lafayette, Ind. 47907. April 1988.

[44] H. Whitney, "Elementary structure of real algebraic varieties", *Annals of Mathematics*, vol. 66, no. 3, pp. 545-556, November, 1957.

[45] C.T.C. Wall, "Regular stratifications", *Lecture notes on dynamic systems* WARWACH, Springer-Verlag, 1974.

[46] G.E. Collins, "Quantifier elimination for real closed fields by cylindrical algebraic decomposition", Lecture notes in Computer Science, No. 33, Springer Verlag, New York, 1975.

[47] F.E.A. Johnson, "On the triangulation of stratified sets and singular varieties", *Trans. Amer. Math. Soc. 275, 1983.*

[48] W. Boothby, *An introduction to differentiable manifolds and Riemannian geometry*, Academic Press, Inc., New York, 1986.

[49] F. Warner, *Foundations of differentiable manifolds and Lie groups*, Scott, Foresman and Co., Glenview, Illinois, 1971.

[50] R.F. Sarraga, "Algebraic methods for intersections of quadric surfaces in GMSOLID", *Computer Vision, Graphics and Image Processing*, vol. 22, no. 2, pp. 222-238, May 1983.

[51] M.A. O'Connor, "Projection of natural quadrics", IBM Research Report RC 11188, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, June 1985.

[52] R.T. Farouki, C.A. Neff, and M.A. O'Connor, "Automatic parsing of degenerate quadric-surface intersection", IBM Research Report RC 13952, IBM Thomas J. Watson Research Center, Yorktown Heights, August 1988.

[53] S. Ocken, J.T.Schwartz, and M. Sharir, "Precise implementation of CAD primitives using rational parametrizations of standard surfaces", Technical Report No. 67, Courant Institute of Mathematical Sciences, New York University, Computer Science Division, 251 Mercer Street, New York, N.Y. 10012, March 1983.

[54] L. Guibas and J. Stolfi, "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams", *ACM Transactions on Graphics,* vol. 4, no. 2, pp. 75-123, April 1985.

[55] E.H. Spanier, *Algebraic Topology*, Springer Verlag, New York, 1966.

[56] M.J. Laszlo, "A data structure for manipulating three-dimensional subdivisions", PhD dissertation, Report CS-TR-125-87, Department of Computer Science, Princeton University, August 87.

[57] K.J. Weiler, "The radial edge structure: a topological representation for non-manifold geometric modeling", in *Geometric Modeling for CAD Applications,* M. Wozny, H. McLaughlin, and J. Encarnacao, Edts., Springer Verlag, was expected December 1987.

[58] K.J. Weiler, "Boundary graph operators for non-manifold geometric modeling representations", in *Geometric Modeling for CAD Applications,* M. Wozny, H. McLaughlin, and J. Encarnacao, Edts., Springer Verlag, was expected December 1987.

[59] H. Bieri and W. Nef, "Elementary set opertations with d-dimensional polyhedra". Institut für Informatik und angewandte Mathematik, Universität Bern, Länggasßtrasse 51, CH-3012 Berne. Lecture given at the 4th Workshop for Computational Geometry, University of Würzburg, March 24/25, 1988.

[60] J. Miller, "Geometric approaches to nonplanar quadric surface intersection curves", *ACM Transactions on Graphics,* vol. 6, no. 4, pp. 274-307, October 1987.

[61] M.A. O'Connor and J.R. Rossignac, "Robust utilities for geometric modelling with natural quadrics", IBM Research Division, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, in preparation.

[62] G. Vanecek and D. Nau, "Non-regular decomposition: An efficient approach for solving the polygon intersectino problem", *Proc. Symposium on Integrated and Intelligent Manufacturing at the ASME Winter Annual Meeting.* pp. 271-279, 1987.