Chapter 15
Mathematically provable correct implementation of integrated 2D and 3D representations

Rodney Thompson[1,2] and Peter van Oosterom[1]

Abstract

The concept of the 'Regular Polytope' has been designed to facilitate the search for a rigorous closed algebra for the query and manipulation of the representations of spatial objects within the finite precision of a computer implementation. It has been shown to support a closed, complete and useful algebra of connectivity, and support a topology, without assuming the availability of infinite precision arithmetic. This paper explores the practicalities of implementing this approach both in terms of the database schema and in terms of the algorithmic implementation of the connectivity and topological predicates and functions. The problem domains of Cadastre and Topography have been chosen to illustrate the issues.

15.1 Introduction

One of the perennial problems in the spatial data industry is interchange of data. It is common for considerable outlay of time and effort (and funds) to be consumed in re-formatting and revalidating data, largely due to the lack of formal definition of spatial primitives and functions. For example, there is no agreed normative meaning of the 'equals' predicate when applied to geometric objects. Definitions of validity are in general defined by implementers – for example (Kazar *et al.* 2007). In addition, the language of spatial databases is couched in terms of the language of mathematics, with operations named

[1]Delft University of Technology, OTB, section GIS Technology,
Jaffalaan 9, 2628 BX the Netherlands
[2]Department of Natural Resources and Water,
Queensland, Australia
Rod.Thompson@nrw.qld.gov.au, oosterom@tudelft.nl

'union' and 'intersection' and using vector-like representations. This naturally leads to the impression that the representations form a topological space, and/or a vector space, which unfortunately is not the case. Generally speaking, the rigorous mathematics of the definition of spatial objects ends outside the database representation, which is only an approximation of the theoretical formalism used to define it. This leads to many cases where unexpected breakdowns in logic occur (Franklin 1984; Hölbling *et al.* 1998; Thompson 2004; Thompson 2007) due to the finite digital arithmetic implemented by computers and the necessary rounding or approximations.

By contrast, the Regular Polytope (Thompson 2005a) has been shown to be a promising candidate for the rigorous representation of geometric objects, in a form that is computable using the finite arithmetic available on digital computers. In order to explore practical issues in the Regular Polytope representation, a series of objects have been written in the Java programming language, and stored in a basic form using an Informix database.

The class of test data chosen was Cadastral property boundaries, since large volumes of data was available, and this topic presents some unique challenges, in particular, the mix of 3D and 2D data that is involved (Stoter 2004). The Regular Polytope representation provides a particularly elegant solution to this issue.

This paper describes alternative database schemas to support the implementation, and discusses some of the practical considerations that arise. This gives an indication of the requirements of a full implementation, and what further development is needed. Also discussed are some of the practicalities involved in converting geo-information to and from the regular polytope form from the conventional vertex representations. First, two different approaches to representation of geo-objects based on regions (resp. boundary-based and boundary-free) and two different application areas (resp. Cadastre and Topography) are discussed. This introductory section ends with an overview of the complete paper.

15.1.1 Boundary Representations and Mereology

The conventional description of a geometric object partitions space into the interior, boundary and exterior of that object. The Egenhofer 9-matrix provides an exhaustive description of possible binary relationships (between two objects), and is frequently used in situations where a clear definition of a complex relationship is required (Egenhofer and Herring 1994). It consists of a 3×3 matrix of Boolean values representing the overlap of the interiors, boundaries and exteriors of the object pairs.

Alternately, there is an advantage in taking a Mereological approach to spatial logic, thus avoiding some of the distinctions between finite and infinite (smooth) sets. In this way, concepts such as 'set contacts set' and 'set includes

set' move easily from the infinite to the finite realm, whereas the definition of a region as a collection of points defined by a boundary set of points (Smith 1997) raises the issue of the density of the representation. Briefly, the distinction is that point-set topology defines regions as sets of points, with boundaries being a separate set of points, either included or not depending whether the region is closed or open. The Mereological approach is to treat the region as the fundamental concept, with the boundary arising as a natural consequence of the region being limited in extent (Borgo *et al.* 1996).

Although the concept of a boundary as a point-set is useful in describing mathematical abstractions, it has no counterpart in the real world. '... it is nonsense to ask whether a physical object occupies an open or a closed region of space, or who owns the mathematical line along a property frontier' (Lemon & Pratt 1998a, page 10).

Similarly, in the computer representation, a boundary point set is problematic. On a line between two points in a 2D vector representation based on integers or floating point coordinates, it has been shown that in about 60% of cases, there will be no representable points at all lying on the line (apart from the end points)(Castellanos 1988; Thompson 2007). Thus, if a region is defined by a conventional polygon, the point set representing its boundary will consist of the vertices plus an insignificant, but highly variable number of points.

Thus, in summary, the concept of a boundary as a set of points does not sit well in the real world, or in the computer representation. It might be thought that a boundary would be needed to ensure a definition of such concepts as tangential contact, but this is not the case. An alternate approach comes from the Region Connection Calculus (RCC), defining such predicates without invoking boundary point-sets (Randell *et al.* 1992). Of the 512 possible relationships that can be defined in the Egenhofer 9 matrix, Zlatanova (2000) has shown that only 8 are possible between objects of the same dimensionality as the embedding space in 2D or 3D. These relationships can be directly modelled by the 8 relations of the RCC.

15.1.2 Application to Cadastre

There are an interesting set of specific requirements in the realm of Cadastral data. Here, the fundamental item of interest is the land parcel. While these parcels are defined by measurement (survey) of their boundaries, there is no question of ownership of the boundary itself. Thus a boundary-free representation is ideal.

There is a growing need to represent 3D 'land' parcels (space) in a cadastre. These include strata parcels, unit rights, properties overhanging roads or rail, etc. (Thompson 2005b; Tarbit and Thompson 2006), but comprise a small minority of cadastral parcels in any jurisdiction, so that there is a

strong argument for integrated 2D and 3D representations in the one database (Stoter and van Oosterom 2006). As pointed out by Stoter (2004), the so-called 2D parcels are really volumetric regions in space. It is merely the definition that is 2D (defining a prismatic volume, with undefined vertical limits), so it should be possible to evaluate any functions and predicates on mixed 2D/3D parcels. E.g. in figure 1, it can be asked whether C intersects D (which it does, since it encroaches on the volume above D.

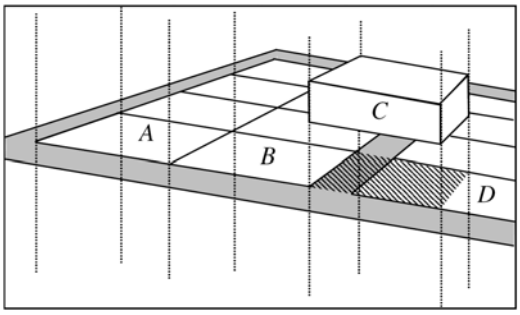

Fig. 15.1 Mixing 3D and 3D Cadastre. C is a volumetric parcel, with its projection onto the ground shown hashed

15.1.3 Topography

The representation of the topography of the earth's surface has some similarity in requirements. The vast majority can be adequately represented by treating the elevation of the surface as a function of two variables (the x and y coordinates) – with only one elevation applying to each point, but this precludes the representation of vertical or overhanging cliffs, natural arches or many man-made structures (see figure 2). In the same way as with the cadastre, the vast majority of practical topographic data does not require full 3D, and could be modelled as a single-valued function of two variables - e.g. elevation(x, y). Only minority of specific regions need true 3D; e.g. a Tetrahedral Network TEN (Verbree *et al.* 2005).

15.1.4 Structure of the Paper

The remainder of the paper is structured as follows:

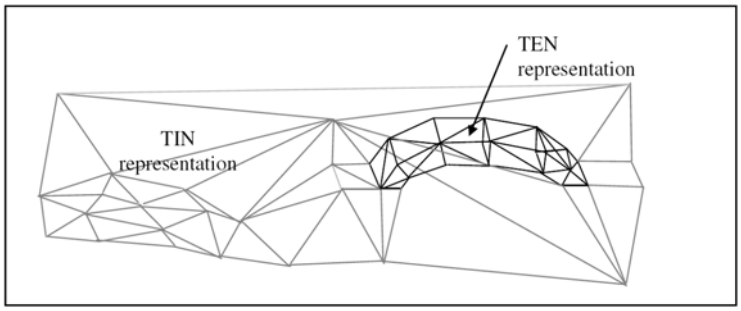

Fig. 15.2 Combination of TIN and TEN representations

- Section 2 defines the concept of domain-restricted rational numbers, and the regular polytope concept.
- Section 3 discusses various database structures that could potentially be used to implement the approach.
- Section 4 discusses a set of Java language classes that have been written to assist in the investigation of, and demonstrate the algebra. This includes some test results with real data.
- Section 5 explores some issues that arise when converting from conventional vertex defined representations.

15.2 The Regular Polytope

In order to provide the required rigorous logic in 2D and 3D, within the computational arithmetic, the concept of a regular polytope has been proposed (Thompson 2005a; 2007) and will be summarized below. In this paper, the terminology and mathematics of 3D will be used, since the equivalent 2D case is usually obvious. Most of the illustrations are 2D for ease of drawing. This section contains a brief definition of the regular polytope representation (Sections 2.1 to 2.3), with some of its properties (Section 2.4), in comparison with more conventional approaches (Section 2.5).

15.2.1 Definition of domain-restricted rational numbers and points

Given two large integers N' and N'', a domain-restricted (dr)-rational number r can be defined with the interpretation I/J, with I, J integers ($-N'' \leq I \leq N''$, $0 < J \leq N'$. The name 'domain-restricted rational' (dr-rational) is used

because the values of I and J are constrained to a finite domain of values. Like floating point numbers, dr-rational numbers do not form a field[1] (in contrast to the true rational numbers) (Weisstein 2005), and therefore cannot span a vector space.

In 3D, a dr-rational point is defined as an ordered triple of dr-rational numbers $p = x,y,z$, with x,y and z representing the Cartesian co-ordinate values. Note that there are also counter-intuitive properties possessed by dr-rational points – e.g. it cannot be assumed that the mid-point between two dr-rational points is a dr-rational point. The advantage possessed by the dr-rational representation is that it is directly implementable in computer hardware.

15.2.2 Half Space Definition

In 3D a half space[2] $H(A,B,\ C,\ D)(A,\ B,\ C,\ D$ integers, $-M < A,B,C < M$, $-3M^2 < D < 3M^2$ is defined as the set of dr-rational points $p(x,y,z)$: $-M \leq x,y,z < M$, for which computation of the following inequalities yields these results:

- $(Ax+By+Cz+D) > 0$ or
- $[(Ax+By+Cz+D) = 0$ and $A > 0]$ or [3]
- $[(By+Cz+D) = 0$ and $A = 0$ and $B > 0]$ or
- $[(Cz+D) = 0$ and $A = 0, B = 0$ and $C > 0]$,

where M is the limit of values allowed for point representations. The choice of M is dependant on the size of the region to be covered as a multiple of the unit of resolution. The values of N' and N" follow from M as:

- for 2D applications $N' = 2M^2$, $N'' = 4M^3$.
- for 3D applications use $N' = 6M^3$, $N'' = 18M^4$.

$H(0,0,0,0)$ is not a permitted half space.

The complement of a half space is defined as:

- $\overline{H} = (-A,-B,-C,-D)$ where $H = (A,B,C,D)$.

---

[1] The set of rational numbers $\mathbf{Q}$ obey the field axioms, including the closure axioms (e.g. $a \in \mathbf{Q}$, $b \in \mathbf{Q} \Rightarrow a.b \in \mathbf{Q}$). This is not the case for dr-rational or floating point numbers.

[2] The equivalent 2D object is known as a 'half plane' which (for convenience) is defined by the integer parameters A,B and D.

[3] This form of the definition with four parts, rather than just $(A.x+B.y+C.z+D) > 0$, is chosen so as to ensure a boundary-free representation. In effect, this eliminates all boundary points.

15.2.3 Regular Polytope Definition

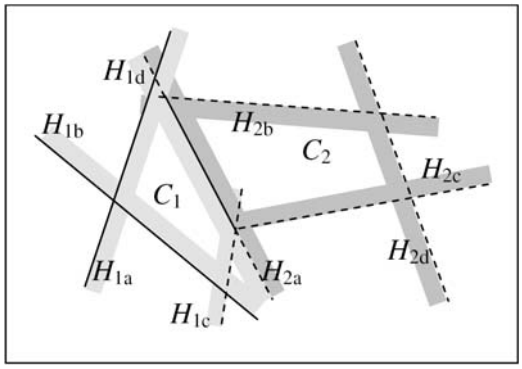

Fig. 15.3 Regular polytope O defined as the union of convex polytopes C_1 and C_2

A regular polytope O is defined as the union of a finite set of (possibly overlapping) 'convex polytopes' (C_1 and C_2 in figure 3), which are in turn defined as the intersection of a finite set of half spaces (H_{1a} to H_{1d} and H_{2a} to H_{2d})(Thompson 2005a; Thompson 2007). Note – a regular polytope may consist of disconnected parts, and parts may overlap. The natural definitions of the union, intersection, and complement of regular polytopes is used, so that the meanings are exactly equivalent to the point set interpretations. E.g.:

$\forall p : p \in O_1 \cup O_2 \Leftrightarrow p \in O_1 \vee p \in O_2$ The concept of connectivity in this approach is based on two forms – so-called 'weak connectivity' C_a in figure 4, and 'strong' or C_b connectivity. The strongest form of connection is actual overlap, so that OV $\Rightarrow C_b \Rightarrow C_a$. This can also be expressed in dimensional terms (Clementini *et al.* 1993), such that in a 3D space, Ca $\equiv$ 0D or 1D meet, Cb $\equiv$ 2D meet, and OV $\equiv$ 3D meet.

15.2.4 Properties of the Regular Polytope

It is relatively simple to show that the space of regular polytopes is a topology (Thompson 2005c), based on the definition of regular polytope as an open set. It is readily apparent that for any regular polytope O, $\forall p : p \in O \Leftrightarrow p \notin \overline{O}$, and that[4] $O \cup \overline{O} = O_\infty$ and $O \bigcap \overline{O} = O_\Phi$. Thus no boundary points exist between $\overline{O}$ and O ($\forall p : p \in O \vee p \in \overline{O}$), in contrast with most conventional approaches

[4] O_φ and O_∞ are the empty and universal regular polytopes respectively such that $\forall p : p \notin O_\varphi$ and $p \in O_\infty$.

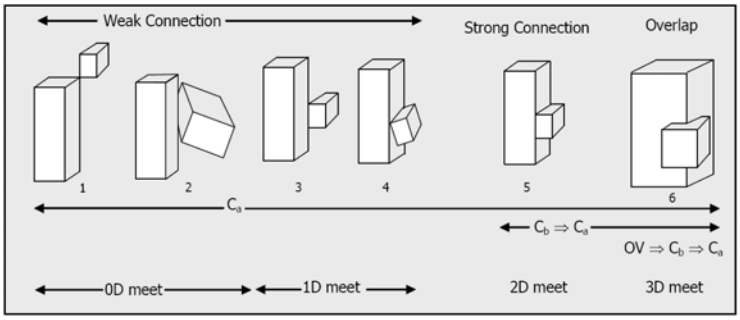

Fig. 15.4 Modes of Connectivity in 3D

where (in the mathematical model) space is partitioned into a region's interior R^o, exterior R^- and boundary ∂R, with $\forall p, p \in R^o \vee p \in R^- \vee p \in \partial R$. A further consequence of being a boundary free representation is that the axioms of a Boolean algebra (Weisstein 1999) are satisfied.

It has been shown (Thompson 2005c; Thompson 2007; Thompson and van Oosterom 2007) that the space of regular polytopes obeys the axioms of the Region Connection Calculus (Randell *et al.* 1992) based on the above definitions, and that it forms a Weak Proximity Space (Naimpally and Warrack 1970); and a Boolean Contact Algebra (Düntsch and Winter 2004). It is important to remember that it is the computational representation that satisfies the axioms, not an abstraction which is approximated by the computational representation. Thus it is possible to computationally apply the operations in any combination with complete confidence that no logic failure can result.

15.2.5 Vertex-based Representations

In two-dimensional applications, the 'Point/Line/Polygon' paradigm for the representation of spatial features is well entrenched, albeit with some significant variations (van Oosterom *et al.* 2004), and provides a degree of comfort in the user. This is spite of some serious difficulties in terms of rigorous definitions of concepts such as validity, and equality (Thompson 2005a). The available 3D structures take various forms (Arens *et al.* 2003), with no one having proved to be the best in all circumstances (Zlatanova *et al.* 2004).

In this paper, the term 'vertex based' representation is used to cover all ways to model spatial data based on point coordinates of vertices as the major determinants of the shape and position of the objects. The vertices are defined as points with coordinates *x,y,z*, while all other geometric objects are defined in terms of sets of vertices or higher order constructive objects. This describes virtually all other two and three dimensional spatial data models,

regardless of the level of topological encoding supported (Ellul and Haklay 2005). In contrast, but following a similar naming convention, the regular polytope in 3D would be called a 'surface based representation' and in 2D an 'edge based representation'. It can also be viewed as a restricted form of constraint-based representation.

15.3 The Proposed Database Structures

In implementing the concepts of the regular polytope, in a database management system, in order to manage large data volumes in a multi-user environment, several decisions need to be made. Principal amongst these is the decision as to how much redundant storage of information is to be tolerated, and more specifically whether 'topological encoding' is to be used. Section 3.1 discusses the basic data model, 3.2 briefly describes topological encoding, while 3.3 applies the concept to the regular polytope.

15.3.1 Discrete Polytope Encoding

This is perhaps the simplest structure, with some redundancy of storage, and no topological encoding. Each regular polytope is stored as a unit, containing its component convex polytopes and their defining half spaces.

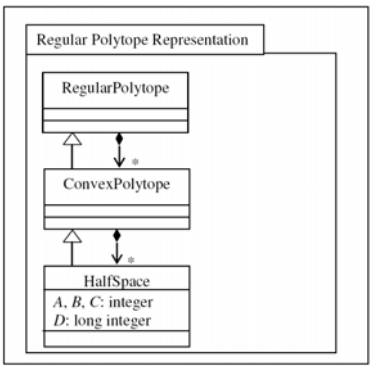

Fig. 15.5 The Regular Polytope model. See also (Thompson 2005a; Thompson 2007)

Figure 5 shows, in the unified Modelling Language (UML) (OMG 1997), a possible implementation of the regular polytope representation discretely encoded (in 3D). Key points in the interpretation of this diagram is that:

- A convex polytope is a specialization of regular polytope.
- A half space is a specialization of convex polytope (which means it is also a regular polytope).
- A regular polytope consists of zero or more convex polytopes. Note that zero implies the empty regular polytope.
- A convex polytope consists of zero or more half spaces. Note that zero implies the infinite convex polytope.
- A half space has the integer attributes A, B, C and D.
- No dr-rational numbers are stored in the database.
- A HalfSpace whose plane separates two convex polytopes will be stored twice (as an 'anti-equal' pair).

15.3.1.1 Model Restrictions

This model is the most basic, intended for demonstration purposes. In practice, additional classes would be added to improve speed and responsiveness. For example, a convex polytope might be associated with an approximate bounding rectangle, which is the basis for a spatial index, however the bounding box could be computed with a function and the spatial index created on the return value of that function. That is, the bounding box need not be stored if a functional spatial index is used. There are a number of issues remaining that apply to this base level model:

- The Regular Polytope storage mechanism differs from the more familiar vertex representations, and requires non-trivial conversion routines to allow interoperability.
- The calculations require the use of large precision integer arithmetic (but these large precision integers are not stored in the database).
- The storage requirements are approximately double those required for simple polygon encoding.
- It is not trivial to map this storage form to/from the topological encoding form.
- Some analytic operations – such as those that require volumes, areas, distances, etc. of objects are inconvenient in the regular polytope representation.

15.3.1.2 Model Advantages

- Data retrieval can readily be optimised since each regular polytope can be stored as an individual self-contained record on disc, and indexed using standard techniques such as r-tree (Guttman 1984).
- All RCC and topological predicates and functions can be rigidly supported in the computer-based representation.

15.3.1.3 The Disjoint Normal Form

A minor variant on the above strategy is to make the decomposition of the regular polytopes into convex polytopes more restricted. In the disjoint normal form (DNF), the convex polytopes that comprise a regular polytope are not permitted to overlap. In deciding whether or not to use DNF considerations include:

- Calculation of the volume of a regular polytope in DNF is simpler. The volume of each convex polytope can be calculated, and the results summed.
- Conversion of the regular polytope to a vertex defined polyhedron is simplified, since the individual convex polytopes can be converted, and the resultant polyhedra can be 'dissolved' together. Polyhedron dissolve is a simpler and faster operation than calculation of a union.
- It is not trivial to convert an arbitrary regular polytope to DNF.
- Some conversion algorithms from vertex representations to regular polytope produce DNF naturally.
- The number of convex polytopes in a conventional regular polytope (allowing overlap) can be less than in the case of DNF (see figure 6).
- The decomposition into disjoint convex polytopes is not unique (additional decision rules would be required to make it unique).

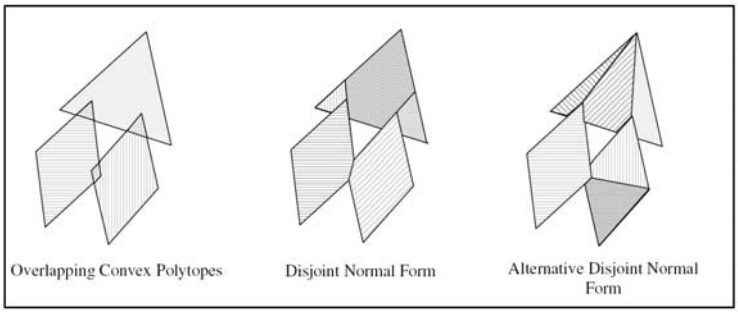

Fig. 15.6 A regular polytope in overlapping, and in disjoint normal form

15.3.2 Topological Encoding

Topological Encoding is a traditional form of storage of spatial data in vertex representation, which provides two major advantages over discrete polygon storage of coverages:

- It gives the option of fast neighbour searches (find adjacent polygons).
- It reduces the redundancy of storage of boundary details.

There are several variants on space partitioning using topological encoding, but all are based on the common storage of boundary details, with links between the storage location of the boundary, and the details of the region(s) delimited by that boundary. It is in the definition of a coverage[5] that this approach is most significant, where every boundary is used in the definition of at least two regions (apart from those few boundaries surrounding the entire coverage).

In figure 7, the line string between node 1 and node 2 defines region A to its left and region B to its right. It is in cases such as this, where there is some complexity in the definition of the common boundary, that the greatest advantages of the approach are realised. (Since the definition of the line string from 1 to 2 contains many points, which do not need to be stored twice as would be the case if A and B were defined as discrete polygons).

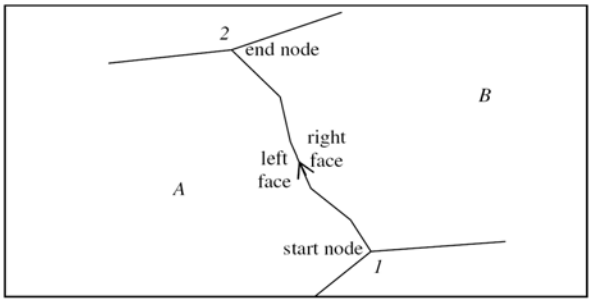

Fig. 15.7 Two regions delimited by a common boundary line

15.3.3 Topological Encoding of Regular Polytopes

In the storage schemes that are appropriate to the Regular polytope representation, there are several possible analogues to topological encoding, but one in particular is quite promising for use in the field of cadastral data. This approach treats each half space as a common object, stored once only (in the way a common boundary is as described above), with links to each convex polytope that it participates in the definition of (either as a direct definition or as a definition by the complement of the half space); see figure 8.

[5] In this context, 'coverage' is used to mean that the entire area of interest is partitioned into non-overlapping regions (with no gaps between regions).

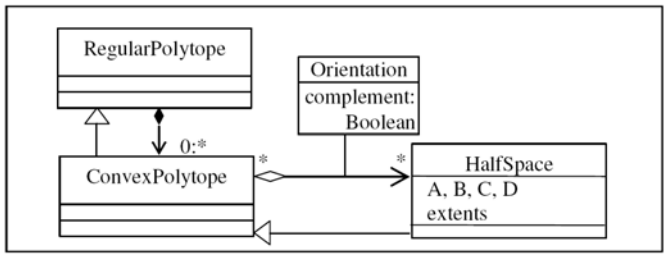

Fig. 15.8 Regular polytope schema with common storage of half spaces

As an example from the cadastral domain, consider a series of property parcels with a common road frontage as depicted in figure 9. The single half space XY participates in the definition of the road, and its complement in the convex polytopes A, B1, C1, D and E1 (see figure 10. This ensures that:

- There are no gaps or overlaps between parcels and the road.
- The road frontages are straight.

Since the true definition of the parcels from the survey plan was probably in terms of a bearing and distance measurement from point X to point Y, this is a particularly appropriate representation, and allows the option of storing such measurement metadata within the half space record. Thus half space XY would be linked by the direct connection to the road section 1, and via the 'complement = true' link to convex polytopes A, B1, C1, D and E1. Note that half spaces can be used more than twice in contrast to the traditional encoding of topology based on edges, where a common edge is used twice (positive and negative).

Even where straight sections of frontage are non-contiguous, the half space record can be used in common. For example, the half space marked Y-Z defines the road section 2, with its complement defining E1, F, road section 3, G etc.

In full 3D parcels, the same is possible, with a half space being able to define a number of parcels in strata, as well as defining a non-stratum (2D) parcel adjoining it. In figure 11, the half space marked as XY, is the boundary of '2D parcel' A, and its complement is the boundary of strata parcels B1 to B5.

The HalfSpace record also should carry attributes defining its extents of use. This would probably be in the form of a minimum enclosing rectangle, and is used for two purposes:

- To distinguish between half spaces which are only co-incident by chance (in which case individual representation of the HalfSpaces is more practical). For example, it is possible that two boundaries many kilometres apart have the identical A,B,C,D values, but are not in any way related, and should not be linked.

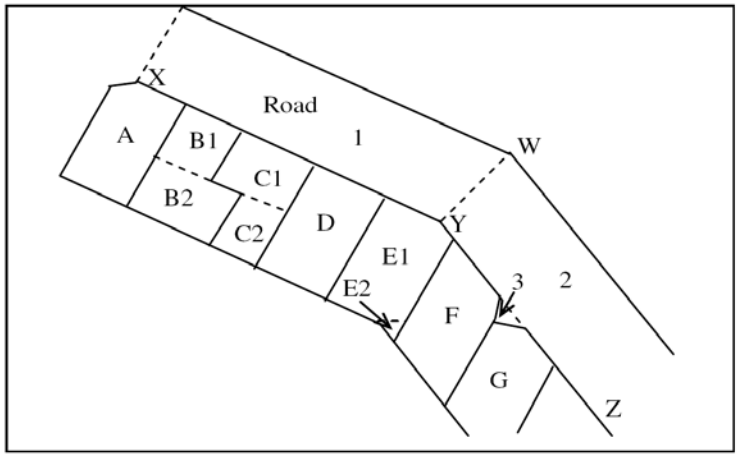

Fig. 15.9 Cadastral data in the topologically encoded regular polytope form

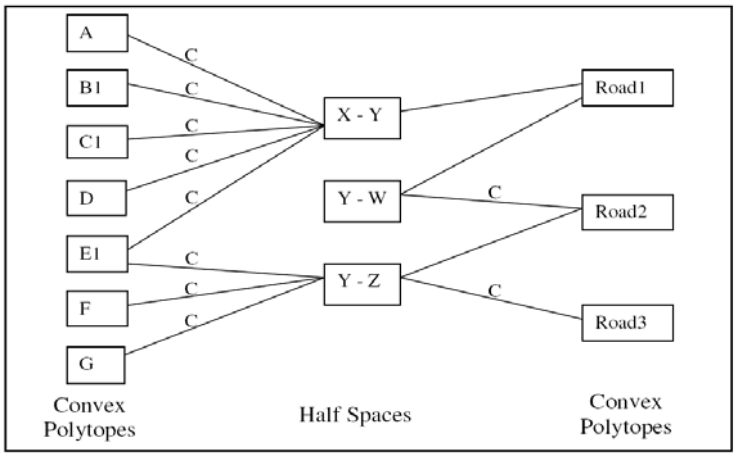

Fig. 15.10 Object diagram showing some of the connections in figure 9 (The linkages marked 'C' are links with complement = true as in figure 8)

- To allow easy application of adjustments such as datum changes. Where an adjustment can be approximated by a 'block shift', the new definition of the halvers in a block can be calculated using the localisation provided by the extents.

The advantages that are created by using conventional topological encoding apply to the topologically encoded regular polytopes well as the rigorous logic of the regular polytope, so that:

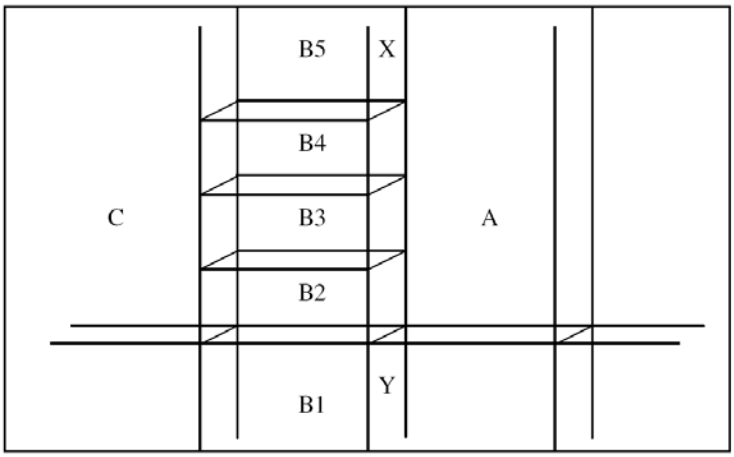

Fig. 15.11 3D parcels encoded using topologically encoded regular polytopes

- Some redundant storage is eliminated.
- Fast neighbour searches are facilitated.
- Accidental creation of overlaps and gaps is prevented.
- Frontages are kept straight.

It is unlikely that there will be much saving in storage requirements using this structure, since the cost of storing a halver redundantly is quite low, and largely offset by the keys and indexing needed to support the encoding. This is also true of the conventional form of topological encoding, and in both approaches the advantages are to be found in the ease of update, while retaining correct adjacency and consistency within the model.

15.4 The Java Demonstration Classes

A set of Java classes have been developed and tested on approximately a thousand cadastral parcels from the Queensland Cadastre, over a semi-urban region of average density and complexity. The region chosen contains primarily base (2D) parcels, but also has a smaller number of easements (secondary interests), and several 3D parcels. It consists of properties associated with residential, light commercial, light industrial, and recreational land usage.

The Java objects as developed parallel the definitions of the components of the regular polytope, and are divided into categories:

- The half space (or half plane),
- The convex polytope,

- The regular polytope.

This object model is intended for manipulation purposes in the processing software, and so differs from the various models given in section 3, which were intended primarily to illustrate a data storage strategy. The Java classes are set up to facilitate the mixing of 2 and 3 dimensional data.

This implementation models 3D and 2D objects, with no extensions to either lower of higher dimensionality. Since this is intended to explore practical issues associated with Cadastral data, no attempt has been made to produce a fully general n-dimensional model. Also there is no provision made for lower-dimensional objects such as lines, points and surfaces to be embedded in the space.

15.4.1 Description of the Java Objects

These classes and interfaces are defined for the calculation of the functions that have been defined on the regular polytopes. They contain redundant information and constructs that assist with these calculations, but are not necessarily stored permanently. Likewise, links that are described below may not be of a permanent nature. In contrast to the earlier class diagrams, which were intended to illustrate possible database table structures, the following diagrams document a series of Java classes that implement the RCC functions.

15.4.2 Classes and Relations

The half space/plane object is characterised by classes based on the Halver and the Face as in figure 12 described below:

- **HalfSpace**: defines a 3D half space and carries the parameters A,B,C and D
- **HalfPlane**: defines a 2D half plane and carries the parameters A,B and D. Parameter C is not needed in 2D.
- *Halver*: a virtual class abstracting the HalfSpace and HalfPlane classes (this is restricted at present to 2D or 3D, since this is what the problem domain requires – see sections 1.2 and 1.3).
- **Point2R** and **Point3R** are domain-restricted rational point classes. They consist of a tuple of rational numbers (2 or 3 respectively), each consisting of a pair of Java BigIntegers (see 4.4.9).

The association in figure 12 is:

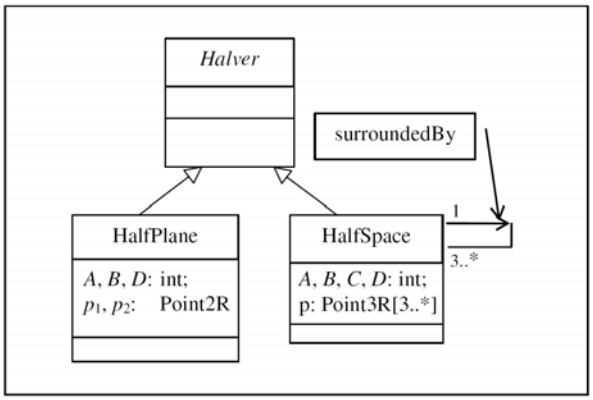

Fig. 15.12 The Half Space, Half Plane and Halver Classes

- **Surrounded By**: a redundant one-way linkage, from a halfSpace, to the halfSpaces that adjoin and define it. It is needed during the calculation of vertices, and each time a new *halver* is added to a convex polytope. In modifying a halfSpace to take account of a new halfSpace, it may be necessary to calculate two new dr-rational points. These are the points of intersection of this halfSpace, the new halfSpace, and existing halfSpaces that surround the face. The relation is not needed in the 2D case, since only two half planes are needed to define a point.

The ***ConvexPoly*** (figure 13) contains a collection of *Halver* objects. In the prototype, a convex poly must contain all 2D or all 3D halvers (and is sub-classed as ConvexPoly2 or ConvexPoly3 respectively). The MBR in the ConvexPolytope and in the RegularPolytope is a 3D box defined by integer coordinates which is guaranteed to contain the whole convex (or regular) polytope.

The ***Polytope*** contains a collection of *ConvexPoly* objects. In this implementation, all ConvexPoly objects in a particular Polytope object must be 2D or all must be 3D. The nrUnitsA, and unitNrA attributes are used for the calculation of C_a (weak) connectivity, nrUnitsB, and unitNrB attributes for C_b (strong) connectivity. A unit in this context is a connected set of convex polytopes, so that that C_a connectivity is defined as nrUnitsA = 1. Since C_b is a stronger form of connectivity, $C_b \Rightarrow C_a$, and therefore nrUnitsA $\leq$ nrUnitsB.

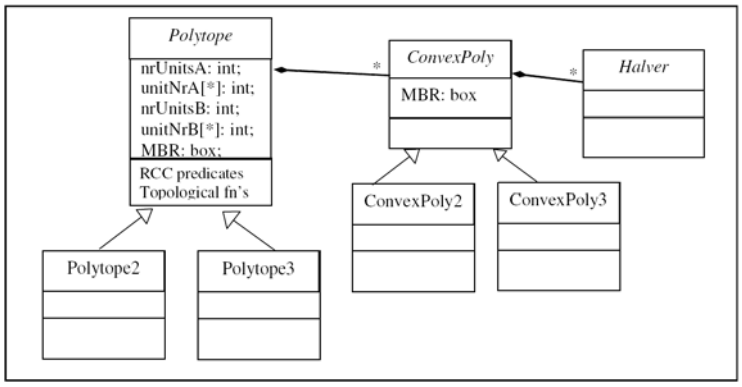

Fig. 15.13 The Regular Polytope and Convex Polytope Classes

15.4.3 Methods

Only the more important methods are described in this section. The main classes based on Polytope, ConvexPoly, and Halver, have methods which convert them to and from database form. In this demonstration suite, only the bare minimum is stored in the database – the A,B,C and D values of the halvers, the structure of the regular polytope and a bounding box. For the purpose of the demonstration, and to assist with development, encoding was via a simple text string, but in a final system, a more sophisticated binary storage mechanism would be used. Vertices could also be stored for speed of processing.

15.4.3.1 Polytope Methods

A regular polytope is constructed by creating an empty regular polytope O_{Φ}, with no convex polytopes, and extending it using Polytope.addConvexPoly(C). The methods provided in the regular polytope classes provide the full implementation of the RCC theory (Randell *et al.* 1992), extended for strong and weak connectivity:

- C_a (p,p_1) Polytope.connectsToA(Polytope)
- C_b (p,p_1) Poyltope.connectsToB(Polytope)
- DC_a (p,p_1) ¬ Polytope.connectsToA(Polytope)
- DC_b (p,p_1) ¬ Polytope.connectsToB(Polytope)
- P(p,p_1) Polytope.isWithin(Polytope)
- PP(p,p_1) Polytope.properPartOf(Polytope)
- EQ(p,p_1) Polytope.equals(Polytope)
- OV(p,$_1$) Polytope.intersects(Polytope)

- EC_a (p,p_1) Polytope.externallyConnectedToA(Polytope)
- EC_b (p,p_1) Polytope.externallyConnectedToB(Polytope)
- TPP_a (p,p_1) Polytope.tangentialProperPartOfA(Polytope)
- TPP_b (p,p_1) Polytope.tangentialProperPartOfB(Polytope)
- $NTPP_a$ (p,p_1) Polytope.nonTangentialProperPartOfA(Polytope)
- $NTPP_b$ (p,p_1) Polytope.nonTangentialProperPartOfB(Polytope)
- PO(p,p_1) Polytope.properOverlap(Polytope)

Note that by RCC theory, all of these relations can be generated from the 'connects' relation. In practice, some are directly calculated (such as 'intersects' – for reasons as given in Chapter 5), but most are simply implemented as their definition suggests. e.g.:

```
/** Determines if this regular polytope is within the other
  * @param other The other Regular Polytope
  * @return True if this regular polytope is within the
    other */

public boolean isWithin(Polytope other) {
  Polytope otherM = other.inverse();
  otherM = otherM.intersection(this);
  return (otherM.convexPolys.size() == 0); }

/** Determines if this regular polytope is equal to the other
  * @param other The other Regular Polytope
  * @return True if every point in this regular polytope is
  * within the other and visa versa. */

public boolean equals(Polytope other) {
   return (this.isWithin(other) && other.isWithin(this)); }
```

15.4.4 Results

Approximately one thousand parcels were selected from the Queensland Cadastre – see figure 14. The area chosen was the region surrounding the 'Gabba' cricket grounds in Woolloongabba Brisbane, because this area contains some 3D parcels of non-trivial shape. The parcels obtained from the database are 2D only, but do include secondary interests (such as easements). Thus overlapping 2D register objects exist. There were several 3D parcels in the region. Two associated with the cricket stadium (figure 15), and one with a restaurant (figure 16) were hand encoded.

In the original data, some inaccuracies had been introduced through rounding, so there are slight overlaps and mismatches between the edges. This will be discussed further in section 5.

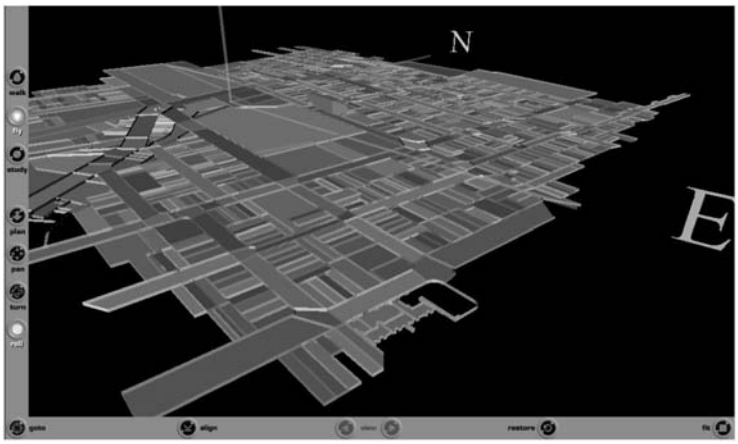

Fig. 15.14 An overview of the test region

Figure 14 shows the data in question. The 2D parcels have been represented by colouring a plane (at $z = 0$) with a randomly selected colour. To further show the division between parcels, a vertical 'fence' has been drawn, of the same shade as the horizontal surface. Since the colour on each side is different, some interfering visualization effects can occur.

Figure 15 shows a detail of some of the 3D parcels (which abut without overlapping); see also (Stoter 2004) pages 269-272 for a view of these same parcels.

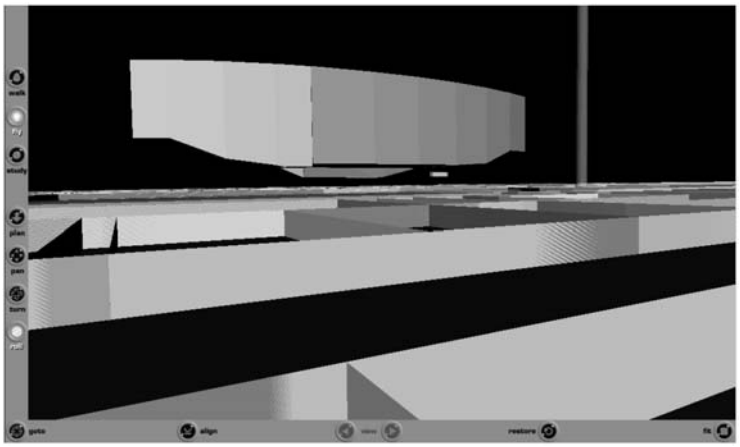

Fig. 15.15 Detail of two 3D contiguous parcels with a third in the far background (the vertical grey cylinder is the Z axis)

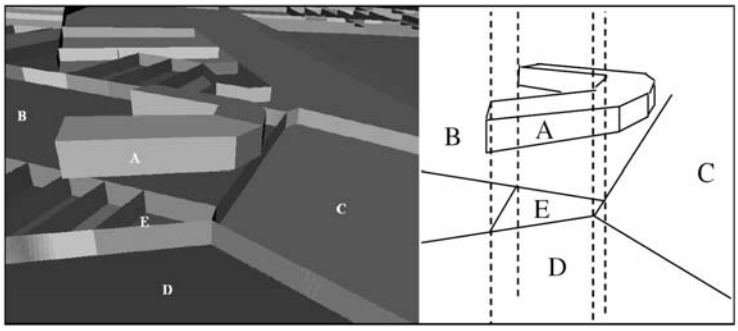

Fig. 15.16 3D parcel amongst 2D parcels. (Parcel A and 2D parcel E together comprise a restaurant. Parcel A overhangs the roadway represented by parcels B, C and D). Figure 16 depicts a 3D parcel A overhanging the footpath, and exactly abutting the 2D parcel E directly below the open space partially enclosed by it

15.4.4.1 Data Quantities

One of the reasons for conducting this investigation was to determine the storage requirements of this approach. Had a more rural region been chosen, the averages below may have been less attractive (because there are more vertices per shared boundary on average, resulting in more halvers and convex polytopes), and this could be the subject of further investigation. The parcels in the test region required the following representation (Table 1). For comparison, an indication of the conventional polygon/polyhedron complexity is shown, but note that in the 3D case, the number of corners is estimated only, as a polyhedral model was not constructed as part of this investigation:

	2D Case 1044 parcels	3D Case 3 parcels
Average Convex Polytopes per Regular Polytope	1.3	3.3
Average Halvers per Regular Polytope	5.3	23.6
Average Halvers per Convex Polytope	4.0	7.1
Worst Case Convex Polytopes per Regular Polytope	44	5
Worst Case Halvers per Regular Polytope	81	17
Worst Case Halvers per Convex Polytope	11	8
Average Corners per Conventional Parcel	6.3	36
Maximum Corners per Conventional Parcel	100	42

Table 15.1 Average complexity of semi-urban parcels

15.4.4.2 Algorithmic Complexity

Java is a difficult language to obtain clear timing tests, since it is interpretive and uses various strategies of partial compilation. It also uses a 'garbage collector' form of memory management, leading to unpredictable timings of operations. For this reason, no strict timing tests were done. On the other hand, the actual algorithms are available for complexity analysis, and this leads to the suggestion that a practical implementation is possible. In the following, only the critical and potentially complex routines of the demonstration implementation are discussed.

15.4.4.3 ConvexPoly.compareWith(ConvexPoly)

This determines the relationship between two convex polytopes, returning the possible results: DISJOINT, CONTACTSa, CONTACTSb, INTERSECTS, CONTAINS, CONTAINED or EQUAL, and is probably the most critical method, since it is used in nearly all other operations. Inspection of the code shows that this operation will execute in $O(f_1 f_2 p^2)$ time, where f_1, f_2 are the number of half spaces or planes in the convex polytope, and p is the average number of vertices in a face. In 2D, $p = 2$, so this becomes $O(f_1 f_2)$.

In 3D, it could be expected that the number of vertices on a face would be fairly constant in the range of about 3 to 6, so this also becomes $O(f_1 f_2)$. Thus it is important that in a practical system, the complexity of a convex polytope be kept limited. Fortunately, this is possible simply by dividing any highly complex convex polytopes into multiple smaller ones.

Thus, if the convex polytope is restricted to a specified maximum complexity, this routine is $O(1)$ (i.e. constant) in complexity. The cost of this simplification is an increase in the complexity of the regular polytope, so that more convex polytopes will be needed.

15.4.4.4 Constructing a Regular Polytope

As a regular polytope is constructed, each convex polytope must be compared with the convex polytopes previously added (to determine connectivity). This operation is thus of $O(n^2)$ where n is the number of convex polytopes in the regular polytope[6]. Since each convex polytope is a well defined geometric object, convex, and contains a MBR, it is relatively easy to optimise this operation. For example an in-memory spatial index could be used to reduce the search-time from $O(n^2)$ to $O(n \log n)$.

[6] This is assuming that the convex polytope complexity has been controlled as described above. Otherwise it would be $O(n^2 f^2)$ in 3D.

15.4.4.5 Polytope.intersection(Polytope)

This operation involves the calculation of the intersection of the Cartesian product of the convex polytopes. Thus it is by nature a *O(nm)* operation, however the construction of the resultant polytope from this Cartesian product raises this in theory to $O(nm \log nm$.

15.4.4.6 Polytope.inverse()

For regular polytope $O = \bigcup_{i=1..n} C_i$, with $C_i = \bigcap_{j=1..m_i} H_j$ the first step is to calculate: $O_i = \overline{C_i} = \bigcup_{j=1..m_i} \overline{H}_j$ for $i = i \ldots n$.

Thus, since each $m_i \leq c$ (by the assumption of the limited complexity of half spaces), this results in n regular polytopes, each of up to c convex polytopes. Each convex polytope consists of one half space only. Thus, this first part of the operation is $O(nc) = O(n)$ (because c is constant). Note that the inverse of a convex polytope consists of a regular polytope with up to c convex polytopes, each defined by one half space.

The second phase consists of forming the intersection of the n regular polytopes $\overline{O} = \bigcap_{i=1..n} O_i$. If approached without any optimisation, this would be disastrous – leading to an operation of order c^n.

Fortunately, at each step in the algorithm, a large number of convex polytopes that are generated by the intersection operation are discarded. At the end of the process, assume there are l convex polytopes left. If it is assumed that the number of convex polytopes in R remains fairly constant during the process, this means that the cost of processing each O_i in the intersection operation will be O(l log l). Since there are n polytopes, this gives O(nl log l). Note, this is an algorithm which could well repay some optimisation effort beyond the simple version used in the demonstration software.

15.4.4.7 Other Regular Polytope Operations

All of the other regular polytope operations (as shown in section 4.3.1) are simple combinations of other operations. So that the worst cases will be of no higher complexity that Polytope.inverse or Polytope.intersection.

15.4.4.8 Indexing and Searches

The programs as developed as a proof of concept do not use any database spatial indexing, and so are not efficient for doing spatial searches. On the other hand, they do generate a minimum bounding rectangle (or solid) sur-

rounding the vertices of each regular polytope and each convex polytope, and so a standard R-Tree algorithm can be used.

15.4.4.9 BigInteger Arithmetic

One of the advantages of implementing these routines in Java was the easy availability of the BigInteger object class. This provides a complete set of arithmetical operations on an integer representation with (effectively) no limit on the size of operands. The disadvantage of BigInteger is the slow speed of the operations, and the fact that the speed of operations is dependant on the size of the numbers involved.

In order to implement this software in a language other than Java (e.g. C), some of this functionality will need to be implemented. This is not a difficulty, since the algorithms are well known and documented. Further, not all functionality is needed. It is not necessary to allow for potentially infinite operands so memory allocation is not an issue. Although large numbers are involved, they are constrained. Further, not all arithmetic operations need be implemented – negation, addition and multiplication are needed, but division is not (this is a considerable simplification).

It is important to note that the use of BigInteger arithmetic is not an attempt to increase the accuracy of the data. The resolution of numeric forms such as 8 byte floating point is easily sufficient to cover the level of accuracy of virtually all spatial data in practical databases. The use of extended arithmetical types is to ensure repeatability and consistency in operations.

15.4.5 Optimizing the Model

Optimising techniques would benefit from control of the complexity of the individual convex polytopes. The calculation of the vertices of a convex polytope is a significant process, strongly dependant on the cardinality of the set of half-spaces in a Convex Polytope. Restricting this cardinality can control this complexity, even at the cost of increasing the cardinality of the set of Convex Polytopes in a Regular Polytope. It is significantly easier to optimize the operations between convex polytopes.

In calculating the intersection of region B with region A in figure 17 (shown as two convex polytopes A_1 and A_2), even though all the half planes which define A_1 intersect all the half planes that define B (since the half-plane definition is theoretically infinite), it can be determined by a conventional bounding box overlap test that all the vertices of A_1 are completely separated from the vertices of B – therefore $A_1 \bigcap B$ is empty. This logic can be used to pre-eliminate large numbers of the partial intersections, and could be augmented

by an in-memory spatial index – e.g. an R-tree based on the bounding boxes (shown as dashed lines) to further improve the calculation speed.

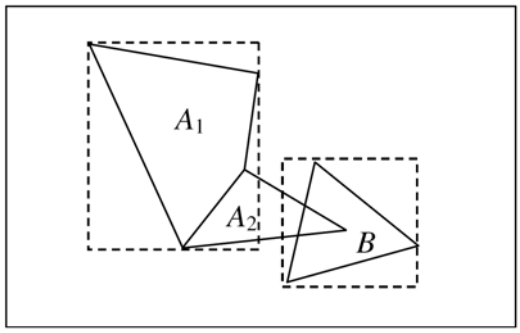

Fig. 15.17 Calculating the intersection of two regular polytopes

While not necessary to the theory, it may improve the efficiency of many operations if the disjoint normal form (DNF) is used in representing regular polytopes (see 3.1.3). The indexing and comparison of convex polytopes can be improved thereby, since the disjoint convex polytopes will have smaller minimum bounding rectangles.

15.5 Data Load Issues

A major issue in the practical implementation of a regular polytope based storage mechanism is that of data conversion. Once the geometry is expressed in regular polytope form the operations between geometric regions are guaranteed to be rigorously correct, but the quality of the source data must be considered. Approximations may well have been made, and inaccuracies introduced to allow the data to be stored in the previous form, and this may create difficulties in data uptake.

15.5.1 Inaccuracy from Previous Systems

In many systems, calculation of the point of intersection between lines will have introduced rounding errors, as in the road frontage in figure 18 which was intended to be a straight line connection *A* and *B*. In addition, to avoid later topological failures, a further displacement of the calculated point may have been applied, as in the case pictured in figure 19.

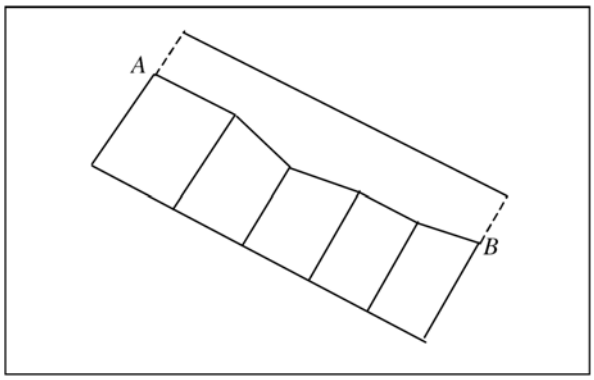

Fig. 15.18 Points moved slightly in the calculation of intersections (shown exaggerated)

Note – this assumes that the data is to be loaded from an existing spatial database. In some cases, it may be possible to capture from the original source – e.g. the survey data. Unfortunately, while it would have been ideal to have captured original data in its uncompromised form, this is rarely the case, and much processing has been done to data before it reached the database. For example, bearings and distances will have been adjusted to 'close' and elevations of 3D points will have been calculated from the raw field notes. Note that there is a trend in which the original observations and measurements are more often stored in the (cadastral) database, in addition to the resulting interpretations (parcels).

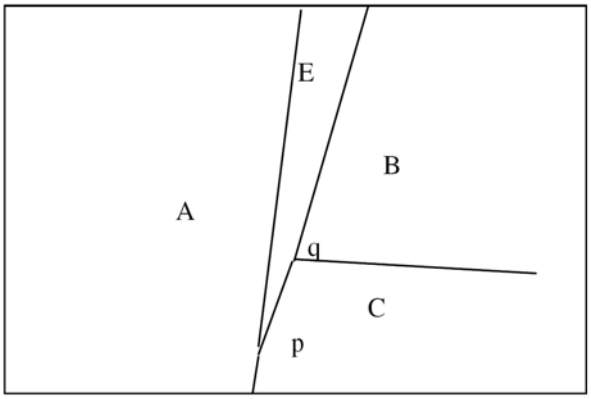

Fig. 15.19 Polygon point q moved (to the right) to prevent topological failure

Ideally, before such parcels are converted into regular polytope form, the lines which were once straight, and were intended to be straight should be identified, and be represented as a single half plane (or at least half planes having the same A, B, D values). As was mentioned above (section 4.4), this was not done in the demonstration software, resulting in small overlaps and mismatched edges. These can, however be detected by the rigorous operators available in the regular polytope representation.

15.5.2 2D Data Conversion to Regular Polytope Form

In the 2D version of the regular polytope, there is no difficulty generating a half plane whose edge passes exactly through any two points with integral coefficients. In the same way, any 2D data that is currently encoded using integer coefficients will create a 3D regular polytope with vertical walls with no loss of precision. In summary, it is possible to generate a half plane in 2D, or a half space parallel to the z axis through any two points with integral coefficients.

For example in figure 20, the incoming data is based on lines $(x_1, y_1)(x_2, y_2)$ and $(x_2, y_2)(x_3, y_3)$. The planes can be defined as $A_1 = y_1 - y_2$, $B_1 = x_2 - x_1$, $C_1 = 0$, and $D_1 = x_1 y_2 - x_2 y_1$, and $A_2 = y_3 - y_2$, $B_2 = x_2 - x_3$, $C_2 = 0$, and $D_2 = x_3 y_2 - x_2 y_3$. Clearly any point on the intersection of these planes will have $x = x_2$ and $y = y_2$.

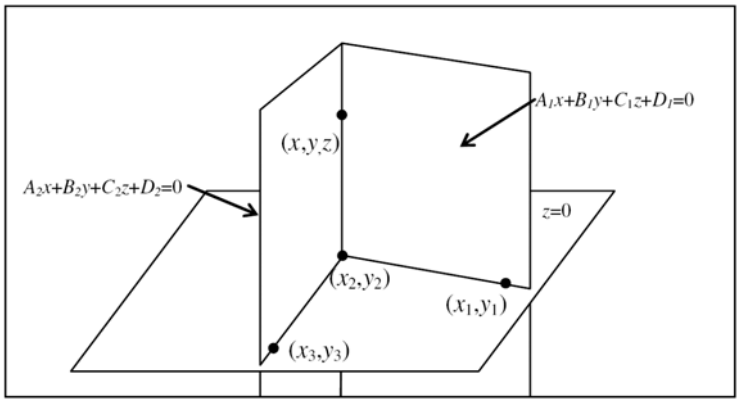

Fig. 15.20 3D planes based on incoming 2D data

Thus, any 2D data currently in a conventional format should easily be converted into regular polytope form with no loss of resolution, and no movement of vertices provided that integer representations are used for each. That is to

say, if no attempt is made to straighten road frontages as part of the data load (as described above in 5.1), existing 2D cadastral data can be loaded unchanged. This is not necessarily the case with 3D data.

15.5.3 3D Data Conversion to Regular Polytope Form

Where 3D data is to be converted to regular polytope form, some care is needed. In general given any three non-colinear points, the best that can be asserted is that a half space can be generated whose boundary plane passes within one unit of resolution of each of the points. (In many special cases – specifically where the half space is parallel to any of the axes, much better results can be expected). If a situation such as that of figure 19 occurs in a 3D situation (the figure should be interpreted as a 'slice' through the 3D coverage), and the spurious bend at point q is straightened, the actual position of point p (as a point of intersection) is subject to a large variation.

In figure 21, where the half spaces that define region E have a possible imprecision of one unit, their point of intersection p has a much larger possible error (shown shaded). If this is a critical issue, it may be solved by introducing a deliberate bias to the approximation, and an additional half space to limit the position of p. The bias is needed because the additional half plane can limit the error to the south (in figure 22) but not to the north. The bias is created by ensuring that all half planes that meet at acute angles are moved away from the convex polytope they define.

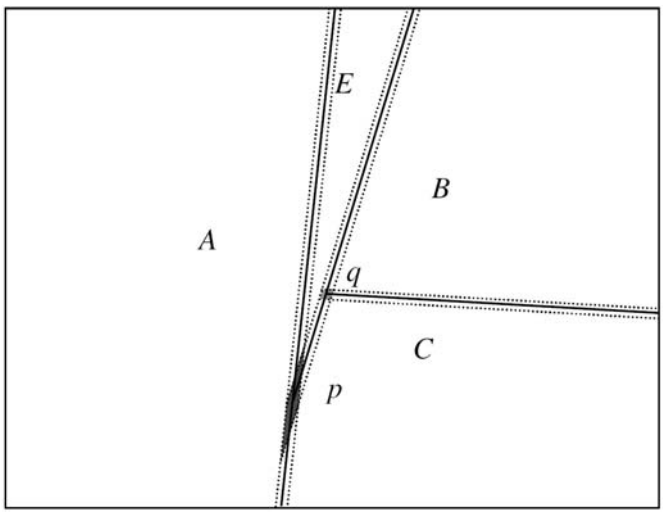

Fig. 15.21 Imprecision in the placement of the point of intersection

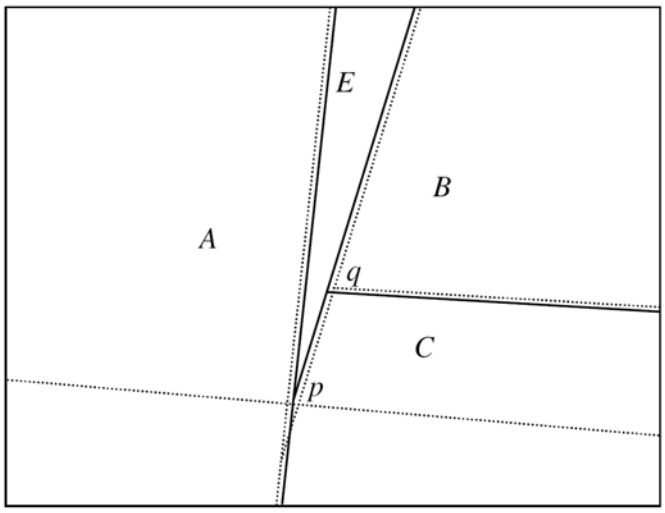

Fig. 15.22 Half space introduced to constrain the point position

For example, in figure 22, region E has been extended (still within one unit of resolution), and the acute angle at p has been truncated. This procedure – of truncating acute dihedral angles could be used as a general procedure in converting geometric objects to regular polytope form. In any case, objects with very acute angles need to be treated with caution in any representation to avoid the possibility of failures in algorithms such as buffer generation, generalisation, etc.

15.6 Conclusions

The Regular Polytope approach is practical, and could be rigorously implemented as a large-scale database (with proven functionality and without unpleasant surprises due to the mismatch of infinite real number mathematics and the finite digital computer). While some more optimization in the area of the regular polytope algorithms could yield speed improvements, for the sort of data used in this pilot system, acceptable results were obtained. In the test region, it was possible to run any combination the standard RCC and topological functions, and the combination and nesting of functions gave completely predictable results. The indication is that, a full implementation could be developed with query and analysis, and edit/update functionality.

It is expected that, as described above, restricting the complexity of convex polytopes will ensure practical speeds. In the case of 2D polytopes, several thousand half planes per complex polytope should be practicable. In 3D,

the number is probably several hundred. This is appropriate particularly for cadastral data, whereas the parcels with large numbers of points (more than 2000) required in their definitions generally occur in rural areas, and are all 2D. Overly complex convex polytopes can be subdivided into a number of simpler convex polytopes.

The overwhelming advantage of the Regular Polytope approach is in the rigorously correct logic that it supports, and this justifies some additional data storage requirements, and the potentially slower processing times, but there is still much potential to improve the implementation of some of the operations – in particular, Polytope.intersection, and Polytope.inverse.

Possible future extensions of the Regular Polotype approach may include: non-linear half-spaces (e.g. circular arc, or polar coordinates defining a parcel boundary) and time as an additional dimension.

It has been shown that the definition of a half space is unique, and that the definition of a convex polytope in terms of half spaces is also unique (Thompson 2005c). If a decomposition of a regular polytope into a unique set of disjoint convex polytopes can be defined, it would be of great algorithmic value, among other things, providing a simple determination of equality.

References

Arens, C., J. Stoter and P. van Oosterom (2003). Modelling 3D Spatial Objects in a Geo-DBMS Using A 3D Primitive. Association Geographic Information Laboritories Europe, Lyon, France.

Borgo, S., N. Guarino and C. Masolo (1996). A Pointless Theory of Space Based On Strong Connection and Congruence. 6th International Conference on Principles of Knowledge Representation and Reasoning (KR96), Morgan Kaufmann.

Castellanos, D. (1988). 'The Ubiquitous pi (Part II)'. Mathematics Magazine 61(3): 148-161.

Clementini, E., P. Di Felice and P. van Oosterom (1993). A Small Set of Formal Topological Relationships Suitable for End-User Interaction. Third International Symposium on Advances in Spatial Databases, Singapore.

Düntsch, I. and M. Winter (2004). 'Algebraization and Representation of Mereotopological Structures.' Relational Methods in Computer Science 1: 161-180.

Egenhofer, M. J. and J. R. Herring (1994). Categorising binary topological relations between regions, lines, and points in geographic databases. The nine

intersection: formalism and its use for naturallanguage spatial predicates. M. J. Egenhofer, D. M. Mark and J. R. Herring, University of California.

Ellul, C. and M. Haklay (2005). Deriving a Generic Topological Data Structure for 3D Data. Topology and Spatial Databases Workshop, Glasgow, UK.

Franklin, W. R. (1984). 'Cartographic errors symptomatic of underlying algebra problems'. International Symposium on Spatial Data Handling, Zurich, Switzerland: 190-208.

Guttman, A. (1984). 'R-Trees: A Dynamic Index Structure for Spatial Searching' ACM SIGMOD 13: 47-57.

Hölbling, W., W. Kuhn and A. U. Frank (1998). 'Finite-Resolution Simplical Complexes.' Geoinformatica 2:3: 281-298.

Kazar, B. M., R. Kothuri, P. van Oosterom and S. Ravada (2007). On Valid and Invalid Three-Dimensional Geometries. In this book '2nd International Workshop on 3D Geo-Information: Requirements, Acquisition, Modelling, Analysis, Visualisation, 12-14 December 2007, Delft, the Netherlands'.

Naimpally, S. A. and B. D. Warrack (1970). Proximity Spaces. University Press, Cambridge.

OMG. (1997). 'UML 1.5'. Retrieved 2004 from http://www.omg.org/ technology/documents/formal/uml_2.htm

Randell, D. A., Z. Cui and A. G. Cohn (1992). A spatial logic based on regions and connection. 3rd International Conference on Principles of Knowledge Representation and Reasoning, Cambridge MA, USA, Morgan Kaufmann.

Smith, B. (1997). Boundaries: An Essay in Mereotopology. The Philosophy of Roderick Chisholm. L. Hahn, LaSalle: Open Court: 534- 561.

Stoter, J. (2004). 3D Cadastre. PhD Thesis. Delft, Delft University of Technology.

Stoter, J. and P. van Oosterom (2006). 3D Cadastre in an International Context. Taylor & Francis, Boca Raton FL.

Tarbit, S. and R. J. Thompson (2006). Future Trends for Modern DCDB's, a new Vision for an Existing Infrastructure. Combined 5th Trans Tasman Survey Conference and 2nd Queensland Spatial Industry Conference. Cairns, Queensland, Australia.

Thompson, R. J. (2004). 3D Topological Framework for Robust Digital Spatial Models. Directions Magazine.

Thompson, R. J. (2005a). 3D Framework for Robust Digital Spatial Models. Large-Scale 3D Data Integration. S. Zlatanova and D. Prosperi. Boca Raton, FL, Taylor & Francis.

Thompson, R. J. (2005b). 3D Cadastral Issues Within NR&M. Brisbane, Department of Natural Resources and Mines (Internal Report).

Thompson, R. J. (2005c). 'Proofs of Assertions in the Investigation of the Regular Polytope'. Retrieved 2 Feb 2007 from `http://www.gdmc.nl/publications/reports/GISt41.pdf`

Thompson, R. J. (2007). Towards a Rigorous Logic for Spatial Data Representation. Geo Database Management Centre. Delft, Delft University of Technology. PhD Thesis.

Thompson, R. J. and P. van Oosterom (2007). 'Connectivity in the Regular Polytope Representation.' submitted to GeoInformatica.

van Oosterom, P., W. Quak and T. Tijssen (2004). About Invalid, Valid and Clean Polygons. Developments In Spatial Data Handling. P. F. Fisher. New York, Springer-Verlag: 1-16.

Verbree, E., A. van der Most, W. Quak and P. van Oosterom (2005). Overlay of 3D features within a tetrahedral mesh: A complex algorithm made simple. Auto Carto 2005, Las Vegas.

Weisstein, E. W. (1999). 'Boolean Algebra'. MathWorld – A Wolfram Web Resource Retrieved 20 Jan 2007 from `http://mathworld.wolfram.com/BooleanAlgebra.html`

Weisstein, E. W. (2005). 'Rational Number'. MathWorld – A Wolfram Web Resource Retrieved 23 May 2005 from `http://mathworld.wolfram.com/RationalNumber.html`

Zlatanova, S. (2000). 3D GIS for Urban Development. Graz, Graz University of Technology.

Zlatanova, S., A. A. Rahman and W. Shi (2004). 'Topological models and frameworks for 3D spatial objects'. Journal of Computers & Geosciences 30(4): 419-428.