ACM SIGGRAPH 1996, Course 29: Representation of Geometry for Computer Graphics

Specification, representation, and construction of non-manifold geometric structures

May 3rd, 1996

Jarek R. Rossignac

IBM, T.J. Watson Research Center
Yorktown Heights, New York 10598

Siggraph 96 course: Computational Representations of Geometry

Abstract

Geometric modelling is central to many applications. Representation schemes that are specialized for a particular application may impose topological and geometric limitations on the domain and thus considerably restrict future extensions. Selective Geometric Complexes (SGCs) provide a practical yet general framework for representing general objects of mixed dimensionality having internal structures and incomplete boundaries. SGCs and their decomposition into regions (i.e., features of interests for a particular applications) may be specified and edited in terms of Constructive Non-Regularized Geometry (CNRG) trees, which define how primitive shapes should be combined through a variety of set-theoretic and topological operators. CNRG operators preserve the structure imposed by their arguments on the underlying set. The combination of CNRG specification and of SGC representation and the associated conversion/evaluation algorithms provide a generalized environment for non-manifold modeling in any dimension. These notes focus on the topological concepts, on the representation and specification schemes, and on the associated algorithms for non-manifold structures, independently of any particular geometric domain (i.e., restriction on the shapes or surfaces) and of the dimension of the underlying space.

Table of Contents

1.0 FORWARD

Computerized models of three-dimensional shapes are central to many applications: manufacturing, geoscience, entertainment, architecture, and medicine are obvious examples. Natural or man-made shapes may be modeled electronically in a variety of ways and with different degrees of accuracy. The choice of a particular modeling scheme and of the associated data-structures often depends on the data acquisition process, on requirements imposed by the application, and of course on the skills, number, ambitions, and preoccupations of the developers.

Four types of representations interplay in many geometric modeling scenarios.

In Computer-Aided Design systems, an intentional model may be used to capture, in an unevaluated form, some of the functional requirements of a product or the designer's intent. One may use a procedural model (programming language, Constructive Solid Geometry, or more general creation history graphs which include Boolean, blending, and deformation operators) or a declarative form (geometric constraints, shape grammars, construction rules). Such models may in general be easily edited or parameterized.

Real or computed shapes may be captured in a sample model derived from physical measurements (seismic data, slices of a medical scan, depth maps from a range finder) or from the results of numeric simulations. Sample models provide shape information only over a set of discrete sample points placed either on the surfaces of the model or distributed in space. Shape between the samples is not well defined and various algorithmic techniques have been proposed to construct continuous interpolations between samples and to decide which neighboring samples should be used for evaluating a given point.

A continuous extensional (i.e. evaluated) model is typically derived algorithmically from an intentional model, a sample model, or both. For example, an operator may design a multi-surface object using crude approximations for each surface and subsequently force each surface to pass through or near a set of sample points measured off a real object or computed via numeric optimization. Most representation schemes for extensional models are enumerative in nature. The most popular representations are dimensionally homogeneous partition of space (voxels, octrees, 3D meshes, BSP trees) or recursive boundary formulations (BReps), where volumes are defined in terms of their bounding faces and where faces are defined in terms of their supporting surfaces and bounding edges.

An abstraction model provides mechanisms for selecting, identifying, or iconifying those subsets of the extensional model's interior or boundary that are appropriate for a particular operation or relevant to a particular annotation. Typical examples include surface features for machining, entities used in the definition of geometric constraints stored in intentional models, or tumors in a segmented 3D medical dataset. Abstraction models may impose an internal structure on the point set of the model.

This lecture focuses on the extension of traditional intentional models (Constructive Solid Geometry) and continuous extensional models (Boundary Representations) to arbitrary topological domains and to pointsets with internal structure. The theoretical underpinnings supporting these extensions are independent of the particular choice of a representation for the individual geometric entities, and hence of the geometric coverage of the modeler. The algorithms developed for constructing and for processing such representation assume however the reliable support of a small set of geometric queries, such as the intersection and ordering of geometric entities.

2.0 INTRODUCTION

The primary schemes for representing three-dimensional geometric objects may be grouped into three broad categories: constructive, boundary, and enumerative representations. By storing a recipe (process) for creating a model from primitive entities and operations, constructive representations capture the designers' intent and provide a powerful design model that is easy to edit and to parameterize. Enumerative representations represent or approximate the desired regions as a collection of primitive entities that are simpler to represent. Enumerative schemes may require that the primitives be mutually disjoint (or at least that their relative interiors be disjoint) and often restrict the primitives to be regularly spaced. Most popular schemes use cubical cells (called voxels), parallelepipeds of uniform cross-section (Ray representations [12]), or constant-thickness slices (cross-sections). These representations may be constructed directly from physical measures or from simulation results or evaluated from other representations. Although often approximate, they have recently gained popularity, because their simplicity is well suited for parallel hardware implementation (see papers in [20]). Boundary representations exploit the fact that a simple enumeration of the bounding (hyper)faces of a bounded region suffices to unambiguously distinguish it from its complement. Most boundary models, however, store additional connectivity information between the geometric boundary elements (vertices, edges, and faces) and exploit it to speed up the boundary traversal parts of fundamental algorithms that build the model from a constructive representation, display it, or extract its topological or geometric properties.

Practitioners often distinguish between the topology and the geometry of a model. While a geometric representation captures the precise shape of each face, or curve of an object, a topological representation focuses on properties that are invariant under continuous deformations, i.e. that are independent of the precise shape of the geometric components. This lecture notes address precise representations of continuous three-dimensional geometric models, because these are important for precise design, visualization, and analysis. Therefore we focus primarily on the topological, representational, and computational aspects of constructive and boundary representation that are independent of any particular geometric domain (i.e. of the nature of the surfaces represented or even of the dimensionality of the problem). Enumerative representations are addressed elsewhere in this course.

Geometric modelling is central to many design, simulation, visualization, analysis, and manufacturing applications. Different applications deal with different geometric and topological entities and thus require different geometric and topological coverages. The geometric coverage of a modeler is characterized by the nature of the geometric entities (such as points, curves, surfaces) it supports and by the ways in which these entities may be created, combined, and manipulated. Topological limitations of a modeler are more subtle to assess. For example, the solid modeling technology is based on a precise definition of solids leading to a complete and unambiguous representation that permits to distinguish between the interior, the boundary, and the exterior of the represented solid [29]. This definition played an essential role in the development of correct algorithms for Boolean operations on solids [30], but has somewhat confined the domain of applications. Indeed, until recently, solid modelers did not explicitly support internal structures nor lower-dimensional (dangling) entities. Surfaces have been extensively used for car body design and are required for representing regions of contacts between solids. Interior faces are used to decompose solids into finite elements or into subsets exhibiting different physical properties. They may also represent cracks in three dimensional sets. Curves may be used as design aides. Although solids, curves, and surfaces can be grouped (overlayed) in the same model and moved or displayed together as a single entity, Boolean and other operations on such groupings have not been implemented, nor even formally defined.

Selective Geometric Complexes (abbreviated SGC) introduced by Rossignac and O'Connor [33] provide 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. A cell is an open connected subset of some n-dimensional manifold. The concept of a cell generalizes the concepts of edges, faces, and vertices used in most solid modelers. The connectivity between the cells of an SGC 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 that are 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, Boolean and other set-theoretic operations (closure, interior, boundary) have been decomposed 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.

Solids may be conveniently specified in CSG (Constructive Solid Geometry) by a construction tree that has solid primitives as leaves and rigid body motions or regularized Boolean operations as internal nodes. Algorithms for classifying sets with respect to CSG trees and for evaluating the boundaries of the corresponding solids are known, at least for simple geometric domains. Emerging CAD applications require that we extend the CSG simplification to support more general and more structured geometric objects. The concept of a Constructive Non-Regularized Geometry (abbreviated CNRG) was introduced by Rossignac and Requicha [34] to support a convenient specification of dimensionally non-homogeneous, non-closed pointsets with internal structures. These cover non-manifold structures possibly composed of several mixed-dimensional regions with dangling or missing boundary elements. CNRG trees extend the domain of CSG by supporting non-regularized primitive shapes as leaves and by providing more general set-theoretic and topological operators at interior nodes. Filtering operations construct CNRG objects from aggregates of selected regions of other CNRG objects. The resulting structures may be evaluated and represented in terms of SGC, where references to individual cells are grouped into CNRG regions.

The combination of CNRG specification and of SGC representation and the associated conversion/evaluation algorithms provide a generalized environment for non-manifold modeling in any dimension. However, most of the concepts are introduced using two or three dimensional instantiations for sake of clarity.

These notes are organized as follows. Section 2 introduces the concepts of topological domains and representation validity through the case study of 2-D polygons. It argues that the correct rephrasing of the question “What is a valid polygon?” leads to a precise definition of a polygon and to a natural canonical representation scheme. Section 3 introduces the fundamental concepts of non-manifold modeling and the diversity of topological domains that fall under this imprecise denomination. The discussion starts with an introduction of a decomposition of space into a collection of disjoint cells induced by a given set of primitives (for example, surfaces and curves). Then it discusses the identification of particular cell in such a decomposition. A variety of topological domains may be obtained by restricting which cells of a decompositions should be active (i.e. contributing to the pointset). Section 4 reviews boundary modeling schemes. It introduces the fundamental entities and the topological and ordering relations between these entities. It overviews several popular data structures, including SGCs. Section 5 reviews the main CNRG concepts and the semantics of CNRG operators for specifying and computing non-manifold sets with structures. The appendix contains a short (informal) review of the topological concepts used elsewhere in these notes.

3.0 WHAT IS A VALID POLYGON?

The naive question: “What is a valid polygon?” should lead to a mathematical definition of validity for polygonal representations accepted, processed, or produced by a particular application or algorithm. Numerous definitions have been published in research papers or user manuals. The reader may be puzzled

by the diversity of such definitions and struggle when asked to establish if two definitions are equivalent, if a particular definition is complete, or even if a specific case satisfies a particular definition.

It is helpful to decompose this questions as follows:

1. "What is a polygon?"
2. "What representation scheme (abstract data structure) are we using?"
3. "How do we semantically interpret the representation (for example when testing whether a point lies inside the represented polygon?"
4. "When is a model expressed using that representation scheme valid (i.e., corresponds to a polygon according to the chosen definition)?"

For the sake of elegance, we propose here a definition that is restrictive, but leads naturally to a simple and canonical (i.e. unique) representation scheme.

A polygon is a connected and bounded s-regular subset of E^2 having for boundary a finite union of mutually disjoint cells. Cells are either **crossings:** (i.e., non-manifold points), **vertices:** (i.e. non-smooth manifold points), or **edges:** (i.e., relatively open connected line segments free of crossings).

A set is s-regular if it is equal to the interior of its closure. Such a set is open and thus does not contain its boundary (i.e. its edges, vertices, or crossings).

According to this definition the following statements hold.

A polygon is open (does not contain its boundary) and connected. It may have holes but no islands.

A polygon has no dangling edges, isolated vertices, interior cracks or missing points.

The boundary of a polygon needs not be manifold. Its vertices, crossings, and edges of a polygon are pairwise disjoint. The vertices, crossings, and edges of any given polygon are always unambiguously defined.

Each edge of a polygon separates the inside of the polygon from the outside. The orientation of each edge (following the convention that the inside is on the left) is thus well defined. Each edge is incident on exactly two points (vertices or crossings) and its orientation defines which is the start point and which is the end point.

Each vertex is the start point of exactly one edge and the end point of exactly one other edge. Each crossing has 2k (k positive integer) edges incident on it. It is the start point of exactly k of these.

The successor of an edge E having vertex V as its end point is defined as the only edge having V as its start point. The successor of an edge E having crossing C as its end point is defined as the first edge that has C as its start point as we circle C clockwise in its immediate vicinity starting from E. The successor of an edge is uniquely defined

A loop is defined as a maximally connected subset of the boundary of a polygon. The loops of a polygon are uniquely defined and pairwise disjoint.

Each edge and each vertex or crossing belong to exactly one loop. The successor of an edge belongs to the same loop.

The successor operator (which returns the successor of the argument edge) induces a unique cyclic ordering for all the edges in a loop.

Edges may be uniquely defined by the references to their start and end points. Since the end point of an edge is equal to the start point of its successor, one reference per edge in a loop suffices for defining the

edges. Thus, a loop is completely represented by an ordered circular list of references to the start points of its edges

Each loop has at least 3 references to different points. The starting points of two consecutive edges in a loop are always different (i.e. consecutive point-references in a loop must be different).

These properties lead to a simple and canonical representation:

A polygon may be represented by the set of its points (vertices and crossings) and the set of its loops. Each point is represented by its coordinates. Each loop is represented by a circularly ordered list of references to its points.

Note that given any polygon (i.e. a pointset that meets the proposed definition), its representation in the above scheme (data structure) is unique. (We do not take into account the possible permutations and various representations for sets and list, which can be addresses by imposing the appropriate lexicographic orderings.) For processing convenience, one could also require that the outer loop (which is also always well defined in the plane and the holes (all other loops) be explicitly identified in the data structure.

The opposite assumption is unfortunately not true. A dataset organized in the above data-structure (sets of points and loops) does not necessarily correspond to a valid polygon. Validity violations may be of different nature: geometric, ordering, or topological. Geometric violations correspond to the wrong choice of point coordinates (for example, a point may coincide with another point or with an edge, or two edges may intersect). Ordering violations may simply correspond to the wrong orientation of the edges in a loop or to the wrong branch taken at a crossing (inconsistent with the definition of a successor). Topological violations may correspond to empty edges (two consecutive references to the same point), to degenerate loops (less than three point references), or to loops with non-manifold parts (for example, the multiple use of an edge).

Imposing additional constraints on the representation further restricts the set of representable polygons. For example, if each vertex is used only once in a loop, the polygon has a manifold boundary. If the polygon has a single loop, it is simply connected.

A number of other schemes for representing polygons are popular in CAD and graphics systems. They include:

- Decompositions into simpler disjoint regions (triangles, trapezoids)
- CSG or BSP trees,
- Possibly overlapping trimming loops

These do not easily lead to canonical representations.

4.0 TOPOLOGICAL BACKGROUND

Any set of geometric primitives may be used to impose a decomposition of the underlying three-dimensional Euclidean space into cells of an SGC structure from which one can select a subset of interest (the “active” cells). We describe here informally several ways of defining such a decompositions.

A single primitive decomposes space into three disjoint parts:

- the primitive's interior
- its boundary
- the interior of its complement

These sets may be recursively decomposed into dimensionally homogeneous subsets (i.e. fully three-dimensional volumes, isolated points, dangling curves, and dangling faces). From each subset, one may extract singular points (cusps and self-crossings where some geometric continuity or topological manifold

properties of the supporting geometry vanish) and boundary points (vertices bounding edge segments and edges bounding dangling faces). The sets of singular points may be further (recursively) decomposed in this manner into singularity-free, dimensionally homogeneous sets. Finally, the maximally connected components. of this decomposition may be identified. For example, a cone primitive may decompose space into: the complement of the cone, the three-dimensional interior of the cone, the circular edge at the base, the disk-like base-face (without its bounding circular edge), the apex (tip of the cone), and the conical face (without the bounding edge nor the apex).

The atomic entities defined by this decomposition process correspond to the cells of a geometric complex. They are connected relatively open subsets of some n-dimensional manifold (the 3-D space or a smooth portion of a surface or of a curve). Any two different cells of such a decomposition are mutually disjoint. The boundary of the set of a cell C lying in a manifold M is the union of other cells in the decomposition, which are either entirely in the manifold M, or entirely out if it. For example the boundary of the conical face of a cone primitive is composed of a circular curve (part of the manifold surface supporting the face) and of the apex (singular point, not in the manifold supporting the face).

When several primitives overlap, the space decompositions induced by each primitive must be combined into a single (finer) decomposition. An algorithm for performing such a merging operation is called subdivision (or refinement) and is outlined in [33]. It basically requires that intersections of each cell of one decomposition with each other cell of each other decomposition be computed and further decomposed into dimensionally homogeneous singularity-free connected components.

The space partition induced by a set of planes is a simple example of decomposition. It cells are: points where three or more planes cross, relatively open (and possibly unbounded) line segments where two or more planes cross, relatively open convex polygonal faces induced on each plane by all other non-coincident planes, and the open convex polyhedra (maximally connected components of the complement of the union of all the planes).

4.1 Identifiable sets of cells

A set of primitives used to induce a decomposition of space provides the means for characterizing specific subsets of these cells. Any cell of such a linear partition may be expressed using a Boolean formula which combines the half-spaces bounded by the original planes and involves only set theoretic intersection and complement operators. Yet, the characterization of the union of several cells may be more conveniently expressed in terms of a filter operator. The filters may use the geometric or topological properties of a cell's set or the relation of a cell to any particular primitive. Filters may be categorized as follows.

- Boolean combinations of primitives half-spaces through set theoretic union, intersection, difference, and complement operators are the most popular filters.
- Dimensionality, adjacency, and incidence may also be used for discriminating cells. For example, given a sphere and its center point, the complement of the sphere may be characterized (among many other ways) as the only 3-cell that is not adjacent to a 0-cell.
- Topological operators, such as relative closure, interior, boundary, may be applied to sets of cells and are unambiguously defined in terms of the union of the point-sets represented by the cells [34].
- Geometric filters, which for example extract the one-dimensional singularities, are also important.

These filters and operators may be composed into expressions that may uniquely identify specific sets of cells. However, they may be insufficient for differentiating between specific pairs of cells: the connected components of the result of filtering expressions. A simple example is an infinite line subdivided by a point. It is impossible to distinguish the two half-lines using only combinations of the above operators. The orientation notions discussed in the next section may help in some cases, but the problem remains unsolved in general, and leads to serious complications for picking faces and defining features in parametric models.

Note that arbitrary subsets of the cells of a decomposition do not necessarily correspond to a geometric complex (the boundaries of some cells may be missing). Therefore, to represent the result of a selection (filtering) process, one should construct a Selective Geometric Complex that includes not only the selected cells, but also all the cells in the closure of the desired set. The added boundary cells will simply be inactive.

Since the above decomposition process often introduces unnecessary subdivisions of the desired set, it may be desirable to simplify the representation, i.e. to produce a simpler SGC representing the same set. A systematic simplification process which preserves the validity of the SGC is described in [33]. The simplification process visits each cell only once (by order of decreasing dimensionality) and applies, if appropriate, one of the following three operations: (1) remove inactive cells that are not bounding other active cells, (2) remove active cells that separate two active cells in the same manifold and are not bounding any other cell, and (3) remove active cells that are interior boundaries of only one active cell. The simplification of [33] provides the means of generating a unique SGC for a given set. That is, if two SGCs, A and B, represent the same sets, their simplifications will be identical SGCs.

4.2 Topological characterization

Developers of geometric modeling systems have often restricted the topological domains in various ways. We characterize such restrictions using topological concepts, regardless of the geometric domain or associated data structures, which are discussed in the following section. The characterization is based on the topological properties of the sets (or collections of sets) that can be represented. We use 2-D terminology to and examples illustrate the differences.

- R-set with manifold boundary: Each vertex is adjacent to exactly two edges.
- R-set with non-manifold boundary: A vertex may bound more than two edges, but the set is equal to the closure of its interior. (Note that regions whose interior are disjoint may share common vertices, but not common edges.)
- S-sets: Open-regularized regions composed of open subsets that are disjoint, but whose boundaries need not be disjoint. Edges bounding two subsets are called “interior”. However, each region is equal to the interior of its closure (i.e. does not contain dangling edges nor cracks).
- Non-regular open sets: Extensions of s-sets that may contain non-separating interior boundary elements, but no dangling lower dimensional entities.
- Inhomogeneous closed sets: Extensions of r-sets with non-manifold to possibly include dangling edges or isolated vertices.
- One-dimensional non-manifold set: Union of edges and vertices that may separate its complement into more than two connected cells. (Such sets are typically called “non-manifold boundaries”, although there does not necessarily exist a bounded and closed or open 2-D region having such a set for boundary.)
- Partially closed sets: Extensions of non-regular open sets to include subsets of their boundaries.
- Disconnected non-regularized set: Extensions of partially closed sets to include dangling edges and vertices. Interior edges are not part of the set.
- Closed non-manifold structures: Subdivisions of closed non-manifold sets into distinguishable subsets. (The boundary of each cell is included in the set.)
- Selective Geometric Complex: Combinations of closed non-manifold structures and disconnected regularized sets. (Interior edges need not be in the set.) SGCs are collections of disjoint relatively open cells—here, 2-cells, edges, and vertices.

Another important characterization addresses restrictions that force the decomposition of natural topological entities into collections of simpler ones. SGCs require that each cell be connected. Thus, the volumes, faces, and edges of a model represented as an SGC must be broken into connected components, each represented by a separate cell.

Restrictions on faces imposed by representations inspired by CW-complexes require that faces with holes be artificially converted into simply connected faces by adding “bridges” (i.e. pairs of edges with opposite orientations that join loops) to merge the various loops into a single loop. Similarly, closed curves and

surfaces that are not homeomorphic to a disk (2-ball) must be artificially split by adding “cuts” (i.e. vertices or edges). Faces are often assumed to be regular (i.e. equal to the interior of their closure). Such a restriction avoids internal face-boundaries, but makes it impossible to represent correctly certain non-manifold r-sets. Restrictions on the face-bounding loops may require that loops be mutually disjoint and 1-manifold. They also lead to serious limitations.

5.0 BOUNDARY REPRESENTATION SCHEMES

We briefly review in this section the most popular data structures for boundary models. A more detailed analysis may be found in [1, 4, 22, 31, 32, 39].

5.1 Fundamental entities

Schemes discussed here use standard data structures, such as a doubly linked list, to store lists of **primary entities** corresponding to cells and organized by dimension (list of vertices, lists of edges...). The geometric information describing the supporting manifolds, such as the equation of a surface that supports a face or of a curve that supports an edge, are accessible from these primary entities.

Grouping entities (such as loops, shells, vertex cones) are sometimes introduced for grouping and ordering primary entities. These grouping entities improve the performance of application algorithms by facilitating the traversal of the cells. For example, certain commands in standard graphics libraries require that polygonal faces be represented a sorted list of vertices along a single loop.

Incidence entities are also used to combine incidence and ordering information in a more compact and regular data structure. Incidence entities are associations of two or three cells of consecutive dimension and having an incidence relation. For example, loops of edges may be represented by associating with each edge-face pair a “next edge” pointer. Because an edge is bounding two faces in manifold models, such a combined entity is sometimes referred to as “half-edge”.

5.2 Incidence orientation and neighborhoods

A great circle splits a spherical surface into two 2-cells (the two connected components of the difference between the sphere and the curve). Given an **orientation** of the circular curve and an orientation of the “outer” normal to the surface, we can define a **“left”** and a **“right”** side to the curve within the surface. Each one of the 2-cells (faces) is adjacent to a different side of the curve. Such a relative orientation is often used to dissociate connected components of a decomposition. We can for example speak of the face that lies “on the left” of the circle (provided that the orientations of the curve and surface are well defined). The relative orientation between a (k+1)-cell (for example a face) and one of its bounding a k-cells (for example a curve) is called a **neighborhood** in [33] and takes three possible values: “left”, “right”, or “both”. The latter value is used for internal boundaries, such as a small disk inside a large ball or a bridge-edge connecting two loops of a face. If the face-edge neighborhood between a face F and an edge E is “left” we will say that “F is incident on E from the left”.

When the boundary cell is not in the manifold containing the higher-dimensional cell, the orientation of the manifold cannot be used to define a relative left and right orientation, and it may not always be possible to define a neighborhood. For instance, consider a self-intersecting surface that exhibits a singular edge where four branches of the surface meet. There is no easily defined “left” of the edge, even if the curve supporting the edge is oriented and if the surface is oriented everywhere except at the self-intersection edge.

Branch numbers could be used to distinguish the surface branches in the neighborhood of the edge. The branches may be defined as the connected components of the portion of the surface (without the self-

intersection curve) that lies inside a sufficiently small tube along the edge. However, the identification of these branches may prove extremely difficult, even for implicit surfaces.

5.3 Circular ordering of incident geometries

Several types of topological ordering relations may be established between cells.

Points on a curve are implicitly ordered by the orientation of the curve. For closed curves, this ordering is cyclic, unless an artificial singularity is introduced (for example the starting and ending point of a parameterization of the curve or a vertex), which permits to treat closed curves as if they were open. (Here the terms “closed” and “open” are not used with the topological meaning defined above, but simply to distinguish curves that form a “closed” loop from curves that do not.) By extension, such an ordering is used for the vertices and edges in a loop bounding a face. A loop is a alternating succession of edges and vertices. The ordering may in fact be used to represent the loop, especially for polyhedral models, where the edges are implicitly defined in terms of their end-vertices.

In the plane or on a manifold surface, edges may be ordered around their common vertex. To be more precise, when the vertex is the starting and ending point of the same edge, the two branches of the edge leaving the vertex appear as distinct entries in the ordering

Consider a sufficiently small circle around the vertex and assume that each edge lies on an oriented curve. We can induce an ordering of the curves around their common vertex by storing the cyclic sequence in which the small circle cuts the curves and, for each curve entry in this sequence, storing a binary flag indicating whether the curve was oriented left-to-right or right-to-left, as seen by an observer traveling on the circle. (The terms left and right may be precisely defined by an orientation of the supporting manifold surface. The circle must be sufficiently small such that there is only one point of contact between the curves in the disc enclosed by the circle.)

The geometric calculations involved in the computation of this ordering may prove very complex and numerically instable. For instance, ordering conic sections around a common point may require computing higher order derivatives, because tangent and curvature values at the contact point are insufficient. (An ellipse and a circle may be tangent to each other and exhibit the same curvature at the contact point.)

The order of faces around an edge, sometimes used to established what is called the “edge neighborhood” is very similar in nature to the order of edges around a vertex. However, the ordering of surfaces around a common intersection curve may change as one travels along the curve. The points where such an ordering changes correspond to zero-dimensional singularities which split the curve into cells (edges) of constant surface-ordering. Except for simple geometries, such as planar surfaces, the numeric computation of the ordering of surface branches around an edge-cell is far more delicate than ordering edges around a vertex.

5.4 Inclusion ordering

A sets of lower-dimensional cells may separate a manifold into two components. For example, a **loop** of edges and vertices may separate a plane into two parts (the interior and the exterior). Similarly, a **shell** of faces, edges, and vertices may separate the Euclidean 3-space into an interior and and exterior parts. Finally, a **vertex cone** of faces and edges may separate the neighborhood of a vertex into two parts. (The neighborhood of a vertex may be viewed as a sufficiently small ball around the vertex, not including the vertex itself.)

A set of nested loops separates a surface into several faces (connected open cells). Each one of these faces is incident on one or more loops and each loop separates exactly two adjacent faces incident upon the loop. A **face-adjacency graph** whose nodes correspond to faces and whose arcs correspond to loops separating adjacent faces is sufficient for capturing the nesting. When the graph is acyclic (i.e. when loops are

mutually disjoint), the graph may be represented as a **face-adjacency tree** by simply picking a root face and by propagating arc orientations away from the root.

In the plane, when loops are bounded, each loop separates the plane into a bounded interior and an unbounded exterior. The loops may then be ordered by saying that a loop A lies inside a loop B if A is contained in the interior side of B. The ordering may be captured in a **loop nesting tree** having the outer loop as root (i.e. the loop that is not contained inside any other loop). Nodes of the nesting tree are loops. The children of a loop L are all the other loops bounding the interior face of L. The nesting tree may be directly derived from the face-adjacency tree when the root face has only one bounding loop.

On a closed surface (a sphere for example), there is no a priori outer loop and in fact any face may be chosen as the root, yielding a different tree each time. It is thus preferable to use the face adjacency graph for capturing the acyclic nature of the partial ordering of loops.

Sets of shells and sets of vertex cones may be ordered in the same way as loops by replacing the nodes of the face adjacency graph by volume nodes that refer to 3-D cells (for ordering shells) or by solid cones of a vertex neighborhood (for ordering vertex cones).

Lower-dimensional elements (isolated vertices in the boundary of a face; dangling faces, edges, or vertices in the boundary of a volume; and dangling edges emanating from a vertex) may also be ordered using the face adjacency graph, the solid adjacency graph, or the vertex cone adjacency graph.

Note that since the connected components of a surface may be represented as genuine cells, the corresponding face adjacency graph is readily imbedded in the more general face-edge-vertex incidence graph, as discussed in the next section. Similarly, the shell nesting is directly available from the general volume-face-edge-vertex incidence graph. On the other hand, the vertex-cone nesting is not explicitly stored in a general adjacency graph since there may not be a unique cell corresponding to each vertex-cone.

5.5 Notation

We use the symbols V, E, F, and R to denote the (vertex, edge, face, and solid region) types of primary entity nodes in the incidence graph. The additional grouping entities for loops and shells are denoted L and S. Incidence types will be written using the concatenation of the letters of the basic types in lower case listed in decreasing order of dimension. For example, the type fe denotes all the associations between faces and their bounding edges.

Arrows indicate incidence relations (or their inverse) and may carry their cardinality (i.e. the number of referenced elements). Simple arrows denote variable number of incident elements. For example, $F \rightarrow E$ implies that faces point to a variable number of edges. A superscript over the arrow indicates the number of these references when it is constant. For example, $E \overset{2}{\rightarrow} V$ indicates that each edge points to exactly two vertices.

Sometimes, incidence references are organized by couples. For example, a face may have one reference to its bounding face-edge couples. We indicate such coupling with parentheses, as in $F \rightarrow (E,V)$.

Many data structures associate to each node of a particular type one or several pointers to other adjacent nodes of the same type. For example, consider pointers from an edge E to a subset of its neighboring edges. The number of such pointers is in general not constant. If we need only one such pointer per face, we can use the notation: $E \overset{F}{\rightarrow} E$. Extending the superscript notation even further, $E \overset{F \times V}{\rightarrow} E$, indicates that from each edge E there are pointers to other edges, one for each face-vertex pair such that the vertex is bounding E and the face is bounded by E.

When the multiple arcs emanating from a node are ordered (possibly in cyclic fashion), we use a double arrow "$\Rightarrow$" instead of "$\rightarrow$". For example, $F \Rightarrow V$ indicates that to each face is associated a list of links to

vertices that are ordered in a circular fashion around the face. (This ordering is often used for simply-connected faces.) For a face with several loops, we write: $F \rightarrow L \Rightarrow V$, ignoring, for simplicity, the fact that nested loops may also be partially ordered.

A entire graph will be described by a syntax that first lists all the node-types used in the graph and then all the types of arcs between these nodes. It is illustrated by the following example:

$$\{R, F, L, V \ : \ R \rightarrow F \rightarrow L \Rightarrow (F,V)\},$$

which indicates that the graph has nodes of type R, F, L, and V, and has a variable number of links from R to F and from F to L. It also has a variable number of link-pairs from L to V and to F that are ordered. (Regions are defined by a variable number of faces; each face is defined by a variable number of loops; each entry in the loop is a double reference to a vertex and to another face.)

5.6 Manifold models

We describe in this sub-section several data-structures for representing manifold models of polyhedra and of curved solids.

5.6.1 Face-vertex structure

For polyhedral models with simply connected faces, the edges are defined implicitly in terms of vertices and are thus not necessary (provided that vertices be ordered along loops). For faces without holes we may use an incidence graph based on face-vertex adjacency:

$$\{R,F,V \ : \ R \rightarrow F \Rightarrow V\},$$

and for multiply connected faces:

$$\{R,F,L,V \ : \ R \rightarrow F \rightarrow L \Rightarrow V\}.$$

5.6.2 Delta

The face-vertex structure may be enhanced with vertex-edge and edge-face back pointers to improve boundary traversal at a small storage cost [1]:

$$\Delta = \{R,V,E,F \ : \ R \rightarrow F \rightarrow V \rightarrow E \overset{2}{\rightarrow} F\} \quad \text{or} \quad \text{reverse-}\Delta = \{V,E,F \ : \ F \rightarrow E \overset{2}{\rightarrow} V \rightarrow F\},$$

and further extended for representing 3D triangulations:

$$\text{3D-}\Delta = \{V,E,F,R \ : \ R \overset{4}{\rightarrow} F \overset{3}{\rightarrow} E \overset{2}{\rightarrow} V \rightarrow R\}.$$

5.6.3 Edge-centered structure

Using edges as the stem of the representation, [40] proposes a different structure targeted at an optimal compromise between space and time efficiency:

$$\{F,E,V \ : \ V \Rightarrow E \overset{2}{\Rightarrow} (V,F) \ , \ F \Rightarrow E\}$$

Each edge points to a next edge around each abutting face. A face points to an ordered list of edges. This data structure only captures manifold topologies with simply-connected faces.

5.6.4 Winged-edge

The pioneering winged-edge representation developed by Baumgart [5, 6] is a bi-directional incidence graph to which ordering information is added as links between edges. The graph can only represent orientable manifold shells. Each edge-node points to four other edge-nodes that share with it a vertex and a face.

$$\text{winged-edge} = \{F,E,V : F \rightarrow E \overset{2}{\Rightarrow} V,\ E \overset{4}{\Rightarrow} E,\ V \rightarrow E \overset{2}{\Rightarrow} (F, E)\}.$$

The winged-edge data structure was extended by adding loops and shells ordering information [11]. In the late seventies a version of it was used by Braid, Eastman, Weiler, and Henrion in the development of GLIDE and another by Braid, Hillyard and Stroud in Cambridge, UK, in the development of BUILD, which later evolved into the ROMULUS system. A comparative analysis of the space and time costs associated with the different data structures for these extensions may be found in [1, 39].

5.6.5 FAHs

A FAH (Face-Adjacency Hypergraph) was used for modeling two-manifold boundaries [2]. The arcs of a FAH define face-face adjacency and simply correspond to edges. Hyperarcs are connecting a vertex to all of its edges. Thus, using E as a symbol for an arc, we have:

$$\text{FAH} = \{F,E,L,V : F \rightarrow L \Rightarrow (E,V),\ E \overset{2}{\Rightarrow} F,\ V \Rightarrow E\}$$

FAH's were subsequently extended for modeling objects at multiple levels of details, reflecting an iterative design process of incrementally adding features and details [14].

5.6.6 Half-edge

Since, in manifold shells, an edge is bounding two faces, it may be convenient to use two fe-nodes to represent each edge. To each fe-node corresponds a different orientation of the edge and is associated one of the two vertices that bound the edge. These fe-nodes have been used in many data structures and have been called “split-edges”, “half-edges”, “edge-uses”, and so on. These half-edges are usually linked to each other half-edges, either directly or through an edge-node so as to capture face-face adjacency [19, 24].

As shown in [1] half-edge data structures correspond to:

$$\text{Half-edge} = \{R,F,L,fe,E,V : R \rightarrow F \rightarrow L \rightarrow fe \overset{1}{\rightarrow} (V,E),\ E \overset{2}{\rightarrow} fe,\ V \overset{1}{\rightarrow} fe \overset{1}{\rightarrow} L \overset{1}{\rightarrow} F \overset{1}{\rightarrow} R\},$$

plus redundant pointers from R to all the L, V, E, and S nodes. The half-edge structure was used by Mäntylä and Sulonen in the GWB system.

5.6.7 Quad-edges

The winged-edge representation was extended by Guibas and Stolfi to subdivisions of orientable surfaces using a quad-edge data structure [15]. Simple primitive operators were provided to move from edge to edge around face loops and around vertices. Each edge refers to four of its neighbors.

5.6.8 Cell-tuples and V-maps

Brisson [7, 8] uses cell-tuples to extend the face-edge data structure [10, 21] and the quad-edge data structures [15] to higher dimensions. A cell-tuple is a combination of cells of all the dimensions, such that each cell (except the full-dimensional one) is in the boundary of the cell of the next dimension in the cell-tuple. For example, in 3D a cell tuple is defined by selecting a region, one of its faces, one of the

edges bounding the face, and a vertex bounding that edge. The switch(k) operator parameterized by the dimension k produces the other tuple that has the same elements, except for the element of dimension k, which is uniquely defined. For example, switch(0) exchanges the two vertices of the edge and switch(1) exchanges the two edges that bound the face and share the vertex. switch is its own inverse. A permutation of switch operators for dimension k and k+1. may be used to order k-cells and (k+1)-cells around (k—1)-cells on a (k+2)-cells. For example, an alternation of switch(1) and switch(2) may be used to visit the edges and faces of the cone of a shell formed around a vertex. Independently, Lienhardt [22, 23] defines n-dimensional generalized maps. For manifold objects, both Lienhardt's and Brisson's representations are equivalent.

5.7 Non-manifold structures

A technique for extending boundary graphs to non-manifold cases, where the solids have internal structures is based on the use of 3D region nodes, R_i, in the delimitation graphs. Each 3D regions is associated with a well defined subset of the boundary that forms a valid shell, or set of shells. (Typically, for simplicity, the solids are restricted to be connected, although not necessarily simply connected. More than one shell may be needed when the solids have internal holes.)

5.7.1 Edge-Less Adjacency Graph

The lack of face-face adjacency information in the simple face-vertex structure may be overcome by extending it into an ELAG (Edge-Less Adjacency Graph) for polyhedra.

In an ELAG, each pair of consecutive vertices in a loop define an edge. (The loop implies a circular ordering and thus the last vertex is followed by the first one. If the loop is non-manifold, several entries in the loop may refer to the same vertex.) We can arbitrarily associate this edge with the first one of the two vertex-entries, in the order of their appearance in the loop. Thus each vertex in a loop implicitly defines a face-edge pair, i.e. an element of type fe. Given a face F and an edge E in the boundary of a solid region R, the pair fe unambiguously defines at most two faces F_1 and F_2, that have E in their boundary and that are adjacent to F in the circular ordering around E of all the faces of R bounded by E. Thus, one can associate two face-pointers with each vertex-entry in each loop. We obtain the following specification:

$$\text{ELAG} = \{R,F,L,V \ : \ R \rightarrow F \rightarrow L \Rightarrow (V, 2F)\},$$

where the notation “$L \Rightarrow (V, 2F)$” indicates that each loop has a variable number of entries, each pointing to one vertex and two faces.

Given the orientations of the faces and of the edges and their neighborhood information with respect to R, we can add to the loop-face links neighborhood information that will enable us to traverse the boundary of a region R by walking from one face to the next in such a manner that the sector specified by these two faces in the vicinity of the edge is inside R and is not intersected by any other face adjacent to E.

Note that for manifold boundaries $F_1 = F_2$, and only one pointer is necessary:

$$\text{Manifold ELAG} = \{R,F,L,V \ : \ R \rightarrow F \rightarrow L \Rightarrow (V, F)\}.$$

When, in addition, faces are simply connected, we can merge the face-nodes with the loop-nodes altogether:

$$\text{Manifold ELAG with simply connected faces} = \{R,F,L,V \ : \ R \rightarrow F \Rightarrow (V, F)\}.$$

When restricted to manifolds shells, the ELAG concept was used in [26] for representing 2D triangulations by associating with each triangular face three pointer-pairs:

$$\text{Triangulation} = \{F,E : F \overset{3}{\Rightarrow} (V,F)\}.$$

The concept was further extended in [9, 13] to higher dimensional triangulations.

5.7.2 Facet-edges

Dobkin and Laszlo extend the approach of [15] to a subdivision of E^3. They define a facet-edge data structure [10, 21] in which each face F points to other adjacent faces that bound the two regions bounded by F.

The two-cycle requirement for shells of regularized solids, does not allow the use of “non-manifold” boundaries in the larger sense of the word, i.e., boundaries that are not two-cycles, because they have dangling faces or edges or because they define a partitioning of the solid into several connected regions. Internal structures may be specified by superimposing on the solid the internal dangling faces and edges.

5.7.3 Half-edges and hybrid-edges

Mantyla's Half-Edge and Kalay's Hybrid-Edge [19, 24] data structures may be used to represent dangling faces together with shells of 3D regions, and are thus useful for extending the topological domain beyond dimensionally homogeneous sets, and even to a limited class of non-manifold boundaries.

5.7.4 Radial-edge

In the winged-edge representation, to each face-edge-vertex and each edge-vertex incidence link is associated a link to a face. Thus, the winged-edge data structure has implicit $ev \overset{1}{\rightarrow} E$ and $fe \overset{1}{\rightarrow} E$ links. Using these auxiliary fe and ev entities as nodes in the graph, Weiler has defined the vertex-edge and the face-edge data structures [37], leading to the radial-edge data structure [38].

5.7.5 Vertex-based structure

The radial-edge data structure explicitly captures how faces are ordered around an edge and how edges are ordered around a face. It does not however provide any information on the vertex-cone nesting, which is important for a consistent traversal of the object's boundary at non-manifold vertices and is addressed in the NOODLES system by Gursoz, Choi, and Prinz [16, 17].

5.7.6 SGCs with NAILs

In a Selective Geometric Complex, introduced by Rossignac and O'Connor, each cell points to all of its bounding and incident cells. When the dimension of two incident cells differs by exactly one, and the lower-dimensional cell is in the manifold supporting the higher-dimensional one, the link is augmented with the (left, right, or both) neighborhood. Cells are tagged as active or inactive.

In addition to the incidence graph used for SGCs, to each edge and to each face node one can associate a two-dimensional table called “NAIL” (for Next cell Around a cell In a cell List). For an edge e, the table is indexed by a reference to a vertex v bounding e and by a reference to a face f incident upon the edge. The corresponding entry e.NAIL(v,f) in the table contains a reference to the next edge around v in f. Where “around” is defined with respect to the orientation of the surface supporting f. For a face f, the table is indexed by a reference to an edge e bounding f and by a reference to a 3-cell r incident upon f. The corresponding entry f.NAIL(e,r) in the table contains a reference to the next face around e bounding r.

6.0 CONSTRUCTIVE NON-REGULARIZED REPRESENTATIONS

A Constructive Solid Geometry (CSG) representation defines a recipe for a solid as a selection of 3-D cells from a decomposition of space induced by the CSG primitives (half-spaces or volume primitives). The operations used to control the selection are the regularized Boolean union, intersection, and difference. A regularized operation returns the closure of the interior of its set theoretic counterpart [27, 28, 36]. Note that, if the arguments are regular, then the result of a set-theoretic union is identical to the result of the regularized union. Furthermore, if the primitives are regular, then the result of an expression involving regularized Booleans is identical to the closure of the interior of the same expression composed of the corresponding set-theoretic operators.

Using this recipe as a fundamental representation instead of a boundary representation has many advantages. The non-evaluated (CSG) representation is always valid and can be easily parameterized. Editing a non-evaluated representation is simple and very efficient; it suffices to change the expression. Non-evaluated representations are less verbose than their evaluated counterparts and lead to considerable storage savings. Finally, many solid modeling algorithms work directly on CSG representations through divide-and-conquer and are numerically more reliable than their counterparts that work on evaluated boundary representations. We discuss in this section several variations of the traditional CSG representation models.

A representation scheme that covers a rich set of inhomogeneous geometric objects and operations was proposed in [34]. It is called Constructive Non-Regularized Geometry (CNRG). CNRG trees represent objects that are aggregates (i.e., unions) of mutually disjoint regions. Each region is a set in $\mathbf{R}^n$ and needs not be connected, regular, or even dimensionally homogeneous. The leaves of a CNRG tree correspond to parameterized primitive shapes such as volumes, faces, curve segments, or points. Internal nodes correspond to intermediate CNRG objects and are associated with topological and Boolean operations.

SGCs provide a model for representing non-regularized internally-segmented sets as a collection of connected open cells defined recursively in terms of their boundaries. CNRG trees yield an alternate representation for these sets as a collection of regions defined in terms of original primitive sets. Regions of CNRG objects need not be open nor connected and typically correspond to unions of SGC cells of various dimensions. CNRG models should be viewed as a primary model for user interaction because they support a high level vocabulary for expressing operations and regions, and because the CNRG trees are, similarly to CSG trees, easy to edit and archive. SGC models should be automatically derived from CNRG representations, as boundary representations are derived from CSG trees.

The scheme proposed in [34] extends CSG in several ways:

1. It uses standard (non-regularized) Boolean operations and topological operations—boundary, closure and interior. The regularized Boolean operations can be implemented in this scheme as three-operator sequences: standard Boolean, followed by interior and closure.
2. It admits as primitives non-solid objects such as points, curves, or surfaces, and higher-dimensional objects.
3. It introduces a new operator, called aggregation, which constructs structured objects composed of several regions. The aggregation operator is a formally-defined and more sophisticated version of the “assembly” operator provided by modelers such as PADL-2. Structured objects are not sets. They are collections of sets, much like the cell complexes of algebraic topology [29]. Structured objects, also called CNRG objects, or simply objects, also have underlying sets, as cell complexes do. The underlying set of a CNRG object is the union of all the regions of the object, and therefore it is a set with no structure or “internal boundaries”.
4. It defines Boolean and topological operations on structured objects.

6.1 Semantics of CNRG operators

A CNRG object is a set of pairwise disjoint possibly non-regular or disconnected sets, called **regions.**

The **set,** pA, of a CNRG object, A, is the union of the sets of its regions.

A CNRG tree is a rooted directed acyclic graph that represents a CNRG object. Leaves of the tree are CNRG primitives, which may be composed of more than one region. For simplicity, we assume that each primitive is given in its absolute position, although in practice, rigid body transformation nodes may be used. Internal nodes represent intermediate CNRG objects obtained by applying Boolean, topological, simplification, or filtering operators to the CNRG objects represented by their child-nodes.

To simplify the following discussion, we use "|" to denote a "gluing" or aggregation binary operator that takes two disjoint regions, aggregates of regions (all of which are pairwise disjoint), or disjoint CNRG objects and produces a CNRG object that aggregates all the regions. Note that expressions involving only aggregate operators are order-independent.

An uppercase letter denotes a CNRG object, and each of the object's regions is denoted by the same letter with a subscript. For example, A_1 and A_2 are two regions of the same CNRG object, A. If A is composed of only these two regions, we write, $A = \{A_1|A_2\}$. By definition, A_i and A_j are disjoint for $i \neq j$. A single-region object, A, is considered distinct from its region A_1. We write $A = \{A_1\}$. Note that $A_1 = pA$ for single region objects.

In the following, we assume that S, A, B, C and D are CNRG objects. Furthermore, we often use C or S to denote the objects produced by applying to A (and to B) a Boolean or topological CNRG operator.

A and B are **disjoint,** if and only if the intersection of their sets, pA and pB, is empty.

Given two regions, A_i and B_j, $A_i \cap B_j$ denotes their intersection and $A_i - B_j$ their difference in the standard set-theoretical sense. Note that $A_i \cap B_j$ and $A_i - B_j$ are single regions that may be empty, disconnected, dimensionally inhomogeneous, and not closed.

A region is said to be **contained** in an object if it is contained in the set of the object. It need not correspond to the union of any subset of the object's regions.

Given n disjoint sets, A_i, the **aggregation** operation, denoted "|", creates the corresponding CNRG object: $A = \{A_1 | A_2 | \ldots | A_n\}$.

The **simplification,** sA, of A is a CNRG object with a single region: the set pA. Thus, $sA = \{pA\}$.

The **complement,** cA, of A is an object composed of a single region that is the set complement of pA. Hence, $cA = \{\overline{pA}\}$.

The **union,** A + B, is an aggregate of regions of the following three types: $A_i \cap B_j$, $A_i - pB$, and $B_j - pA$, for all combinations of regions, A_i, of A and regions, B_j, of B. Potentially each region of A is split into two sets: the part in B and the part outside of B. The second set is a single region. The first set may be decomposed according to the decomposition of B into regions. Union produces a subdivision of $(pA) \cap (pB)$ that is compatible with the decomposition of A and B into regions. We call it "union" because the set p(A + B) equals the set theoretic union, $pA \cup pB$.

The **intersection,** A*B, of A and B is the aggregate of all regions of the type $A_i \cap B_j$. A region, A_i, of A is truncated to $A_i \cap pB$, and is subdivided according to the subdivision of B into regions. Consequently, $p(A*B) = (pA) \cap (pB)$, which justifies the name for this operator.

The **difference,** A\B, is the aggregate of all regions of the type $A_i - pB$. Clearly, $p(A\backslash B) = (pA) - (pB)$. It follows from the definition of union, intersection, and difference that $A + B = \{(A\backslash B) | (A*B) | (B\backslash A)\}$.

The topological **interior,** iA, of A in $\mathbf{R}^n$ is the aggregate of regions, $A_i \cap$ interior(pA), that are the intersection of the regions of A with the topological interior of A in $\mathbf{R}^n$. Note that, although iA is always full-dimensional, regions of iA need not be full-dimensional. Thus the *interior* operator returns a subset of the original object and preserves its internal decomposition into regions. The set, piA, of the interior of A equals the topological interior of pA.

The **closure,** kA, of A, is the aggregate composed of all the regions of A plus a single new region defined as the difference between the topological closure of pA and pA itself. Thus, we can write: $k\{A_1, ..., A_n\} = \{A_1, ..., A_n, | (closure(pA) - pA)\}$, and pkA equals the topological closure of pA.

The topological **boundary,** ∂A, of A is defined as: ∂A = kA\iA. The boundary operator does not necessarily return a single-region object. Note that p(∂A) equals the topological boundary of pA.

The **regularized** version, rA, of A is defined as kiA. The set, prA, spanned by rA corresponds to a regularized solid as defined in [27] and equals rpA. Note that regularization does not imply simplification (i.e., a regularized object may be composed of many non-regularized regions). However, taking the boundary, C, of a regularized object, A, will produce a lower-dimensional object whose set, pC, is the topological boundary of the set of the regularized object, prA.

6.2 Evaluation

CNRG graphs may be evaluated directly or converted into an expanded SGC form, where SGC cells are grouped to form sets that represent CNRG regions.

Point inclusion test for a particular CNRG region may be carried out using an extension of the divide-and-conquer techniques popular with CSG trees (see [34] for details). On the other hand, it may be advantageous to compute and store the SGC representation of the non-manifold structure defined by a CNRG expression. The computation may be simply carried out as an incremental (bottom-up) execution of the CNRG operations.

Each one of these operations may be constructed from combinations of the primitive SGC operations:

- Subdivision, which takes two SGCs and adds to the boundary of each cell of each SGC its lower-dimensional intersections with cells of the other SGC. This addition may result in splitting the cell into separate connected components.
- Selection, which activates or deactivates cells based logical predicated involving topological, geometric, connectivity filters and inclusion in specific regions, and which also assigns cells to the regions of the resulting SGC.
- Simplification, which simplifies the representation of each region by deleting or merging its cells.

The algorithms for these operations have been described in more detail in [33] without limitations to any particular geometric domain.

7.0 CONCLUSION

These notes present the key concepts for analyzing the topological limitations of geometric representation schemes independently of the geometric domain. They also overview a number popular data structures for manifold and non-manifold boundary representations and an extension of CSG representations to non-regularized sets. They promote CNRG expressions for designing and editing non-manifold geometric structures and SGC representations and associated algorithms for computing and storing a boundary representation of these structures.

8.0 APPENDIX: REVIEW OF KEY TOPOLOGICAL CONCEPTS

This appendix provides an informal summary of the key topological notions relevant to this notes. Formal and complete definitions may be found in many textbooks on Algebraic and Combinatorial Topology ([18, 25, 35]).

A **topological space** is a set W with a choice of a class of subsets of W (its **open sets**), each of which is called a **neighborhood** of its points, such that every point of W is in some neighborhood and that the intersection of any two neighborhoods of a point contains a neighborhood of that point. The three-dimensional Euclidean space, in which the models discussed here are constructed, is so "topologized".

The **interior** of a set S is the set of points having a neighborhood in S. (Intuitively the interior of a 3-D set is is the whole set except for points on its surface, or more precisely on its boundary, defined later.). A set is **open** if it contains a neighborhood for each one of its points.

The **complement** of a set S in R is the set of points of R that are not in S. The **closure** of S is the complement of the interior of S. (Intuitively, the closure of a set 3-D includes the bounding surfaces, whether they were part of the original set or not.) S is **closed** if its complement is open. (Intuitively, a 3-D set is closed if it includes its surface.)

The **boundary** of S is the difference between its closure and its interior. (Intuitively, for a 3D set, it is the surface. However, if S is the result of subtracting the center point from a ball, then the boundary of S is not only the spherical surface, but also the missing point.)

Two sets are **homeomorphic** if one is the image of the other through a bijective map that is continuous and has a continuous inverse. (Intuitively, we can map each point of any one of these two sets into a unique point of the other set in such a way that they have topologically identical neighborhoods. It does not necessarily mean that we can deform one object in a continuous manner to produce the other object.)

An **open k-ball** of radius r around a point s in a k-dimensional Euclidean space is the set of points at a distance less than r from s, where the distance between two points is defined as the Euclidean norm of the vector separating the two points.

The **dimension** of a set is the minimum dimension of the topological spaces containing the set. A set is **full-dimensional** (with respect to some topological set) if its interior is not empty. An open set is thus always full-dimensional. In a topological space of dimension n, a set of dimension k lower than n is **relatively open** if it is homeomorphic to an open k-ball as a subset of some topological space of dimension k. (A **relative topology** for a set S as a subset of a topological space W may be inherited from W by considering as the open sets of S as the intersections of S with the open sets of W.) For example, a face F has no interior in three-space. However, the **relative interior** of F, defined as F without its bounding edges and vertices is relatively open. This "relativity" concept may be also applied to the boundary: the **relative boundary** of F is its bounding edges and vertices (i.e., the boundary of F in the relative topology of the two-manifold, or surface, supporting it).

A set is **regular** if it is equal to the closure of its interior. A regular set is thus closed and does not contain boundary elements that do not have in their neighborhood any interior point of the set. A set is **s-regular** if it is equal to the interior of its closure. An s-regular set is therefore open and does not contain cracks or lower-dimensional holes. The complement of an s-regular set is regular. S-regular sets were used in [3] for modeling assemblies of sets that share faces.

Two sets are **disjoint** if their intersection is empty, i.e.: if no point belongs to both sets. A set is **connected** if any two of its points may be joined by a continuous curve inside the set. A connected set cannot be divided into two sets, such that the closure of one be disjoint from the other. The (maximally) **connected components** of a set are uniquely defined. A set is **interior-connected** if its interior is connected. Two sets are **quasi-disjoint** if their intersection is not empty, but is not full-dimensional. (For example, two cubes touching at a vertex for a connected set that is not interior-connected. The two cubes are quasi-disjoint.)

A **closed k-ball** of strictly positive radius r around a point s is the set of points at a distance from s less or equal to r. A **k-sphere** is the boundary of the corresponding k-ball. A **k-half-ball** is the intersection of a k-ball with a planar half-space containing s. A **half-space** is a full-dimensional set of points usually

assumed to be connected and regular, and defined by an algebraic or analytic inequality. (A strict inequality defines an open half-space.) The set of points where the first coordinate is positive or null is a good example of closed a planar half-space through the origin.

A **k-manifold** is a set of points whose neighborhoods are homeomorphic to an open k-ball. A **k-manifold with boundary** is a set of points whose neighborhoods are homeomorphic to an open k-ball or to a half-k-ball.

An open set is **simply connected** if it is homeomorphic to an open ball.

A **cavity** in a bounded (i.e. non infinite) set S is a bounded connected component of the complement of S. In two dimensions, cavities correspond to the intuitive notion of holes. In three dimensions, the term **hole** is ill-defined. We prefer to use the term cavity (such as the one found inside a soccer ball) and the term **handle,** which denotes a tunnel or "way through" the set (such as the handle of a tea cup). The number of holes through a set is important for assessing whether two sets are homeomorphic, but does not reflect how they are embedded in in three-space.

The **zero Betti number,** b_0, denotes the number of connected components in a set. The **first Betti number,** b_1, (also called 1-connectivity) specifies the number of handles in a 3D set. It may be defined as the maximum number of "cuts" through the set that may be made without disconnecting it (i.e. producing two separate pieces). A cut through a set may be thought of as the surface swept by drawing a closed curve on the boundary of the solid and contracting it to a single point while maintaining it inside the set. For example the 1-connectivity is 0 for a ball, and 1 for a solid torus, and 2 for the surface of a torus (the zero Betti number for a closed surface is twice the zero Betti number for the solid bounded by the surface). The **second Betti number,** b_2, denotes the number of cavities.

The **genus** of a surface is the maximum number of closed curves (contained in it) that may be subtracted from it without disconnecting it. The genus, also denotes the number of handles, H. One closed surface may be mapped into another by a continuous bijection if they have the same genus. The genus of a closed surface is half its first Betti number and is also equal to the Betti number of the 3D set bounded by the surface.

A **k-simplex** is the convex hull of k+1 linearly independent points in R. An **m-face** of a k-simplex is the convex hull spanned by m of the k points of the k-simplex and is an m-simplex. All k-simplices are closed and homeomorphic to a closed k-ball. The boundary of a k-simplex is homeomorphic to a k-sphere. A **simplicial complex** is a finite union of simplices glued together such that for any pair (A,B) of these simplices: either A and B are disjoint, or A and B share a common m-face, or A is an m-face of B, or B is an m-face of A (for some m). The **polytope** of a simplicial complex is the union of the sets of all of its simplices.

A **CW complex** is a finite union of mutually disjoint relatively open cells, each being homeomorphic to an open ball and having for boundary the union of the sets of other cells in the complex. The intersection of the closure of two cells is either empty or is the union of other cells in the complex. CW complexes generalize the notion of simplicial complexes because their cells are not restricted to simplices (i.e. points, line segments, triangles, and tetrahedra), but may include relatively open sets of arbitrary shape and of an arbitrary finite number of bounding simplices (k-faces), provided that the 2-D cells have no holes and that the 3-D cells have no handles.

Two distinct k-cells of a (simplicial, CW, or geometric) complex are **adjacent** if they share one or more bounding cells. A (k+1)-cell c is **incident** on a k-cell b if b is a bounding cell of c.

For a two-dimensional two-manifold closed surface without boundary made of F faces, E edges, and V vertices, the **Euler characteristic** (or Euler number) is equal to $V - E + F$.

For a 3-D manifold CW complex (a set bounded by a two-manifold surface) made of R 3-D cells, F faces, E edges, V vertices, the Euler characteristic is a topological invariant independent of the subdivision subdivision and is equal to $V - E + F - R$. The **Euler equation** states that the Euler characteristic is equal to $b_0 - b_1 + b_2$. Since $b_0 + b_2$, the number of connected components plus the number of cavities is the number of shells, S, in the surface bounding the 3D set, we have for a 3D set:

$$V - E + F - R = S - H,$$

where H is the number of handles through the entire 3-D set.

The Euler characteristic of a surface in 3-D is twice the Euler characteristic of the solid bounded by the surface. Since b_0, the number of shells is equal to the number of cavities, b_2, and since the maximum number of non-separating cuts in the surface is twice the number of handles ($b_1 = 2H$), the Euler equation for a surface is:

$$V - E + F = 2(S - H).$$

Since V, E, F, R are readily available and S may be easily computed by constructing connected components, the above formulae yield a practical means for computing H! The formulae are restricted to manifold sets, although they can be extended to non-manifold CW complexes by incorporating the counts of various on-manifold situations, such as the additional cones of faces incident upon a vertex.

We can verify the Euler equation for a solid torus represented as a CW complex. We need to introduce a vertex and two curves that will split the surface of the torus into a single face whose relative interior is homeomorphic to an open disk (2-D ball). Furthermore, we need to introduce a cut-face through the interior of the torus, so that the 3-D interior be homeomorphic to an open ball. The Euler equation for the resulting complex has: $V = 1$, $E = 2$, $F = 2$, $e = 0$, $R = 1$, $f = 0$, $S = 1$, $H = 1$.

The popular geometric primitives, such as cylinders or cones, cannot be represented directly as simplicial complexes nor even as CW complexes if their faces, edges, and vertices are not simply connected. Artificial bridge-edges and cut-faces may have to be introduced. **Geometric Complexes** [33] generalize the concept of CW complexes allowing cells to be open sets of arbitrary genus (rather than being restricted to be homeomorphic to open balls). For example, a torus may be represented as a geometric complex by only two cells: its 3-D interior and its 2-D boundary.

Simplicial complexes, CW complexes, and Geometric Complexes are closed, i.e.: they contain the boundaries of all of their cells. A **Selective Geometric Complex** (also called SGC) developed in [33] further extends the notion of a geometric complex by associating with each cell an attribute stating whether the cell is **active** (i.e. contributes to the final set) or not. For example, an open sphere without its center point may be modeled by an SGC with three cells: the 3-D interior without the point, the 2-D bounding sphere, and the central vertex. Only the interior is active.

9.0 References

1. Ala, S. Universal Data Structure: A tool for the Design of Optimal Boundary Data Structures. In J. Rossignac and J. Turner, editors, *ACM/SIGGRAPH Sym. on Solid Modeling Foundations and CAD/CAM Applications ACM Symp. on Solid Modeling Foundations and CAD/CAM Applic.*, 13-24, ACM Press, Order number 429912, Austin, TX, June 5-7 1991.
2. Ansaldi, S., De Floriani, F. and Falcidieno, B. Geometric Modelling of Solid Objects by Using a Face Adjacency Graph Representation. *ACM Computer Graphics*, 19(3):131-139, July 1985.
3. Arbab, F. Set Models and Boolean Operations for Solids and Assemblies. Computer Science Dept., Univ. of S. Calif., Los Angeles, CA. July 1988.

4. Bardis, L. and Patrikalakis, N. Topological Structures for Generalized Boundary Representation. MIT, Design Laboratory Memo 91-18. 1992.
5. Baumgart, B.G. Winged Edge Polyhedron Representation. *AIM-79, Stanford Univ.*, Report STAN-CS-320, 1972.
6. Baumgart, B. A Polyhedron Representation for Computer Vision. *AFIPS Nat. Conf. Proc.*, 44:589-596, 1975.
7. Brisson, E. Representing Geometric Structures in D-Dimensions: Topology and Order. *Fifth ACM Symposium on Computational Geometry, Saarbruchen,*, 218-227, June 1989.
8. Brisson, E. *Representation of d-Dimensional Geometric Objects*, PhD thesis. Dept. of Compt. Sci. and Engr. University of Washington, Seattle, WA, 1990.
9. Cattani, C. and Paoluzzi, A. Solid Modeling in Any Dimension. In Dip. di Mathematica, Univer. "La Sapienza", Rome, Italy, editor, *Report*, Univ. 'La Sapienza, Rome, Italy, 1989.
10. Dobkin, D.P. and Laszlo, M.J. Primitives for the Manipulation of Three-Dimensional Subdivisions. *Third ACM Symp. on Computational Geometry, Waterloo, Canada*, 86-99, June 1987.
11. Eastman, C.M. and Weiler, K. Geometric Modelling Using the Euler Operators. *Proc. 1st Annual Conf. on Computer Graphics in CAD/CAM*, 248-259, 1979.
12. Ellis, J.L., Kedem, G., Lyerly, T.C., Thielman, D.G., Marisa, R.J. and Menon, J.P. The Ray Casting Engine and ray representations. In J. Rossignac and J. Turner, editors, *ACM Symp. on Solid Modeling Foundations and CAD/CAM Applic.*, 255-268, ACM Press, Order number 429912, Austin, TX, June 5-7 1991.
13. Ferrucci, V. and Paoluzzi, A. Extrusion and Boundary Evaluation for Multidimensional Polyhedra. *Computer-Aided Design*, 23(1):40-50, January/February 1991.
14. Floriani, L. and Falcidieno, B. A Hierarchical Boundary Model for Solid Object Representation. *ACM Trans. Graphics*, 7(1):42-60, 1988.
15. Guibas, L. and Stolfi, J. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. *ACM Trans. on Graphics*, 4(2):74-123, April 1985.
16. Gursoz, E. Levent and Prinz, F.B. Node-based Representation of Non-Manifold Surface Boundaries in Geometric Modeling. In M. Wozny, J. Turner and K. Preiss, editors, *Geometric Modeling for Product Engineering, Proc. of the*, North-Holland, 1989.
17. Gursoz, E., Choi, Y. and Prinz. F. Boolean Set operations on Non-Manifold Boundary Representation Objects. *Computer-Aided Design*, 23(1):33-39, January/February 1991.
18. Henle, M. *A Combinatorial Introdution to Topology*. W.H. Freeman and Co., San Francisco, 1979.
19. Kalay, Y.E. The Hybrid Edge: A Topological Data Structure for Vertically Integrated Geometric Modeling. *Computer-Aided Design*, 21(3): 130-140, 1989.
20. Kaufman, A. *Volume Visualization*. IEEE Computer Society Press, Los Alamitos, CA, 1991.
21. Laszlo, M.J. *A Data Structure for Manipulating Three-Dimensional Subdivisions*, PhD thesis. Princeton Univ., August 1987.
22. Lienhardt, L. Topological Models for Boundary Representation: A Comparison With N-dimensional Generalized Maps. *Computer-Aided Design*, 23(1):59-82, January/February 1991.
23. Lienhardt, P. Subdivision of N-Dimensional Spaces and N-Dimensional Generalized Maps. *ACM Sumposium on Computational Geometry*, 228-236, Saarbruecken, Germany, June 1989.
24. Mäntylä, Martti. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, Maryland, 1988.
25. Mendelson, B. *Introduction to Topology*. Volume 3rd ed. Allyyn and Bacon, Inc,, Boston, MA, 1975.
26. Paoluzzi, A., Ramella, M. and Santarelli, A. Boolean Algebra Over Linear Polyhedra. *CAD*, 21(8):474-484, 1989.
27. Requicha, Aristides A.G and Tilove, Robert B. Mathematical Foundations of Constructive Solid Geometry: General Topology of Regular Closed Sets. In Production Automation Project, editor, *Tech. Memo.*, No. 27a, Univ. of Rochester, June 1978.
28. Requicha, Aristides A.G. and Voelcker, Herbert B. Constructive Solid Geometry. *Production Automation Project, Univ. of Rochester*, Tech. Memo No.25, November 1977.
29. Requicha, A.A.G. Mathematical Models of Rigid Solid Objects. *Technical Memo*, No.28, Univ. of Rochester, November 1977.

30. Requicha, A.A.G. and Voelcker, H.B. Boolean Operations in Solid Modelling: Boundary Evaluation and Merging Algorithms. *Proc. IEEE*, 73(1):30-44, January 1985.
31. Rossignac, J. Representing Solids and Geometric Structures. In S. Kodiyalam, M. Saxena, editor, *Geometry and Optimization Techniques for Structural Design*, 1-44, Computational Mechanics Publications,, Southhampton, 1993.
32. Rossignac, J. Through the cracks of the solid modeling milestone. In S. Coquillart, W. Strasser, P. Stucki, editor, *From object modelling to advanced visualization*, 1-75, Springer Verlag, 1994. (State of the art report, Eurographics'91, Vienna)
33. Rossignac, J.R. and O'Connor, M.A. SGC: A Dimension-independent Model for Pointsets with Internal Structures and Incomplete Boundaries. In M. Wozny, J. Turner, K. Preiss, editor, *Geometric Modeling for Product Engineering*, 145-180, North-Holland, Rensselaerville, NY, September 1989.
34. Rossignac, J.R. and Requicha, A.R. Constructive Non-Regularized Geometry. *Computer-Aided Design, Special Issue: Beyond Solid Modeling*, 23(1):21-32, January/February 1991.
35. I. M. Singer and J. A. Thorpe. *Lecture Notes on Elementary Topology and Geometry*. Scott, Foresman, Glenview, IL, 1967.
36. Tilove, R.B. and Requicha, A.A.G. Closure of Boolean Operations on Geometric Entities. *Computer-Aided Design*, 12(5):219-220, September 1980.
37. Weiler, K. Edge-based Data structures for Solid Modeling in Curved-Surface Modeling Environments. *IEEE Computer Graphics and Applications*, 5(1):21-40, January 1985.
38. Weiler, K.J. The Radial Edge Structure: A Topological Representation for Non-Manifold Geometric Modeling. In M. Wozny, H. McLaughlin and J. Encarnacao, editors, *Geometric Modeling for CAD Applications*, Springer Verlag, May 1986.
39. Woo, T.C. A Combinatorial Analysis of Boundary Data Structure Schemata. *IEEE Computer Graphics and Applications*, 5(3):19-27, March 1985.
40. Woo, T.C. and Wolter, J.D. A Constant Expected Time, Linear Storage Data Structure for Representing Three-Dimensional Objects. *IEEE Trans. Systems, Man and Cybernetics*, SMC-14(3):510-515, May/June 1984.

Specification, representation, and construction of non-manifold geometric structures

Jarek Rossignac
IBM Research

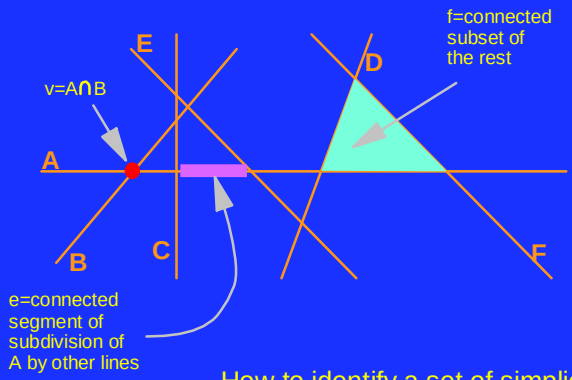
Context

- INPUT
  - Sample surface points
  - Space meshes (FEM, voxels)
  - Slices, ray-reps
  - CAD construction steps
  - Procedural models
- REPRESENTATION
  - Geometric primitives (surfaces, curves, points)
  - Topological relations (adjacency, boundary of)
  - Ordering (next vertex, edge, face, branch)
  - Pointsets and structures (non-manifold, features)

Content

- Linear decomposition
- Hierarchical specification
- Constructive Non-Regular Geometry
- Boundary representations
- Selective Geometric Complexes
- SGC operations
- Curved geometry

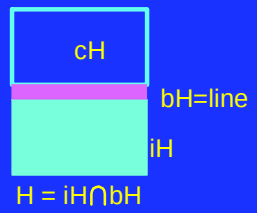
Lines decompose space into simplices



How to identify a set of simplices?

Use Boolean expressions

Half-spaces

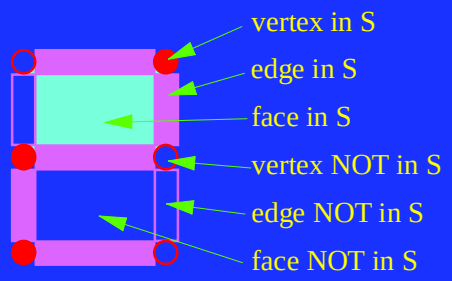


Operators

∪ = union
∩ = intersection
− = difference
b = boundary
i = interior
c = complement
k = closure
0 = zero cells
1 = one cells...

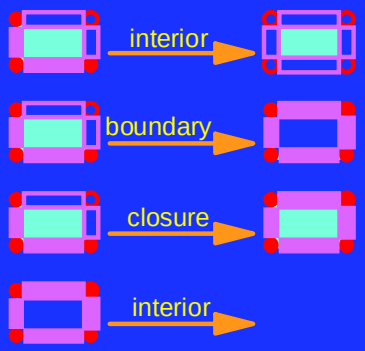
Two operators suffice,∩ and c, but all are convenient

Convention

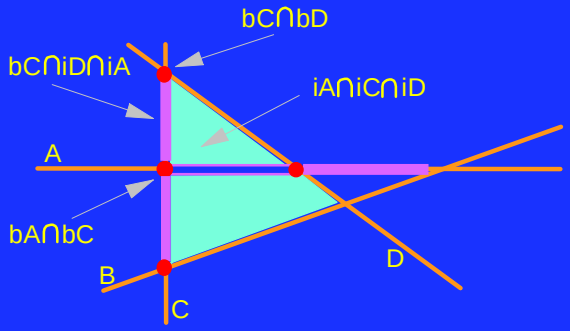


Simplices NOT in S may be omitted

Topological set operators

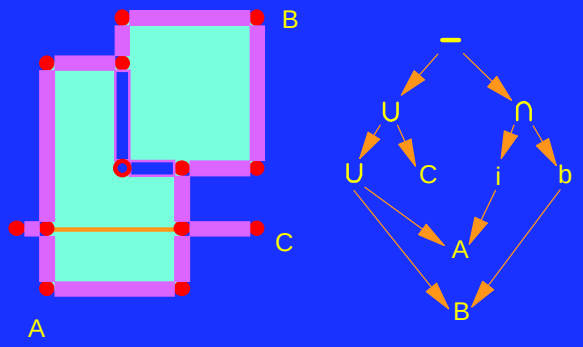


Any collection of simplices

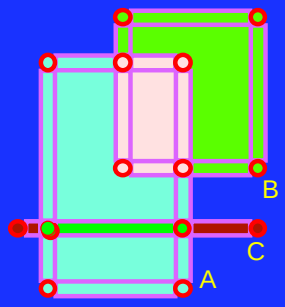


may be expressed using half-space combinations,
which should be computed from higher-level input

Hierarchical construction graph



Internal structure



A+B+C

"+" does union of sets but preserves internal structures

Here 5 regions:

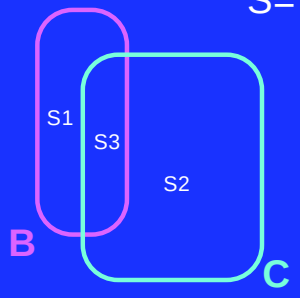
B - A - C
A ∩ B - C
A - B - C
A ∩ C - B
C - A - B

out of the 7 possible

Regions are broken into simplices

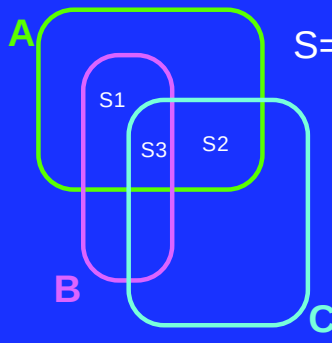
Boolean CNRG operators

S= (B+C)



+ yields 3 regions

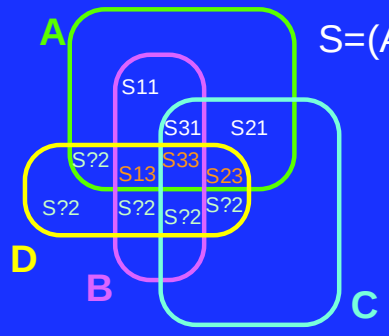
Boolean CNRG operators



S=(A&(B+C))

& trims down these 3 regions to the interior of A

Boolean CNRG operators



S=(A&(B+C))+D

S23

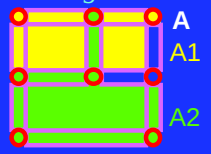
S?2

don't care for
D - (A&(B+C))

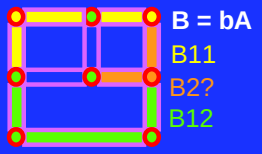
+ splits each one of the 3 regions: iD and cD
+ adds a 7th region: S?2 (the rest of D)

Topological CNRG operators

2 regions



A
A1
A2

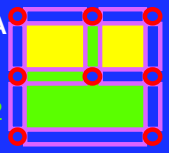


B = bA
B11
B2?
B12

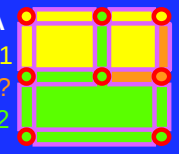
2 regions (trimmed)
+ new 3rd region

2 regions (trimmed)

I = iA
I1
I2



K = kA
K11
K2?
K12



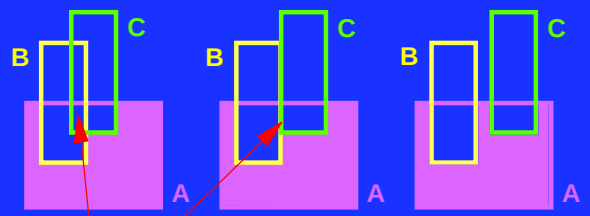
2 original regions
+ new 3rd region

Constructive Non-Regular Geometry

Operators on aggregates of disjoint regions:

- Aggregation — *combines disjoint objects*
- Simplification — *returns single region*
- Complement — *set complement*
- Interior — *restricts each region*
- Exterior — *interior of complement*
- Boundary — *restricted region + one*
- Closure — *old regions + one*
- Intersection — *pairwise combinations*
- Difference — *restricted regions*
- Union — *3 combinations*
- k-cells — *only cells of dimension k*

Features of CNRG objects



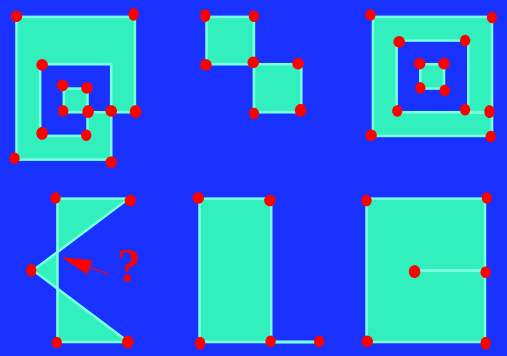
S = (A - B) - C

The feature of interest here is in cS: **S33 = A&B&C**
Extended signature semantics: treat - as a +

CNRG summary

- Compact specification
- Hierarchical (bottom-up) design
- Full topological coverage
- Preserves internal structure
- Signature identifies each region
- Regions have arbitrary topology
- Region existence and connectivity can only be assessed through evaluation

What is a valid polygon?

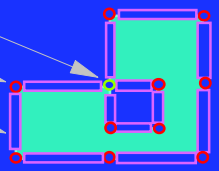


Wrong question! First, ask: "What is a polygon?"
Then select a representation scheme.

A definition

A polygon is a:

- connected
- bounded
- subset of the plane
- equal to the interior of its closure
- having as boundary a finite union of pairwise disjoint cells:
  - crossings
  - vertices
  - edges

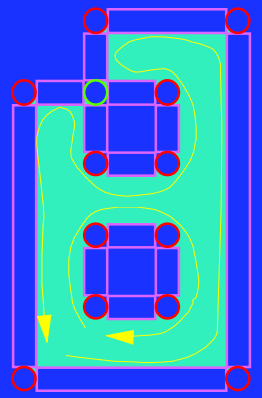


BRep of a polygon

Geometry defines:

- crossings
- vertices
- edges
- loops

crossing = non manifold
vertex = non smooth
edge = connected
orientation = first left
loop = max connected subsets of Bdry

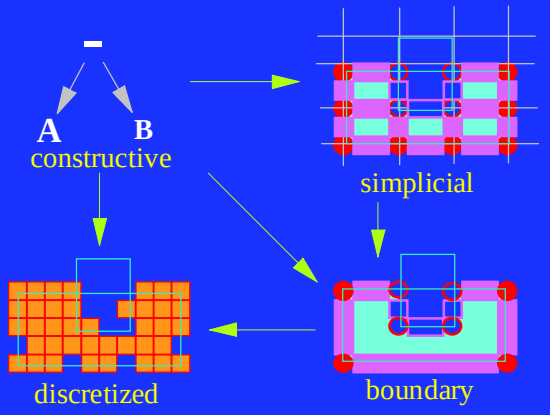


Possible representation of a polygon

- List of points (coordinates)
  - List of loops (external + holes)
    - Each loop = cyclic list of point uses

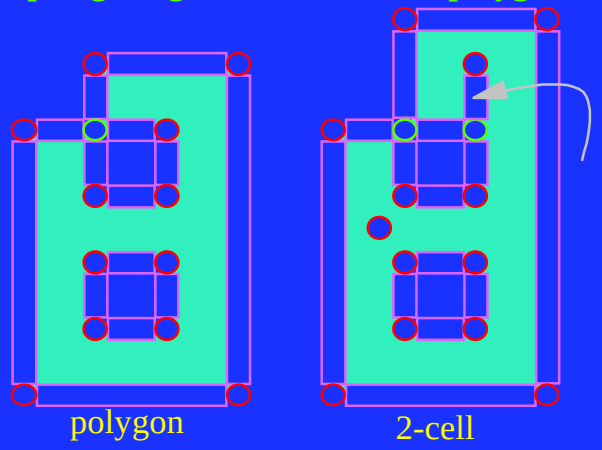
This representation is completely defined for a given geometry (provided that we choose an ordering scheme for points and uses).

Representation schemes



	Recipe	Voxels	Cells	BRep
Data size	Compact	Huge	Big	Medium
Speed	Slow	Very fast	Medium	Fast
Code	Elegant	Simple	Complex	Very hard
Design	Efficient	Tedious	No way!	Tricky

Topological generalization of a polygon



Generalized 2-cell

- connected
- bounded
- relatively open
- subset of manifold
- bounded by a finite union of pairwise disjoint 1-cells and 0-cells

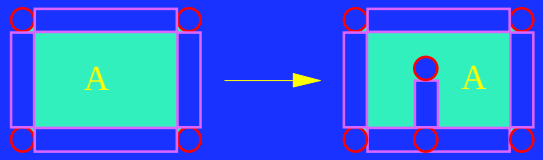
Representation:

- reference to supporting surface
- list of bounding vertices
- list of bounding edges (with nbhd)

Selective Geometric Complex

- Object = list of pairwise disjoint cells
  - Cell = connected, relatively open, bounded, subset of a manifold
  - Boundary of each cell = union of other cells in the object
- Selection of "active" cells
- Features = different selections

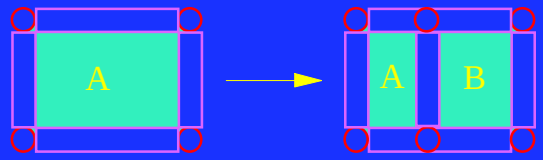
Internal structure in SGCs



Add the 2 vertices and the edge to the boundary of cell A

This removes them from cell A, but they may still be active in the object or in a feature

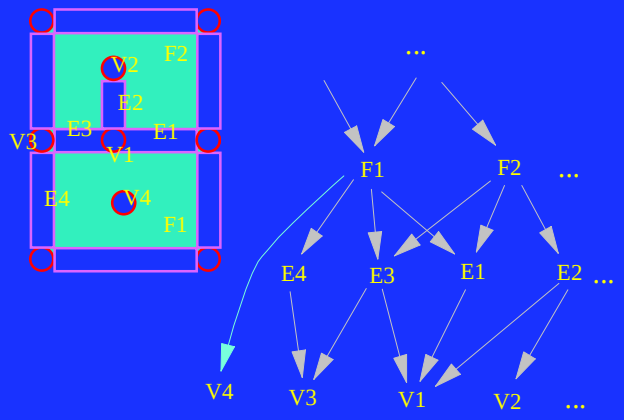
Splitting an SGC cell



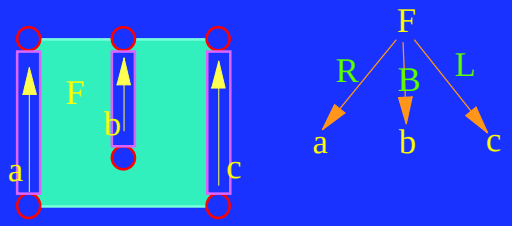
Adding the new edge splits the face

A new 2-cell (B) is created.
Both A and B may retain the attributes of A.

Representation of SGCs

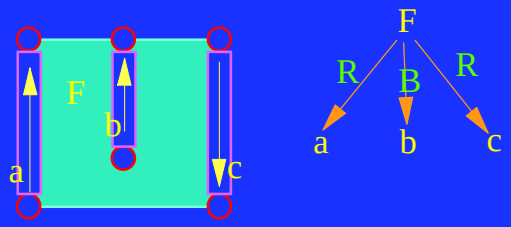


Orientation and neighborhood side



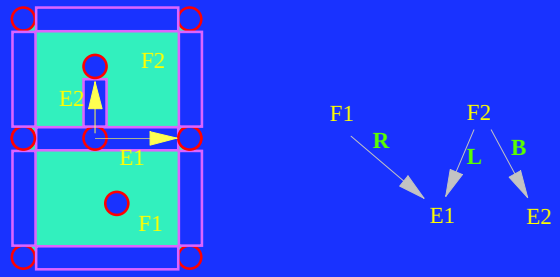
- F is on the right of a
- F is on both sides of b
- F is on the left of c

Orientation and neighborhood side

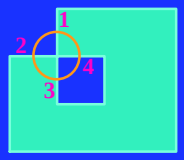


- F is on the right of a
- F is on both sides of b
- F is on the right of c

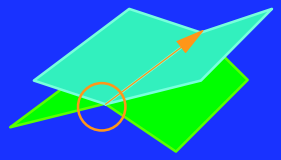
Neighborhoods in SGCs



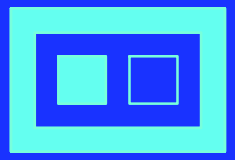
Ordering incidence relations



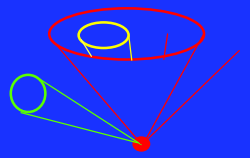
edges around vertices



faces/branches around edges



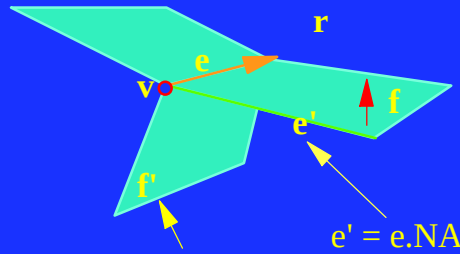
nesting loops and shells



vertex cones

NAILs: Order in SGCs

(Next cell Around cell In cell List)



e' = e.NAIL(v,f)

f' = f.NAIL(e,r)

Store NAIL table with each cell of SGC

Putting it all together

Users manipulate CNRG objects

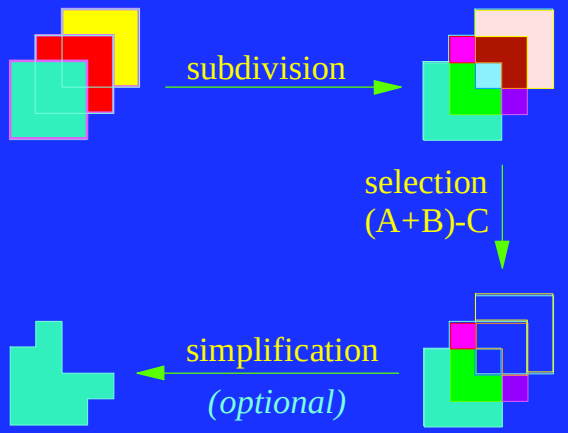
- primitives
- operators
- regions signatures
- features

An SGC model is derived from CNRG

SGC cells are (unions of) connected components of simplices induced by primitives' cells

CNRG regions and features are represented as lists of SGC cells

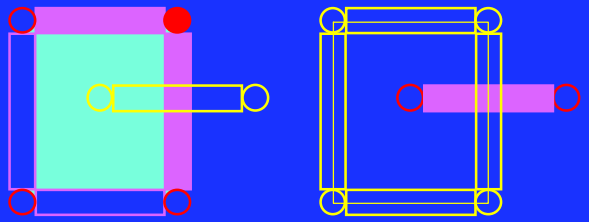
CNRG-to-SGC conversion



Subdivision

Insert lower-dimensional cells of other primitives into the boundary of each cell

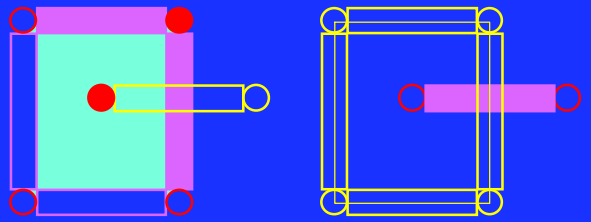
A B



Subdivision

Add vertices of A to boundaries of cells of B and vice versa.

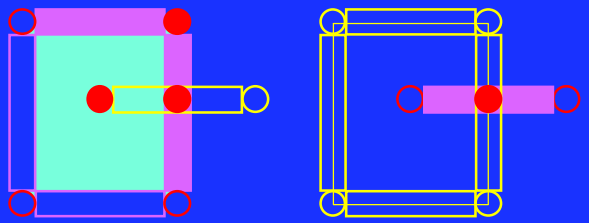
A B



Subdivision

Subdivide edges of both by inserting their pairwise intersections

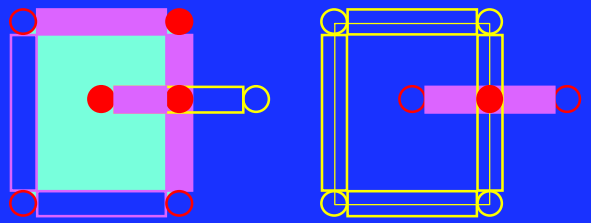
A B



Subdivision

Insert edges of A in the boundaries of the faces of B and vise versa

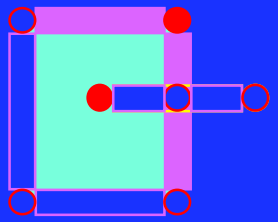
A\B B\A



Selection

Combines sells from both and select which cells should be ACTIVE

A-B

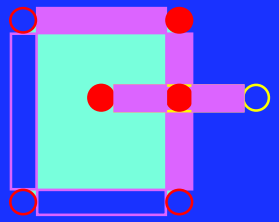


Could apply any filter using CNRG operators and features

A different selection

Combines sells from both and select which cells should be ACTIVE

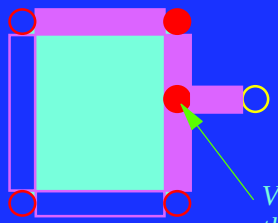
A+B



Simplification

Merge cells without changing the pointset nor the structure

s(A+B)



Vertex required to preserve the validity of SGC models

Properties of simplification

One pass algorithm:

For each cell by decreasing dimension

- Delete if non-active and not needed
- Absorb inside a single higher dim cell
- Join with other cells of same dim

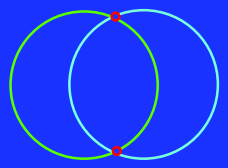
sA=ssA

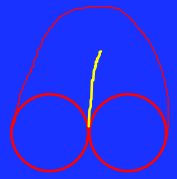
Produces a unique rep for a pointset

Preserves desired structure

Issues for curved geometries

- Cost and reliability of geometric intersections
- Geometric singularities (cusps)
- Identification of branches and components
- Multiple representations (trimmed patch)
- Computing order and inclusion





Software architecture issues

- Geometry independent API
- Interface between geometry and topology
  - Intersection returns a complex
  - Bidirectional links
- Robustness (floating point errors)
- Persistent references to user selected cells
- Merge objects from different modellers
- Capture and resolve constraints
- Editing the structure or the creation steps

CONCLUSIONS

- Design/edit in CNRG terms
- Interrogate / mark using features
- Algorithmic conversion to SGCs
- Efficient editing of feature selection
- No topological restrictions
- No geometric or dimension restrictions
- Independent of geometric reps