Use of Rational Numbers in the Design of Robust Geometric Primitives for Three-Dimensional Spatial Database Systems

Brian E. Weinrich & Markus Schneider*
University of Florida
Department of Computer & Information Science & Engineering
Gainesville, FL 32611, USA
{brianew, mschneid}@cise.ufl.edu

ABSTRACT

A necessary step in the implementation of three-dimensional spatial data types for spatial database systems and GIS is the development of robust geometric primitives. The authors have previously shown the need for 3D spatial data types and rigorously defined them. In this paper, we propose a set of 3D geometric primitives that can be used to implement them robustly. We provide for their robustness by specifying them using rational numbers. In the discretization of space, the developers of two-dimensional spatial data types have used simplicial complexes, realms or dual grids to produce robustness, but extending any of these to 3D is not adequate. Furthermore, rational number theory is sufficiently developed to apply to 3D implementation primitives. Efforts are lacking, however, in the field of spatial databases to show that spatial operations involving 3D spatial data types are closed under rational arithmetic. We therefore define four geometric primitives using rational numbers: *point, segment, facet* and *solid* which correspond to 0D, 1D, 2D and 3D spatial objects respectively. Also, we compare the rational specification of 3D primitives to the discretization methods used in 2D. Finally, we show that intersections involving these primitives have rational closure. We therefore conclude that use of rational numbers in the design of geometric primitives provides for a robust implementation of three-dimensional spatial data types.

Categories and Subject Descriptors: H.2.8 Database Applications — Spatial Databases and GIS.

General Terms: Design, Algorithms, Theory.

Keywords: geometric primitives, rational numbers, discrete model, 3D spatial data types, spatial database, GIS.

1. INTRODUCTION

The development of a three-dimensional spatial database system requires determining how to robustly represent three-dimensional spatial data types in computer memory. Spatial database systems have more demanding robustness requirements than standard programming languages can provide. Implementations of both the definitions of spatial data types and spatial operations must produce results that are much closer to exactness than is possible in floating or fixed point operations. They must especially provide for topological correctness. We will propose in this paper that implementers of 3D spatial database systems use arbitrary precision rational arithmetic rather than simplicial complexes, realms or dual grids that the developers of two-dimensional database systems have used.

Developing a robust implementation of 3D spatial data types initially requires two things. First of all, we must specify 3D geometric primitives through which we can represent the data types in computer storage. We will specify them using rational numbers so that we can use an arbitrary precision rational number system to perform high precision arithmetic in the spatial database system.

Secondly, for rational arithmetic to be feasible for 3D spatial database systems, spatial operations in 3D must have rational closure. Thus, we will show that the primitives have rational intersections. Since mathematicians long ago proved the closure of rational arithmetic, we will therefore show that a 3D spatial database system is closed for both arithmetic and spatial operations when using rational numbers.

This work is an important step in the implementation of 3D spatial data types. We proceed in the following way. Section 2 discusses related work. Section 3 presents the 3D geometric implementation primitives. Section 4 shows the rational closure of intersections of the 3D primitives. Section 5 draws conclusions and discusses future work.

2. RELATED WORK IN DESIGN, DISCRETIZATION AND RATIONAL NUMBERS

The work presented in this paper has as a basis work that has been done in three areas related to the robust implementation of 3D spatial database systems. These three areas are the design of abstract three-dimensional spatial data types, the discretization of spatial objects and the use of rational numbers. We now proceed one by one to discuss these related areas.

If we are to talk about implementation issues for abstract 3D spatial data types and about robustness in particular, their design must exist so that it is clear what we are implementing. It must also be clear that they are important. The authors' work in [9] provides a comprehensive and rigorous definition of 3D data types. In [9], we rigorously define the data types and specify geometric spatial operations. We also reveal a need for them. We intend this modeling

*This work was partially supported by the National Science Foundation under grant number NSF-CAREER-IIS-0347574.

GIS'05, November 4, 2005, Bremen, Germany.

of the 3D spatial data types as a first step in developing them. We give both unstructured and structured definitions of the data types which are *point3D*, *line3D*, *surface*, *relief* and *volume*. We refer the reader to [9] for the definitions of the abstract data types and other detailed information on our abstract design.

It is impossible to exactly represent the continuous boundaries or interiors of spatial objects in finite computer memory. Like the implementers of 2D spatial data types have done, we must therefore use simple objects called geometric primitives to form discretized approximations of 3D spatial objects. When we specify 3D geometric primitives, we must define operations including predicates which give relationships between the primitives. The work reported in [12] performs the important task of classifying 3D topological relationships between spatial objects using the 9-intersection model, but does not deal with primitives and does not provide precise mathematical formulas for operations between primitives.

To discretize spatial objects, we must also discretize space. The developers of 2D systems have used simplicial complexes or realms to discretize space and have suggested the use of dual grids. We therefore now review some literature on the use of simplicial complexes, realms or dual grids in 2D spatial database systems.

The discussion in [6] provides an excellent overview of simplicial complexes and realms in 2D spatial database systems. Simplicial complexes have shortcomings which [8] points out. Their independence from computer arithmetic makes it difficult to connect them to finite arithmetic and spatial databases. Also, the triangular networks formed by the 2-complexes contain more information than is needed for a spatial database system.

Thus, many current implementations of 2D spatial database systems rely on realms which were originally developed in [8]. A realm in 2D is a grid with points and line segments. The grid points and line segments are geometric primitives for 2D spatial data types. The work in [8] provides a thorough, robust design and implementation of realms for a 2D spatial database system. Although use of realms overcomes the problems of simplicial complexes, it has some problems of its own. Most of these problems relate to the redrawing process required when a spatial object is updated or inserted into a database. See [3] for a treatment of these issues. These problems are minor relative to the problems solved by realms, but they invite attempts at improvements.

Dual grids, as proposed in [3], rely on realms also. However, two grids are formed instead of only one, a grid for point objects and a grid for line segments. Dual grids even use rational numbers to obtain exact representations of the grid points. Use of dual grids has the potential to overcome many of the problems of realms.

Developers have successfully used realms and simplicial complexes to implement 2D spatial databases. Dual grids also show promise. It is not adequate, however, to extend these discretization methods to three dimensions. Realms have closure problems when extended to 3D. When using realms, the intersection of a line and plane must occur at a grid point of the realm. However, even the nearest realm grid point in 3D is probably not in the plane. With the rational implementation that we propose in this paper, we essentially have no grid beyond what the limits of finite computer memory impose on an arbitrary precision rational number system. We discuss more later on how the use of rational numbers in 3D provides a better discretization method and, thus, better robustness.

Next, we discuss what has been done to use rational numbers to produce robust results in spatial systems. To do so, we must answer two questions. First of all, what arbitrary precision rational number arithmetic packages have been developed and are they adequate? Also, what have spatial database researchers done to determine how rational software handles the intersection of 3D spatial objects?

In the past ten years researchers in computational geometry have done a lot of development work with rational software with a view towards applying the software to spatial objects. The evidence presented in [11] shows that research and development in computational geometry was headed in that direction ten years ago. Since then, developers have produced packages such as MAPC [7], PARI [1], apfloat [10] and rational arithmetic components within both LEDA and CGAL [2], to name a few. The cited papers report on the robustness of the rational arithmetic in these systems. Thus, plenty of adequate rational systems exist.

It is, of course, possible to perform robust arithmetic using rational numbers because they are both dense and closed under the basic arithmetic operations. What about spatial operations involving a line or plane in three dimensions? This is an important question because two of the 3D geometric primitives (which we present in Section 3) come from finite restrictions of lines and planes. The spatial database community has seen some work applying rational numbers to the representation of spatial objects. The work in [5, 4, 3] shows many are aware of the possibilities of rational representation of geometric primitives in both 2D and 3D. In order for rational specification of 3D geometrics primitives to produce a robust 3D spatial database system, however, the intersections of the primitives must be rational. We are not aware of any work within the spatial database community showing that the intersections of 3D geometric primitives are rational. Therefore, after defining the 3D primitives in the next section (Section 3), we will prove that they have rational intersections in Section 4.

3. GEOMETRIC PRIMITIVES FOR THREE DIMENSIONS

Geometric primitives are simple objects that are used to approximate and implement spatial objects in computer storage. The geometric primitives that we will specify for three-dimensional spatial databases systems are, intuitively, single points, line segments, plane segments and solids in 3D space. By plane segments, we mean a finite, contiguous part of a plane. Single points represent a *point3D* object and many small line segments approximate a curved *line3D* object. A collection of plane segments approximate a *surface*, *relief* or a boundary of a *volume*. One or more solids approximate a *volume*. So, we can specify all five abstract 3D spatial data types using the four 3D primitives that we are about to specify.

We name the 3D geometric primitives *point*, *segment*, *facet* and *solid*. Note that since these objects are primitives, they do not need to have holes or multi-component parts. It is the abstract spatial data types that must provide for holes and multi-component parts. Rather, we represent a hole in a *surface*, *relief* or *volume* using a primitive. Furthermore, we use a set of primitives to represent each component in an object that has multi-component parts.

In theory, we would like to specify the primitives using the abstract set of rational numbers $\mathbb{Q}$. In practice, this is impossible because of the finiteness of computer memory. So, instead, we establish a set Q^{repr} which contains as many rational numbers as are possible to represent in the memory available. Despite this finite representation, there are advantages to this approach over other implementations of spatial database systems including those employing realms and dual grids. First of all, the use of Q^{repr} takes us beyond the fixed lengths available in existing spatial database systems by using arbitrary precision rational number systems. The set Q^{repr} of representable rational numbers has much greater precision than types such as `float`, `double` and `longdouble`. Secondly, the set Q^{repr} contains many more numbers than there are along an axis in a realm. Thus, we can represent many more coordinates

than a realm can. In effect, we have a very fine grid here in which the number of grid points available to represent the primitives is $|Q^{repr}|^3$. This is also an improvement over dual grids because of their need, even in 2D, for two grids, a point grid and segment grid. Our approach requires only one grid. Also, we are not aware of anyone who has actually implemented dual grids, not even in 2D. Finally, because we rationally specify our 3D primitives, the realm based and geometric primitive based layers of [8] merge into a single layer here.

As we specify the primitives, we will, as might be expected, give definitions that indicate what they look like. Lack of space, however, prevents us from giving all the operations of the primitives. Thus, we have several priorities in selecting operations to specify. Most importantly, we define operations on primitives needed to define other primitives. Also, we concentrate on operations for the first three primitives, *point*, *segment* and *facet*. This is because the *solid* operations are not necessary for defining other primitives. Also, there are far too many operations involving *solid* to fit in this paper. We must concentrate on predicates as opposed to operations that yield another primitive because predicates are more important for defining other primitives. We do, however, define the important intersection operations that yield a primitive. We will give both signatures and semantics for all operations that we define. We will also provide figures for most of the operations involving *facet*s. Figures for operations involving two *segment*s, however, are already available in the literature such as [8].

3.1 The Primitives *point* and *segment*

First, we define *point* and *segment*. We define these two primitives together because operations between them are necessary to define operations involving *segment*s. The primitive *point* is a point in 3D space which has representable rational numbers as its elements. We define it as follows.

$$point = \{(x,y,z) \,|\, x,y,z \in Q^{repr}\}.$$

We need the following two predicates on *point*s to define *segment*. Suppose $p_1 = (x_1,y_1,z_1)$ and $p_2 = (x_2,y_2,z_2)$ are two *point*s. When the two *point*s are *equal*,

$$\begin{aligned} &=: \quad point \times point \rightarrow bool \\ &p_1 = p_2 \;\Leftrightarrow\; x_1 = x_2 \wedge y_1 = y_2 \wedge z_1 = z_2. \end{aligned}$$

When they are *not equal*,

$$\begin{aligned} &\neq: \quad point \times point \rightarrow bool \\ &p_1 \neq p_2 \;\Leftrightarrow\; x_1 \neq x_2 \vee y_1 \neq y_2 \vee z_1 \neq z_2. \end{aligned}$$

When $p_1 \neq p_2$, we also say that p_1 and p_2 are *disjoint*.

Now, we can define *segment*. It is an ordered pair of *point*s.

$$segment = \{(p_1,p_2) \mid p_1,p_2 \in point,\ p_1 < p_2\}.$$

Thus, p_1 and p_2 are the endpoints of a three-dimensional *segment*. Next, we define two predicates between *point* and *segment* that are important in defining operations between two *segment*s. Suppose p is a *point* and $s = (p_1,p_2)$ is a *segment* where $p_1 = (x_1,y_1,z_1)$ and $p_2 = (x_2,y_2,z_2)$. Then we say that p *on* s if p lies on the *segment* s. That is,

$$\begin{aligned} &on: \quad point \times segment \rightarrow bool \\ &p \;on\; s \;\Leftrightarrow\; p \in \{q = (x,y,z) \mid \\ &\qquad \text{(i)} \quad (x_1 \le x \le x_2) \vee (x_2 \le x \le x_1) \\ &\qquad \text{(ii)} \quad (y_1 \le y \le y_2) \vee (y_2 \le y \le y_1) \\ &\qquad \text{(iii)} \quad (z_1 \le z \le z_2) \vee (z_2 \le z \le z_1) \\ &\qquad \text{(iv)} \quad \frac{x-x_1}{x_2-x_1} = \frac{y-y_1}{y_2-y_1} = \frac{z-z_1}{z_2-z_1}\}. \end{aligned}$$

The *in* predicate is almost the same as the *on* predicate. The only difference is that the *point* cannot lie on the endpoints of the *segment*. Thus,

$$\begin{aligned} &in: \quad point \times segment \rightarrow bool \\ &p \;in\; s \;\Leftrightarrow\; p \in \{q = (x,y,z) \mid \\ &\qquad \text{(i)} \quad q \;on\; s \\ &\qquad \text{(ii)} \quad (x \neq x_1) \wedge (x \neq x_2) \\ &\qquad \text{(iii)} \quad (y \neq y_1) \wedge (y \neq y_2) \\ &\qquad \text{(iv)} \quad (z \neq z_1) \wedge (z \neq z_2)\}. \end{aligned}$$

When p *in* s, we can also say that s *overlaps* p or that s *intersects* p. Also, when $\neg(p \;on\; s)$, we say that p and s are *disjoint*.

Now, we define operations between two *segment*s. We refer the reader to [8] for a figure illustrating them. Suppose $s_1 = (p_1,p_2)$ and $s_2 = (q_1,q_2)$ are two *segment*s. Also, suppose $p_1 = (x_1,y_1,z_1)$, $p_2 = (x_2,y_2,z_2)$, $q_1 = (x_3,y_3,z_3)$ and $q_2 = (x_4,y_4,z_4)$. When s_1 and s_2 are *equal*,

$$\begin{aligned} &=: \quad segment \times segment \rightarrow bool \\ &s_1 = s_2 \;\Leftrightarrow\; p_1 = q_1 \wedge p_2 = q_2. \end{aligned}$$

When they are *not equal*,

$$\begin{aligned} &\neq: \quad segment \times segment \rightarrow bool \\ &s_1 \neq s_2 \;\Leftrightarrow\; p_1 \neq q_1 \vee p_2 \neq q_2. \end{aligned}$$

The two *segment*s s_1 and s_2 *overlap* if they share a common *segment*. This means that the *segment*s must lie in the same line. Thus,

$$\begin{aligned} &overlaps: \quad segment \times segment \rightarrow bool \\ &s_1 \;overlaps\; s_2 \Leftrightarrow \\ &\qquad \text{(i)} \quad (p_1 \;in\; s_2) \vee (p_2 \;in\; s_2) \\ &\qquad\qquad \vee (q_1 \;in\; s_1) \vee (q_2 \;in\; s_1) \;\wedge \\ &\qquad \text{(ii)} \quad \frac{x_3-x_1}{x_2-x_1} = \frac{y_3-y_1}{y_2-y_1} = \frac{z_3-z_1}{z_2-z_1} \;\wedge \\ &\qquad \text{(iii)} \quad \frac{x_4-x_1}{x_2-x_1} = \frac{y_4-y_1}{y_2-y_1} = \frac{z_4-z_1}{z_2-z_1}. \end{aligned}$$

They *meet* if they have exactly one common endpoint and no other common *point*s. Therefore, if the two *segment*s *meet*, it is not possible for them to intersect at any other place besides their common endpoint. That is, if $\oplus$ is the XOR operator,

$$\begin{aligned} &meets: \quad segment \times segment \rightarrow bool \\ &s_1 \;meets\; s_2 \;\Leftrightarrow \text{(i)}\; \neg\,(s_1 \;overlaps\; s_2) \wedge \\ &\qquad\qquad \text{(ii)}\; (p_1 = q_1) \oplus (p_1 = q_2) \\ &\qquad\qquad\quad \oplus (p_2 = q_1) \oplus (p_2 = q_2). \end{aligned}$$

They *intersect* if they have one common point other than their endpoints. In the next section (Section 4) we show that this common point is rational. More precisely,

$$\begin{aligned} &intersects: \quad segment \times \; segment \rightarrow bool \\ &s_1 \;intersects\; s_2 \;\Leftrightarrow\; \exists p \in point:\; (p \;in\; s_1) \\ &\qquad\qquad\qquad\qquad\qquad\qquad \wedge\, (p \;in\; s_2). \end{aligned}$$

One *segment touches* the other *segment* if they have one common *point* which is the endpoint of exactly one of the *segment*s.

$$\begin{aligned} &touches: \quad segment \times segment \rightarrow bool \\ &s_1 \;touches\; s_2 \;\Leftrightarrow \text{(i)}\; \neg\,(s_1 \;overlaps\; s_2) \wedge \\ &\qquad\qquad\quad \text{(ii)}\; (p_1 \;in\; s_2) \vee (p_2 \;in\; s_2) \\ &\qquad\qquad\qquad \vee (q_1 \;in\; s_1) \vee (q_2 \;in\; s_1). \end{aligned}$$

They are *collinear* if they are in the same line, but have no common *point*s.

$$collinear: \quad segment \times segment \rightarrow bool$$

$$\begin{aligned} s_1 \ collinear \ s_2 \Leftrightarrow \\ &\text{(i)} \quad s_1 \neq s_2 \wedge \neg (s_1 \ meets \ s_2) \\ &\qquad \wedge \neg (s_1 \ overlaps \ s_2) \ \wedge \\ &\text{(ii)} \quad \frac{x_3-x_1}{x_2-x_1} = \frac{y_3-y_1}{y_2-y_1} = \frac{z_3-z_1}{z_2-z_1} \wedge \\ &\text{(iii)} \quad \frac{x_4-x_1}{x_2-x_1} = \frac{y_4-y_1}{y_2-y_1} = \frac{z_4-z_1}{z_2-z_1}. \end{aligned}$$

The two *segment*s are *coplanar* if they lie in the same plane. Note that if two *segment*s are not *coplanar*, then it is impossible for them to *meet*, *intersect*, *overlap* or *touch*. Neither can they be *collinear*.

$$coplanar: \quad segment \times segment \rightarrow bool$$

$$s_1 \ coplanar \ s_2 \ \Leftrightarrow \begin{vmatrix} x_2-x_1 & y_2-y_1 & z_2-z_1 \\ x_3-x_1 & y_3-y_1 & z_3-z_1 \\ x_4-x_1 & y_4-y_1 & z_4-z_1 \end{vmatrix} = 0.$$

Two *segment*s are *disjoint* if they have no common *point*s. Stated in terms of the previous definitions, they are *disjoint* if they are not *equal* and they do not *meet*, *intersect*, *overlap* or *touch*.

There is also an operation involving *segment*s that yields another primitive as its result rather than a boolean value. It is *intersection*. When two *segment*s intersect, the *intersection* operation yields the *point* at which they intersect. An *intersection* may occur whenever the *segment*s *meet*, *intersect* or *touch*. If the *segment*s s_1 and s_2 *meet*, *intersect* or *touch*, then

$$\begin{aligned} &intersection: \quad segment \times segment \rightarrow point \\ &intersection(s_1, s_2) = (x_1 + l_1 t, \ y_1 + m_1 t, \ z_1 + n_1 t) \end{aligned}$$

where

$$t = \frac{l_3 - m_3}{-l_1 m_3 + l_3 m_1}, \qquad l_1 = \frac{x_2 - x_1}{d_1},$$

$$m_1 = \frac{y_2 - y_1}{d_1}, \qquad n_1 = \frac{z_2 - z_1}{d_1},$$

$$d_1 = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2 + (z_2-z_1)^2},$$

$$l_3 = \frac{x_4 - x_3}{d_3}, \qquad m_3 = \frac{y_4 - y_3}{d_3}, \quad \text{and}$$

$$d_3 = \sqrt{(x_4-x_3)^2 + (y_4-y_3)^2 + (z_4-z_3)^2}.$$

We will show that this intersection is rational in the next section (Section 4) despite the presence of square roots.

3.2 The Primitive *facet*

Now we can turn our attention to defining *facet*. Intuitively, a *facet* is a finite, contiguous part of a plane in three-dimensional space. We use *facet*s as the discrete building blocks for *surface*s, *relief*s and boundaries of *volume*s. To define *facet* we use *segment*s just like we used *point*s to define *segment*. That is, *segment*s give the boundary of a *facet* like *point*s give the boundary or endpoints of a *segment*. We will refer to the *segment*s forming the boundary of a *facet* as boundary *segment*s. Rigorously,

$$\begin{aligned} facet = \{\{s_0, s_1, ..., s_{m-1}\} \mid \\ &\text{(i)} \quad \forall i \in \{0, ..., m-1\}, \ s_i \in segment \\ &\text{(ii)} \quad \forall i \in \{0, ..., m-1\}, \\ &\qquad s_i \ meets \ s_{(i+1) \bmod m} \\ &\text{(iii)} \quad \text{No more than two of } s_0, s_1, ..., \text{ and} \\ &\qquad s_{m-1} \ meet \text{ at any one } point \\ &\text{(iv)} \quad \forall i, j \in \{0, ..., m-1\}, \\ &\qquad s_i \text{ and } s_j \text{ is } coplanar\}. \end{aligned}$$

Thus, the boundary *segment*s of a *facet* are $s_0, s_1, ..., s_{m-1}$.

To define some of the operations involving *facet*s, we need to know whether two objects are in the same plane. So, the first operation we define for *facet*s is the *coplanar* predicate. We will use the type variable γ ranging over $\{point, segment, facet\}$. Let $o \in \gamma$ and let f be a *facet*. Furthermore, let $p = (x,y,z)$ be a *point* of o and let $p_3 = (x_3, y_3, z_3)$, $p_4 = (x_4, y_4, z_4)$ and $p_5 = (x_5, y_5, z_5)$ be *disjoint point*s (i.e., endpoints) in the boundary *segment*s of f. Then,

$$coplanar: \quad \gamma \times facet \rightarrow bool$$

$$o \ coplanar \ f \ \Leftrightarrow \begin{vmatrix} x-x_3 & y-y_3 & z-z_3 \\ x_4-x_3 & y_4-y_3 & z_4-z_3 \\ x_5-x_3 & y_5-y_3 & z_5-z_3 \end{vmatrix} = 0.$$

More generally, to define operations involving *facet*s, we must consider operations that involve a *point*, a *segment* and another *facet*. We first consider predicates involving a *point* and a *facet*. These are the predicates *on*, *in* and *out*. They test whether a *point* is on one of the boundary *segment*s of a *facet*, or is inside or outside of the boundary formed by the boundary *segment*s. The *on* and *in* operations, in particular, are superficially analogous to the *on* and *in* operations for *point*s and *segment*s. If f is a *facet*, as before, and $p = (x,y,z)$ is a *point*,

$$\begin{aligned} &on: \quad point \times facet \rightarrow bool \\ &p \ on \ f \ \Leftrightarrow \ \exists s \in f : p \ on \ s. \end{aligned}$$

Suppose x_{max}, y_{max} and z_{max} are the maximum coordinate values in the boundary *segment*s defining the *facet* f. Furthermore, let $p_{big} = (x_{big}, y_{big}, z_{big})$ be a *point* that is *coplanar* with the *facet* f such that $x_{big} > x_{max}$, $y_{big} > y_{max}$ and $z_{big} > z_{max}$. Thus, p_{big} must be outside the *facet* f. Next, let s_p be the *segment* (p, p_{big}). Intuitively, if p is inside the boundary formed by the boundary *segment*s of f, then s_p must *intersect*, *touch* or *meet* an odd number of the boundary *segment*s. Formally, let $S_{on}(f)$ be the set of *segment*s in f whose right (but not left) endpoint is *on* s_p. Also, let $S_i(f)$ be the set of *segment*s in f that *intersect* s_p. Then,

$$\begin{aligned} &in: \quad point \times facet \rightarrow bool \\ &p \ in \ f \ \Leftrightarrow \ \neg(p \ on \ f) \ \wedge \ |S_{on}(f)| + |S_i(f)| \text{ is odd}, \end{aligned}$$

$$\begin{aligned} &out: \quad point \times facet \rightarrow bool \\ &p \ out \ f \ \Leftrightarrow \ \neg(p \ on \ f \ \vee \ p \ in \ f). \end{aligned}$$

When $p \ out \ f$, we also call p and f *disjoint*.

There are many operations involving a *segment* and a *facet*. Figure 1 illustrates them. Let us again suppose that $s = (p_1, p_2)$ is a *segment* and that $f = \{ s_0, s_1, ..., s_{m-1} \}$ is a *facet*. The *segment* *overlaps* the *facet* when part of it lies inside the *facet*. That is,

$$\begin{aligned} &overlaps: \quad segment \times facet \rightarrow bool \\ &s \ overlaps \ f \ \Leftrightarrow \text{(i)} \ s \ coplanar \ f \\ &\qquad \text{(ii)} \ ((p_1 \ in \ f \ \wedge \ p_2 \ out \ f) \\ &\qquad\qquad \vee \ (p_2 \ in \ f \ \wedge \ p_1 \ out \ f)) \\ &\qquad\qquad \vee \ (p_1 \ out \ f \ \wedge \ p_2 \ out \ f \\ &\qquad\qquad\quad \wedge \ (\exists s_i \in f : s_i \ intersects \ s \\ &\qquad\qquad\qquad \vee \ s_i \ touches \ s)). \end{aligned}$$

It *meets* the *facet* when it *touches* or *meets* a boundary *segment* of the *facet*. If the *segment touches* the *facet*, however, then its endpoint must lie on the boundary *segment* of the *facet* rather than the other way around. (This distinguishes *meets* from a predicate we will cover later, *tangent*.)

$$
\begin{aligned}
&\mathit{meets}: \quad \mathit{segment} \times \mathit{facet} \rightarrow \mathit{bool}\\
&s \ \mathit{meets}\ f \Leftrightarrow\\
&\qquad \text{(i)}\ \neg\,(s\ \mathit{overlaps}\ f) \wedge\\
&\qquad \text{(ii)}\ \exists i \in \{0,\ldots,m-1\}, \exists s_i \in f:\\
&\qquad\qquad s_i\ \mathit{meets}\ s\\
&\qquad\qquad \vee\,(s_i\ \mathit{touches}\ s\\
&\qquad\qquad\quad \wedge\,(p_1\ \mathit{on}\ s_i\ \vee\ p_2\ \mathit{on}\ s_i)).
\end{aligned}
$$

They *intersect* if the interior of the *segment* goes through one *point* in the interior of the *facet*. In the next section (Section 4), we show that the point where they *intersect* is rational and, thus, is a *point*.

$$
\begin{aligned}
&\mathit{intersects}: \quad \mathit{segment} \times \mathit{facet} \rightarrow \mathit{bool}\\
&s\ \mathit{intersects}\ f \ \Leftrightarrow\ \exists p\ \mathit{in}\ s : p\ \mathit{in}\ f\\
&\qquad\qquad\qquad \wedge \neg\,(s\ \mathit{coplanar}\ f).
\end{aligned}
$$

The *segment touches* the *facet* when an endpoint of the *segment* is *in* the interior of the *facet*.

$$
\begin{aligned}
&\mathit{touches}: \quad \mathit{segment} \times \mathit{facet} \rightarrow \mathit{bool}\\
&s\ \mathit{touches}\ f \ \Leftrightarrow\ (p_1\ \mathit{in}\ f\ \oplus\ p_2\ \mathit{in}\ f)\\
&\qquad\qquad\qquad \wedge \neg\,(s\ \mathit{overlaps}\ f).
\end{aligned}
$$

When a *segment overlaps* a boundary *segment* of a *facet*, the *segment* and *facet* are *incident*.

$$
\begin{aligned}
&\mathit{incident}: \quad \mathit{segment} \times \mathit{facet} \rightarrow \mathit{bool}\\
&s\ \mathit{incident}\ f \ \Leftrightarrow\ \exists i \in \{0,\ldots,m-1\}, \exists s_i \in f:\\
&\qquad\qquad\qquad s_i\ \mathit{overlaps}\ s.
\end{aligned}
$$

They are *tangent* when the *segment intersects* one of the boundary *segment*s of the *facet*. It can also *touch* the boundary *segment*, but only if the endpoint of the boundary *segment* is *on* the *segment*.

$$
\begin{aligned}
&\mathit{tangent}: \quad \mathit{segment} \times \mathit{facet} \rightarrow \mathit{bool}\\
&s\ \mathit{tangent}\ f \ \Leftrightarrow \text{(i)}\ \neg\,(s\ \mathit{overlaps}\ f) \wedge\\
&\qquad\qquad \text{(ii)}\ \exists i \in \{0,\ldots,m-1\}, \exists s_i \in f:\\
&\qquad\qquad\qquad (s_i\ \mathit{intersects}\ s)\\
&\qquad\qquad\qquad \vee (s_i\ \mathit{touches}\ s \wedge \neg\ p_1\ \mathit{on}\ s_i\\
&\qquad\qquad\qquad\quad \wedge \neg\ p_2\ \mathit{on}\ s_i).
\end{aligned}
$$

Note that a *segment* can be *tangent* with a *facet* at more than one *point*. The *segment* lies *inside* a *facet* when it is completely within the *facet*.

$$
\begin{aligned}
&\mathit{inside}: \quad \mathit{segment} \times \mathit{facet} \rightarrow \mathit{bool}\\
&s\ \mathit{inside}\ f \ \Leftrightarrow \text{(i)}\ (p_1\ \mathit{in}\ f\ \vee\ p_1\ \mathit{on}\ f)\\
&\qquad\qquad\qquad \wedge (p_2\ \mathit{in}\ f\ \vee\ p_2\ \mathit{on}\ f) \wedge\\
&\qquad\qquad\quad \text{(ii)}\ \neg\,(\exists s_i \in f : s_i\ \mathit{intersects}\ s\\
&\qquad\qquad\qquad\qquad \vee\ s_i\ \mathit{touches}\ s).
\end{aligned}
$$

It lies *outside* the *facet* when none of the *segment* is in the *facet*.

$$
\begin{aligned}
&\mathit{outside}: \quad \mathit{segment} \times \mathit{facet} \rightarrow \mathit{bool}\\
&s\ \mathit{outside}\ f \ \Leftrightarrow \text{(i)}\ p_1\ \mathit{out}\ f\ \wedge\ p_2\ \mathit{out}\ f \wedge\\
&\qquad\qquad\quad \text{(ii)}\ \neg\,(s\ \mathit{intersects}\ f)\\
&\qquad\qquad\qquad \wedge \neg\,(s\ \mathit{overlaps}\ f)\\
&\qquad\qquad\qquad \wedge \neg\,(s\ \mathit{tangent}\ f).
\end{aligned}
$$

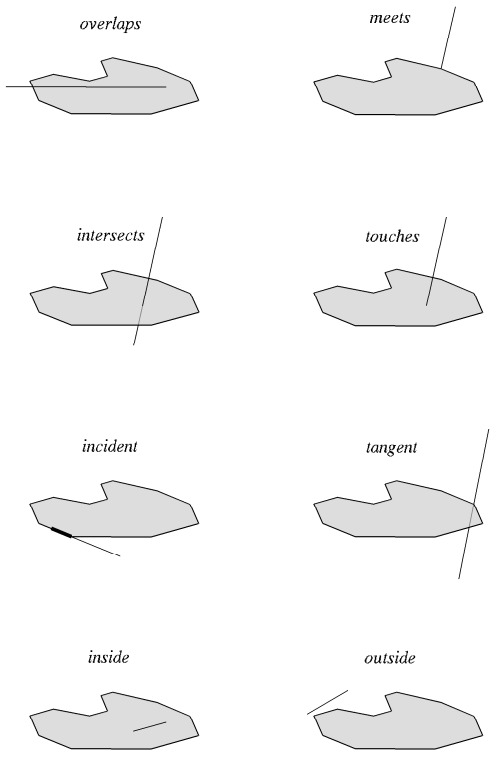

Figure 1: Predicates between a *segment* and a *facet*

Equivalently, we say that s and f are *disjoint* when s *outside* f.

Next, we define an operation involving a *segment* and a *facet* that yields another primitive as its result. It is the *intersection* operation for a *segment* and a *facet*. If the *segment* and *facet* are not *coplanar* and *meet*, *intersect*, *touch* or are *tangent*,

$$
\begin{aligned}
&\mathit{intersection}: \quad \mathit{segment} \times \mathit{facet} \rightarrow \mathit{point}\\
&\mathit{intersection}(s,f) = (x_1 + lt,\ y_1 + mt,\ z_1 + nt)
\end{aligned}
$$

where

$$
t = \frac{\Gamma(x_3 - x_1) + \Delta(y_3 - y_1) + \Phi(z_3 - z_1)}{\Gamma l + \Delta m + \Phi n},
$$

$$
l = \frac{x_2 - x_1}{d}, \qquad m = \frac{y_2 - y_1}{d}, \qquad n = \frac{z_2 - z_1}{d},
$$

$$
d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2},
$$

$$
\Gamma = \begin{vmatrix} y_4 - y_3 & z_4 - z_3 \\ y_5 - y_3 & z_5 - z_3 \end{vmatrix},
$$

$$
\Delta = \begin{vmatrix} x_4 - x_3 & z_4 - z_3 \\ x_5 - x_3 & z_5 - z_3 \end{vmatrix},
$$

$$
\text{and} \qquad \Phi = \begin{vmatrix} x_4 - x_3 & y_4 - y_3 \\ x_5 - x_3 & y_5 - y_3 \end{vmatrix}.
$$

Γ, Δ and Φ are the coefficients in the general equation of the plane in which the *facet* is located. We will show more about how a solution such as this is obtained and that it is rational in Section 4.

There are also many operations involving two *facet*s. Figure 2 illustrates them. The first four predicates we define are crucial to defining the 3D geometric primitive, *solid*. They are *overlaps*, *meets*, *intersects* and *touches*. Let $f_1 = \{s_{10}, s_{11}, ..., s_{1,m-1}\}$ and $f_2 = \{s_{20}, s_{21}, ..., s_{2,l-1}\}$ be two *facet*s where $s_{1j} = (p_{1j1}, p_{1j2})\ \forall j \in \{0, ..., m-1\}$ and $s_{2j} = (p_{2j1}, p_{2j2})\ \forall j \in \{0, ..., l-1\}$. The *overlaps* operation between two *facet*s indicates whether part of their interiors coincide. Thus, two *facet*s *overlap* when they are *coplanar* and a *point* (i.e., an endpoint) of at least one boundary *segment* of one of the *facet*s is *in* the other *facet*. Thus,

$$
\begin{aligned}
&\mathit{overlaps}: \quad \mathit{facet} \times \mathit{facet} \rightarrow \mathit{bool}\\
&f_1\ \mathit{overlaps}\ f_2 \Leftrightarrow \text{(i)}\ \ f_1\ \mathit{coplanar}\ f_2 \wedge\\
&\qquad\qquad \text{(ii)}\ \exists s_{1j} \in f_1 : (p_{1j1}\ \mathit{in}\ f_2)\\
&\qquad\qquad\qquad \vee\ (p_{1j2}\ \mathit{in}\ f_2)\\
&\qquad\qquad\quad \vee\ \exists s_{2j} \in f_2 : (p_{2j1}\ \mathit{in}\ f_1)\\
&\qquad\qquad\qquad \vee\ (p_{2j2}\ \mathit{in}\ f_1).
\end{aligned}
$$

The *meets* operation for *facet*s will be very important in defining *solid*. Two *facet*s *meet* if they have boundary *segment*s which are *equal*. Equivalently, we could say that they have at least one common *segment*. When two *facet*s *meet*, the two boundary *segment*s must be *equal* and not merely *overlap*. Furthermore, they have nothing else in common besides *segment*s that are *equal*. We will handle the case of when the two boundary *segment*s *overlap* later in a different predicate called *incident*. This makes our definition of *meets* more restricted than the predicate usually called meets in the 9-intersection model. We find this restriction necessary in order to define *solid*.

$$
\begin{aligned}
&\mathit{meets}: \quad \mathit{facet} \times \mathit{facet} \rightarrow \mathit{bool}\\
&f_1\ \mathit{meets}\ f_2 \Leftrightarrow \text{(i)}\ \ \neg\,(f_1\ \mathit{overlaps}\ f_2)\\
&\qquad\qquad\qquad\qquad \wedge \neg\,(f_2\ \mathit{overlaps}\ f_1) \wedge\\
&\qquad\qquad\qquad \text{(ii)}\ \ \exists s_{1i} \in f_1, \exists s_{2j} \in f_2 :\\
&\qquad\qquad\qquad\qquad s_{1i} = s_{2j} \wedge\\
&\qquad\qquad\qquad \text{(iii)} \neg\,(\exists s_{1i} \in f_1, \exists s_{2j} \in f_2 :\\
&\qquad\qquad\qquad\qquad s_{1i}\ \mathit{overlaps}\ s_{2j}\\
&\qquad\qquad\qquad\qquad \wedge\ s_{1i}\ \mathit{intersects}\ s_{2j}\\
&\qquad\qquad\qquad\qquad \wedge\ s_{1i}\ \mathit{touches}\ s_{2j}).
\end{aligned}
$$

The *facet*s *intersect* if they have at least one common *segment* in their interiors. Therefore, *facet*s that *intersect* cannot be *coplanar*. It is correct to refer to their intersection as containing a *segment* because *segment* is rationally defined as we show in Section 4.

$$
\begin{aligned}
&\mathit{intersects}: \quad \mathit{facet} \times \mathit{facet} \rightarrow \mathit{bool}\\
&f_1\ \mathit{intersects}\ f_2 \Leftrightarrow\ (\exists s_{1j} \in f_1 : s_{1j}\ \mathit{intersects}\ f_2)\\
&\qquad\qquad\qquad\qquad \vee\ (\exists s_{2j} \in f_2 : s_{2j}\ \mathit{intersects}\ f_1)\\
&\qquad\qquad\qquad\qquad \vee\ ((\exists s_{1j} \in f_1 : s_{1j}\ \mathit{tangent}\ f_2)\\
&\qquad\qquad\qquad\qquad\quad \wedge\ (\exists s_{2j} \in f_2 : s_{2j}\ \mathit{tangent}\ f_1)).
\end{aligned}
$$

One *facet* *touches* another when a single *point* of one of the *facet*s is *in* the interior of the other *facet*. This can only happen when a *point* (i.e., an endpoint) of a boundary *segment* of one of the *facet*s is *in* the other *facet*. The *meets* predicate does not allow an entire boundary *segment* to be *inside* the other *facet*. The *brushes* predicate, which we will define later, allows a boundary *segment* of one *facet* to be *inside* the other *facet*.

$$
\begin{aligned}
&\mathit{touches}: \quad \mathit{facet} \times \mathit{facet} \rightarrow \mathit{bool}\\
&f_1\ \mathit{touches}\ f_2 \Leftrightarrow\\
&\qquad\qquad \text{(i)}\ \ \neg\,(f_1\ \mathit{overlaps}\ f_2)\\
&\qquad\qquad\qquad \wedge \neg\,(f_1\ \mathit{intersects}\ f_2) \wedge\\
&\qquad\qquad \text{(ii)}\ (\exists s_{1i} \in f_1 : s_{1i}\ \mathit{touches}\ f_2)\\
&\qquad\qquad\qquad \vee\ (\exists s_{2i} \in f_2 : s_{2i}\ \mathit{touches}\ f_1).
\end{aligned}
$$

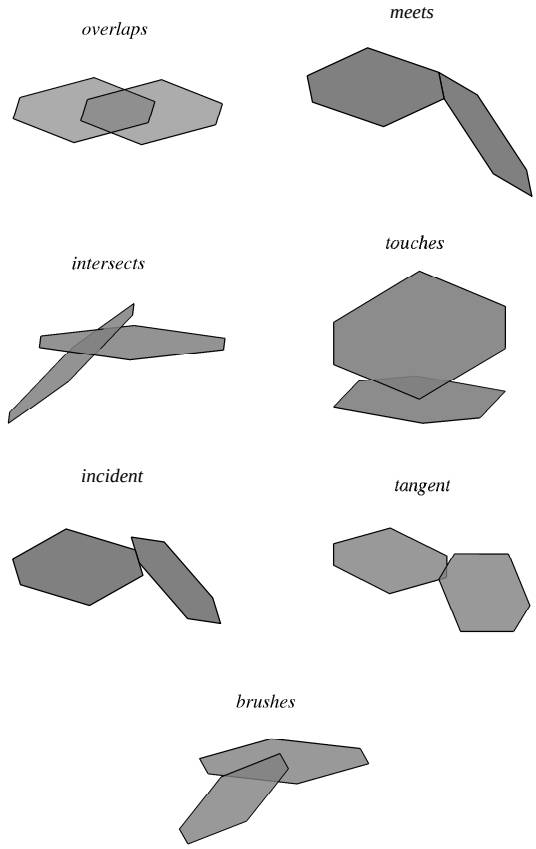

Figure 2: Predicates between two *facet*s

They are *incident* when a boundary *segment* from both *overlap*.

$$
\begin{aligned}
&\mathit{incident}: \quad \mathit{facet} \times \mathit{facet} \rightarrow \mathit{bool}\\
&f_1\ \mathit{incident}\ f_2 \Leftrightarrow\\
&\qquad\qquad \text{(i)}\ \ \neg\,(f_1\ \mathit{overlaps}\ f_2) \wedge\\
&\qquad\qquad \text{(ii)}\ (\exists s_{1i} \in f_1 : s_{1i}\ \mathit{incident}\ f_2)\\
&\qquad\qquad\qquad \vee\ (\exists s_{2i} \in f_2 : s_{2i}\ \mathit{incident}\ f_1).
\end{aligned}
$$

They are *tangent* if their boundaries have a common point.

$$
\begin{aligned}
&\mathit{tangent}: \quad \mathit{facet} \times \mathit{facet} \rightarrow \mathit{bool}\\
&f_1\ \mathit{tangent}\ f_2 \Leftrightarrow\\
&\qquad\qquad \text{(i)}\ \ \neg\,(f_1\ \mathit{overlaps}\ f_2)\\
&\qquad\qquad\qquad \wedge \neg\,(f_1\ \mathit{intersects}\ f_2) \wedge\\
&\qquad\qquad \text{(ii)}\ (\exists s_{1i} \in f_1 : s_{1i}\ \mathit{tangent}\ f_2)\\
&\qquad\qquad\qquad \vee\ (\exists s_{2i} \in f_2 : s_{2i}\ \mathit{tangent}\ f_1).
\end{aligned}
$$

The two *facet*s *brush* if part of one or more *segment*s of one *facet* lie in the interior of the other *facet* when the *facet*s are not *coplanar*.

$$
\begin{aligned}
&\mathit{brushes}: \quad \mathit{facet} \times \mathit{facet} \rightarrow \mathit{bool}\\
&f_1\ \mathit{brushes}\ f_2 \Leftrightarrow \text{(i)}\ \ \neg\,(f_1\ \mathit{coplanar}\ f_2)\\
&\qquad\qquad\qquad\qquad \wedge \neg\,(f_1\ \mathit{intersects}\ f_2) \wedge\\
&\qquad\qquad\qquad\quad \text{(ii)}\ (\exists s_{1i} \in f_1 : s_{1i}\ \mathit{inside}\ f_2\\
&\qquad\qquad\qquad\qquad\quad \vee\ s_{1i}\ \mathit{overlaps}\ f_2)\\
&\qquad\qquad\qquad\qquad \vee\ (\exists s_{2i} \in f_2 : s_{2i}\ \mathit{inside}\ f_1\\
&\qquad\qquad\qquad\qquad\qquad \vee\ s_{2i}\ \mathit{overlaps}\ f_1).
\end{aligned}
$$

Moreover, two *facet*s are *disjoint* if they have no points in common. That is, when they are *disjoint*, they do not *overlap*, *meet*, *intersect*, *touch* or *brush* or are not *incident* or *tangent*.

Finally, we define the *intersection* operation for two *facet*s. It yields *point*s and *segment*s as its result. Its signature indicates this:

$$\begin{aligned} intersection: \quad & facet \times facet \\ & \rightarrow \{s \mid s \in point \cup segment\}. \end{aligned}$$

The *intersection* operation is critical for the definition of *solid*. To define the *intersection* of two *facet*s we determine the *point* set of the line formed by the intersection of the supporting planes of the two *facet*s and confine the point set to the *facet*s themselves. In doing so, we use the *intersection* operation for a *segment* and a *facet*. The result is a set of *point*s and *segment*s that are all collinear. Suppose $s_{1i} = (p_{1i1}, p_{1i2})$ and $s_{1j} = (p_{1j1}, p_{1j2})$ are boundary *segment*s of *facet* f_1 such that $p_{1i1} = (x_3, y_3, z_3)$, $p_{1i2} = (x_4, y_4, z_4)$ and $p_{1j1} = (x_5, y_5, z_5)$. Suppose further that $s_{2i} = (p_{2i1}, p_{2i2})$ and $s_{2j} = (p_{2j1}, p_{2j2})$ are boundary *segment*s of *facet* f_2 such that $p_{2i1} = (x_6, y_6, z_6)$, $p_{2i2} = (x_7, y_7, z_7)$ and $p_{2j1} = (x_8, y_8, z_8)$. Assume that the two *facet*s are not *coplanar*.

To give the semantics of this operation, we start by defining a function that gives the *point* set for the line formed by the supporting planes of the *facet*s. We will need this function when we put into lexicographic order the boundary *point*s of *segment*s in the intersection. In the equations that define this *point* set, $\Gamma 1$, $\Delta 1$, $\Phi 1$, $\Psi 1$, $\Gamma 2$, $\Delta 2$, $\Phi 2$ and $\Psi 2$ are the coefficients and constant terms in the general equations of the two supporting planes.

$$\begin{aligned} line: \quad & facet \times facet \rightarrow \{p \mid p \in point\} \\ line(f_1, f_2) = \{p = (x,y,z) \mid \forall t \in Q^{repr}, & \\ & x = x_0 + lt, y = y_0 + mt, z = z_0 + nt\} \end{aligned}$$

where

$$l = \begin{vmatrix} \Delta_1 & \Phi_1 \\ \Delta_2 & \Phi_2 \end{vmatrix}, \qquad m = \begin{vmatrix} \Phi_1 & \Gamma_1 \\ \Phi_2 & \Gamma_2 \end{vmatrix},$$

$$n = \begin{vmatrix} \Gamma_1 & \Delta_1 \\ \Gamma_2 & \Delta_2 \end{vmatrix},$$

$$x_0 = \frac{m \begin{vmatrix} \Psi_1 & \Phi_1 \\ \Psi_2 & \Phi_2 \end{vmatrix} - n \begin{vmatrix} \Psi_1 & \Delta_1 \\ \Psi_2 & \Delta_2 \end{vmatrix}}{l^2 + m^2 + n^2},$$

$$y_0 = \frac{n \begin{vmatrix} \Psi_1 & \Gamma_1 \\ \Psi_2 & \Gamma_2 \end{vmatrix} - l \begin{vmatrix} \Psi_1 & \Phi_1 \\ \Psi_2 & \Phi_2 \end{vmatrix}}{l^2 + m^2 + n^2},$$

$$z_0 = \frac{l \begin{vmatrix} \Psi_1 & \Delta_1 \\ \Psi_2 & \Delta_2 \end{vmatrix} - m \begin{vmatrix} \Psi_1 & \Gamma_1 \\ \Psi_2 & \Gamma_2 \end{vmatrix}}{l^2 + m^2 + n^2},$$

$$\Gamma_1 = \begin{vmatrix} y_4 - y_3 & z_4 - z_3 \\ y_5 - y_3 & z_5 - z_3 \end{vmatrix},$$

$$\Delta_1 = \begin{vmatrix} x_4 - x_3 & z_4 - z_3 \\ x_5 - x_3 & z_5 - z_3 \end{vmatrix},$$

$$\Phi_1 = \begin{vmatrix} x_4 - x_3 & y_4 - y_3 \\ x_5 - x_3 & y_5 - y_3 \end{vmatrix},$$

$$\Psi_1 = \Gamma_1 x_3 + \Delta_1 y_3 + \Phi_1 z_3,$$

$$\Gamma_2 = \begin{vmatrix} y_7 - y_6 & z_7 - z_6 \\ y_8 - y_6 & z_8 - z_6 \end{vmatrix},$$

$$\Delta_2 = \begin{vmatrix} x_7 - x_6 & z_7 - z_6 \\ x_8 - x_6 & z_8 - z_6 \end{vmatrix},$$

$$\Phi_2 = \begin{vmatrix} x_7 - x_6 & y_7 - y_6 \\ x_8 - x_6 & y_8 - y_6 \end{vmatrix},$$

and $\quad \Psi_2 = \Gamma_2 x_6 + \Delta_2 y_6 + \Phi_2 z_6.$

Next, we get the places where the *facet*s intersect in a single *point* which we denote as the set P_p. For this purpose, we define the set S_p which is the set of all boundary *segment*s in one of the *facet*s that make contact with the other *facet* at the places where the *facet*s *intersect* in a *point*. Let

$$\begin{aligned} S_p = \{s \in f_1 \cup f_2 \mid & \\ & \text{(i) } f_1 \; touches \; f_2 \vee f_1 \; tangent \; f_2 \\ & \text{(ii) } \exists i \in \{1,2\} : s \; touches \; f_i \vee s \; tangent \; f_i\} \end{aligned}$$

and let

$$\begin{aligned} P_p = & \{intersection(s, f_1) \mid s \in S_p\} \\ & \cup \{intersection(s, f_2) \mid s \in S_p\}. \end{aligned}$$

We will eventually put the *point* set P_p in the *intersection* of the two *facet*s. Before we do that, however, we determine the places where the *facet*s intersect in a *segment*. This is the set P_s. To do so, we define the set S_s which contains all boundary *segment*s in one of the *facet*s that *intersect* the other *facet* at the places where the *facet*s *intersect* in a *segment*. Let

$$\begin{aligned} S_s = \{s \in f_1 \cup f_2 \mid & \\ & \text{(i) } f_1 \; meets \; f_2 \vee f_1 \; intersects \; f_2 \\ & \quad \vee f_1 \; incident \; f_2 \vee f_1 \; brushes \; f_2 \\ & \text{(ii) } \exists i \in \{1,2\} : s \; meets \; f_i \vee s \; intersects \; f_i \\ & \quad \vee s \; touches \; f_i \vee s \; incident \; f_i \\ & \quad \vee s \; tangent \; f_i\} \end{aligned}$$

and let

$$\begin{aligned} P_s = & \{intersection(s, f_1) \mid s \in S_s\} \\ & \cup \{intersection(s, f_2) \mid s \in S_s\}. \end{aligned}$$

Lastly, take the *point*s P_s, reorder them in lexicographic order along the $line(f_1, f_2)$. We denote the reordered set as P_o. It must have an even number of *point*s because all of its *point*s come from places where the boundary of one *facet* must go through the other *facet* and, therefore, must come back through the other *facet* again. Thus,

$$\begin{aligned} P_o = \{\{p_1, p_2, ..., p_k\} \mid & \\ & k \text{ is even} \wedge \forall i \in \{1,2,...,k\} : p_i \in P_s\}. \end{aligned}$$

Furthermore, every pair of *point*s represents a *segment* in the *intersection* of the two *facet*s. Thus, for $P_o = \{p_1, p_2, ..., p_k\}$, let

$$S_o = \{(p_1, p_2), (p_3, p_4), ..., (p_{k-1}, p_k)\}.$$

S_o contains the *segment*s in the *intersection* of the *facet*s. Thus, recalling that P_p is the set of single *point*s in the *intersection*,

$$intersection(f_1, f_2) = P_p \cup S_o.$$

This definition of *intersection* insures that the *intersection* is empty when the two *facet*s do not *meet*, *intersect*, *touch* or *brush* or are not *incident* or *tangent*.

3.3 The Primitive *solid*

As we begin the discussion of the geometric primitive *solid*, we emphasize that while the abstract data type *volume* must provide for holes, *solid* does not need them. A 3D object with a hole is not a primitive. Rather, we would represent such an object in computer memory by using two primitives.

Just like we used *segment*s to define *facet*, we now use *facet*s to define *solid*. The *segment*s that define a *facet* essentially give the boundary of the *facet* and we assume the *facet* to be inside the boundary. The *segment*s are connected in a way that they *meet* in pairs so that they enclose the area which is the *facet*. Similarly, we shall define a *solid* by using connected *facet*s that *meet* at boundary *segment*s so that they enclose a section of 3D space. Thus, the enclosing *facet*s are the boundary of a *solid*. Furthermore, the boundary *facet*s and the 3D space enclosed by them are the *solid*. Note that not just any collection of connected *facet*s defines a *solid*. If a collection of connected *facet*s does not enclose a section of 3D space, it is simply a *surface*. For a collection of *facet*s to define a *solid*, every *facet* in the collection must *meet* exactly one other *facet* in the collection at everyone of its boundary *segment*s. No boundary *segment* in a *facet* can fail to be involved in *meeting* another *facet*. More formally,

$$
\begin{aligned}
solid = \{\{ & f_0, f_1, ..., f_n\} \mid \\
& \text{(i)} \quad \forall i \in \{0,...,n\} : \ f_i \in facet \\
& \qquad \text{with } f_i = \{s_{i0}, s_{i1}, ..., s_{i,m-1}\} \\
& \text{(ii)} \quad \forall i \in \{0,...,n\}, \forall s_{ik} \in f_i, \\
& \qquad \exists j \in \{0,...,n\} \text{ with } j \neq i : \\
& \qquad\quad (f_i \ meets \ f_j) \\
& \qquad\qquad \wedge \ s_{ik} \in intersection(f_i, f_j) \\
& \text{(iii) No more than two of } f_0, f_1, ..., \\
& \qquad\quad \text{and } f_n \ meet \text{ at any one } segment\}.
\end{aligned}
$$

There is such a large proliferation of operations involving *solid*s that, by themselves, they would fill a large part of this paper. There are operations between *point*s and *solid*s, between *segment*s and *solid*s, between *facet*s and *solid*s and between two *solid*s. Predicates for *point*s and *solid*s are similar to what we have seen between *point*s and the other two primitives: *on*, *in* and *out*. Examples of operations for *segment*s and *solid*s are *meets*, *intersects*, *touches*, *incident*, *tangent*, *inside* and *outside* (*disjoint*). Examples of operations for *facet*s and *solid*s are *meets*, *intersects*, *cuts*, *touches*, *incident*, *tangent*, *brushes*, *inside*, *outside* (*disjoint*), *cutabove* and *cutbelow*. Finally, examples of operations involving two *solid*s are *equals*, *not equals*, *meets*, *joins*, *intersects*, *incident*, *connects*, *links*, *tangent*, *brushes*, *inside*, *contains*, *disjoint* and *intersection*.

Due to space limitations, however, we cannot include the definitions of operations of *solid*. For the same reason, we also cannot include the specification of directional predicates such as *above*, *below*, *rightside*, *leftside*, *before* and *behind*.

Thus, the four geometric primitives in 3D are *point*, *segment*, *facet* and *solid*. The definition of *point* uses a rational coordinate. Furthermore, we use *point*s to build a *segment*, *segment*s to build a *facet* and *facet*s to build a *solid*. Therefore, all four geometric primitives, *point*, *segment*, *facet* and *solid* have a rational specification. We are about to show in Section 4 that the four primitives are closed under intersection up to the limits of computer memory. Once we have done so, it paves the way to implement a 3D spatial database system that relies on arbitrary precision rational arithmetic rather than on realms or dual grids. Hence, we completely eliminate the need to use realms or dual grids. In effect, a rational implementation results in an arbitrarily fine grid. This introduces greater exactness of computation to a spatial database system.

Because of the closure problems of extending realms to 3D that we noted in Section 2, comparison of the efficiency of a rational implementation of a 3D system is not crucial when considering its robustness. We still state, however, that we believe that a rational implementation saves execution time when compared with a hypothetical extension of realms to 3D. This is because a rational implementation does not need the redrawing and enveloping process required by realms and dual grids, not to speak of the pre- and post-processing algorithms required by the enhanced form of dual grids. Most of the operations involving our rationally specified primitives involve only mathematical formulas that require constant time. This benefit is partially offset because we assume that arithmetic will be performed by an arbitrary precision rational number system where the lengths of numbers can vary. We feel, though, that the problems of realms and dual grids that we have previously mentioned more than outweigh the complexities introduced by arbitrary precision rational arithmetic. Therefore, greater robustness results from the rational implementation of geometric primitives.

4. THE RATIONAL CLOSURE OF INTERSECTIONS IN THREE DIMENSIONS

We now proceed to discuss the closure of 3D geometric primitives under intersection operations. We will prove that intersections involving rationally specified *segment*s and *facet*s are rational. Taken together with the closure of rational arithmetic, the closure of intersections means that a closed system results, as far as spatial operations are concerned, up to the limit of the precision with which computer memory can store rational numbers. This provides for a large amount of robustness in a 3D spatial database system. We need not consider other spatial operations such as union, difference and symmetric difference as the proofs of the closures of those operations follow from the closure of intersections.

Note that considering the intersection of a line and a plane or two lines amounts to considering the intersection between a *segment* and a *facet* or between two *segment*s. However, the intersection of two *facet*s is different from the intersection of two planes. Thus, the specific intersections we will consider are of a line and a plane, of two lines and of two *facet*s. All other intersections, including all that involve *solid*s, can be easily derived using these three results.

4.1 The Intersection of a Line and a Plane

We first prove that the intersection of a rationally specified line and plane in 3D is rational. We do this first because the proofs involving the other intersections depend on this one. Indeed, once this proof is given, the proofs for the other two intersections follow fairly quickly.

Let (x_1, y_1, z_1) and (x_2, y_2, z_2) be rational points defining a line. We will work with the equations of a 3D line in parametric form. The parametric equations of a line are

$$x = x_1 + lt \qquad y = y_1 + mt \qquad z = z_1 + nt \tag{1}$$

where l, m, n are the direction cosines for the line defined by (x_1, y_1, z_1) and (x_2, y_2, z_2). The direction cosines are

$$l = \frac{x_2 - x_1}{d} \qquad m = \frac{y_2 - y_1}{d} \qquad n = \frac{z_2 - z_1}{d} \tag{2}$$

$$\text{where} \quad d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}. \tag{3}$$

The intercept form of the definition of a plane in 3D is

$$\frac{x}{a} + \frac{y}{b} + \frac{z}{c} = 1 \tag{4}$$

where a, b and c are the x, y and z intercepts of the plane.

Now, suppose (x_0, y_0, z_0) is the point where the line and the plane intersect. Let t_0 be the t in the parametric equations at the intersection point (x_0, y_0, z_0). Then, the equations 1 and 4 at the intersection point (x_0, y_0, z_0) are

$$x_0 = x_1 + lt_0 \qquad y_0 = y_1 + mt_0 \qquad z_0 = z_1 + nt_0 \tag{5}$$

$$\frac{x_0}{a} + \frac{y_0}{b} + \frac{z_0}{c} = 1. \tag{6}$$

The resulting equations 5 and 6 are a system of four equations in the four variables x_0, y_0, z_0, t_0 which correspond to the intersection point (x_0, y_0, z_0). We solve it by substitution starting with plugging the equations 5 into equation 6.

$$\frac{x_1 + lt_0}{a} + \frac{y_1 + mt_0}{b} + \frac{z_1 + nt_0}{c} = 1$$

$$\Rightarrow \quad \frac{lt_0}{a} + \frac{mt_0}{b} + \frac{nt_0}{c} = 1 - \frac{x_1}{a} - \frac{y_1}{b} - \frac{z_1}{c}$$

$$\Rightarrow \quad t_0\left(\frac{l}{a} + \frac{m}{b} + \frac{n}{c}\right) = 1 - \frac{x_1}{a} - \frac{y_1}{b} - \frac{z_1}{c}$$

$$\Rightarrow \quad t_0 = \frac{1 - \frac{x_1}{a} - \frac{y_1}{b} - \frac{z_1}{c}}{\frac{l}{a} + \frac{m}{b} + \frac{n}{c}}$$

Now we can plug the resulting formula for t_0 back into equations 5 to get the coordinates at the intersection point (x_0, y_0, z_0).

$$x_0 = x_1 + l\left(\frac{1 - \frac{x_1}{a} - \frac{y_1}{b} - \frac{z_1}{c}}{\frac{l}{a} + \frac{m}{b} + \frac{n}{c}}\right) \tag{7}$$

$$y_0 = y_1 + m\left(\frac{1 - \frac{x_1}{a} - \frac{y_1}{b} - \frac{z_1}{c}}{\frac{l}{a} + \frac{m}{b} + \frac{n}{c}}\right) \tag{8}$$

$$z_0 = z_1 + n\left(\frac{1 - \frac{x_1}{a} - \frac{y_1}{b} - \frac{z_1}{c}}{\frac{l}{a} + \frac{m}{b} + \frac{n}{c}}\right) \tag{9}$$

The only variables that can produce irrational results in the equations 7 to 9 are l, m, n, the direction cosines, and a, b and c, the x, y and z intercepts of the plane. Everything else in these equations involves coordinates that we assume to be rational.

Let us first look at the direction cosines l, m, n in the equations 7 to 9. The thing that suggests the possibility of irrational results from the direction cosines are the square roots seen in the equations 2 and 3 that define the direction cosines. However, the following discussion shows that the square roots all cancel each other. We will use d from equation 3 to represent the square root. Observe what happens to the denominator of the equations 7 to 9.

$$\frac{l}{a} + \frac{m}{b} + \frac{n}{c} = \frac{(x_2 - x_1)/d}{a} + \frac{(y_2 - y_1)/d}{b} + \frac{(z_2 - z_1)/d}{c}$$

$$= \frac{1}{d}\left(\frac{(x_2 - x_1)}{a} + \frac{(y_2 - y_1)}{b} + \frac{(z_2 - z_1)}{c}\right)$$

Thus, $\frac{1}{d}$ factors out of the denominator of all of the equations for the intersection coordinates (equations 7 to 9). Now, note that there is a direction cosine as a coefficient to the fractions in all these equations. Thus, the numerator in each equation for a coordinate also has a $\frac{1}{d}$ factor. Thus, the two factors cancel each other eliminating the square root d from the intersection coordinate equations. Thus, the direction cosines l, m, n do not introduce an irrational component to the intersection coordinates.

Next, we will show that the x, y and z intercepts a, b and c must be rational. Let $(x_3, y_3, z_3), (x_4, y_4, z_4)$ and (x_5, y_5, z_5) be rational coordinates defining the plane. Then, plugging these coordinates into equation 4 we get

$$\frac{x_3}{a} + \frac{y_3}{b} + \frac{z_3}{c} = 1$$

$$\frac{x_4}{a} + \frac{y_4}{b} + \frac{z_4}{c} = 1$$

$$\frac{x_5}{a} + \frac{y_5}{b} + \frac{z_5}{c} = 1.$$

This is a system of three linear equations for $\frac{1}{a}, \frac{1}{b}$ and $\frac{1}{c}$. Therefore, Cramer's Rule gives as solutions

$$\frac{1}{a} = \frac{\begin{vmatrix} 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \\ 1 & y_5 & z_5 \end{vmatrix}}{\begin{vmatrix} x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ x_5 & y_5 & z_5 \end{vmatrix}}, \qquad \frac{1}{b} = \frac{\begin{vmatrix} x_3 & 1 & z_3 \\ x_4 & 1 & z_4 \\ x_5 & 1 & z_5 \end{vmatrix}}{\begin{vmatrix} x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ x_5 & y_5 & z_5 \end{vmatrix}}, \tag{10}$$

$$\frac{1}{c} = \frac{\begin{vmatrix} x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \\ x_5 & y_5 & 1 \end{vmatrix}}{\begin{vmatrix} x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ x_5 & y_5 & z_5 \end{vmatrix}}. \tag{11}$$

Determinants are completely evaluated using additions, subtractions and multiplications. Also, we use a division of determinants to determine $\frac{1}{a}, \frac{1}{b}$ and $\frac{1}{c}$. Since the rational numbers are closed under arithmetic, equations 10 and 11 show that a, b and c must be rational.

So, we have shown that the direction cosines l, m, n do not introduce an irrational component to the intersection coordinates and that x, y and z intercepts of the plane must be rational. Thus, the equations 7 to 9 give rational coordinates of the intersection. So, a line and plane in 3D defined using rational numbers must have a rational intersection.

4.2 The Intersection of Two Lines

Next, we prove that two rationally specified intersecting lines in 3D have a rational intersection. Suppose we have two intersecting lines in 3D which are defined using rational points. Let (x_1, y_1, z_1) and (x_2, y_2, z_2) be the rational points defining one of the lines and let (x_3, y_3, z_3) and (x_4, y_4, z_4) be the rational points defining the other line. Furthermore, let (x_0, y_0, z_0) be the point where they intersect. Choose any other rational point (x_5, y_5, z_5) that is not coplanar with the two lines. Then, the points (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_5, y_5, z_5) rationally specify a plane. Furthermore, the line defined by (x_3, y_3, z_3) and (x_4, y_4, z_4) intersects this plane at (x_0, y_0, z_0). Both the plane and the line are rationally specified. Thus, by the result of the previous subsection (Subsection 4.1), (x_0, y_0, z_0) must be rational. This point, however, is the point at which the two lines intersect. Thus, the two lines have a rational intersection.

4.3 The Intersection of Two *facet*s

Finally, we prove that two *facet*s have a rational intersection. First of all, note that both *facet*s involved in such an intersection have rationally defined supporting planes. This is because we defined the endpoints of the *segment*s forming the boundaries of *facet*s to have rational coordinates. A *facet* must have at least three of these rational endpoints (and, in fact, probably has many more). Thus, the supporting planes are rationally defined.

Secondly, recall that there can be both *point*s and *segment*s in the intersection of two *facet*s. Therefore, when considering the intersection of two *facet*s, there are two cases. There is the case when there are only points in the intersection and the case when the intersection includes segments. When there are only points in the intersection, they must occur at places where the *facet*s *touch* or are *tangent*. When an intersection point is produced because the *facet*s *touch*, it must occur at the endpoint of a rationally defined *segment*. The endpoints of *segment*s are *point*s and, therefore, rational. When an intersection point is where the two *facet*s are *tangent*, it must occur where two *segment*s of the *facet*s *intersect* or *touch*. In either situation, the previous subsection (Subsection 4.2) shows that the intersection is rational. Thus, the points in the intersection of the *facet*s are rational. Observe that, when there are more than two of these intersection points, they must be on the same line. This line is the intersection of the supporting planes of the two *facet*s. The line of intersection of the supporting planes is therefore rationally defined (i.e., has at least two rational coordinates).

Suppose there are one or more line segments at the intersection of the *facet*s. To show that the intersection is rational amounts to showing that the supporting line of the line segments formed by the intersection of the *facet*s is rationally defined. Two boundary *segment*s of the *facet*s must intersect the supporting line at separate points. Now, suppose both intersecting boundary *segment*s of a *facet* cross the other *facet* at an endpoint. These two endpoints are rational. Since they lie on the supporting line of the intersection of the *facet*s, the line is rationally defined.

Let us assume therefore that the intersections do not occur at an endpoint of the *segment*s. Thus, they occur in the interior of the two *segment*s crossing the *facet*s. Both *segment*s and both *facet*s are rationally defined. So, by Subsection 4.1, the points at the intersection of the *segment*s and the supporting planes of the *facet*s are rational. Both of these points, however, are in the supporting line at the intersection of the *facet*s. Thus, the supporting line is rationally defined implying that the *facet*s have a rational intersection. (Now, the situations where one *segment* intersects a *facet* at an endpoint and the other in its interior follow from these results.)

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have given a design of robust three-dimensional geometric primitives. Their robustness comes from the use of rational numbers to specify them. We have compared our design for the primitives with extensions from two dimensions of simplicial complexes, realms and dual grids (near the beginning and at the end of Section 3). This has included reviewing the potential of rational arithmetic and discussing the amount of representable numbers, time requirements and the rational closure of intersections of the primitives.

In the future, we need to complete the definitions of the operations for the geometric primitives, especially those involving *solid*, those that yield another primitive as their result and those that are directional predicates. We also need to investigate the storage requirements of our primitives compared to 3D extensions of realms and dual grids. Thirdly, we have developed a rational number system that allows number representations of arbitrary, finite length and called it RATIO. We have not discussed RATIO in this paper. Thus, in the future, we need to document and present RATIO. Finally, we need to use our primitives to implement the abstract three-dimensional spatial data types in a spatial database system.

6. REFERENCES

[1] C. Batut, K. Belabas, D. Bernardi, H. Cohen, and M. Olivier. *User's Guide to PARI/GP*. The PARI Group, `http://pari.math.u-bordeaux.fr/`, 2000.

[2] C. Burnikel, R. Fleischer, K. Mehlhorn, and S. Schirra. Efficient Exact Geometric Computation Made Easy. *Proc. of the 15th Annual Symp. on Computational Geometry*, pp. 341–350, Miami Beach, Florida, USA, June 13-16 1999.

[3] J. A. Cotelo Lema and R. H. Güting. Dual Grid: A New Approach for Robust Spatial Algebra Implementation. *GeoInformatica*, 6(1):57–76, 2002.

[4] F. Dumortier, M. Gyssens, L. Vandeurzen, and D. Van Gucht. On the Decidability of Semi-Linearity for Semi-Algebraic Sets and Its Implications for Spatial Databases. *Proc. of the 16th ACM SIGACT - SIGMOD - SIGART Symp. on Principles of Database Systems*, pp. 68–77, Tucson, Arizona, USA, 1997.

[5] F. Geerts and B. Kuijpers. Linear Approximation of Planar Spatial Databases Using Transitive-Closure Logic. *Proc. of the 19th ACM SIGMOD - SIGACT - SIGART Symp. on Principles of Database Systems*, pp. 126–135, 2000.

[6] R. H. Güting. An Introduction to Spatial Database Systems. *VLDB Journal*, 3(4):357–399, 1994.

[7] J. Keyser, J. M. Rojas, and K. Ouchi. The Exact Rational Univariate Representation and Its Application. In D. Duffa, R. Janardan, and M. Smid, editors, *AMS/DIMACS Volume on Computer Aided Design and Manufacturing*. American Mathematical Society/Center for Discrete Mathematics and Computer Science, 2005.

[8] M. Schneider. *Spatial Data Types for Database Systems - Finite Resolution Geometry for Geographic Information Systems*, volume LNCS 1288. Springer-Verlag, Berlin Heidelberg, 1997.

[9] M. Schneider and B. E. Weinrich. An Abstract Model of Three-Dimensional Spatial Data Types. *Proc. of the 12th ACM Int. Symp. on Advances in Geographic Information Systems (ACM GIS 2004)*, pp. 67–72, 2004.

[10] M. Tommila. *apfloat: A C++ High Performance Arbitrary Precision Arithmetic Package*. World Wide Web, `http://www.apfloat.org/apfloat/2.40/`, February 22 2003.

[11] C. K. Yap. Towards Exact Geometric Computation. *Computer Geometry: Theory and Applications*, 7(1-2):3–23, 1997.

[12] S. Zlatanova. On 3D Topological Relationships. *Int. Workshop on Database and Expert System Applications*, pp. 913–919, 2000.