IMPLEMENTATION ISSUES IN THE STORAGE OF SPATIAL DATA AS REGULAR POLYTOPES

RODNEY JAMES THOMPSON (1, 2) and Peter van Oosterom (1)

(1) Delft University of Technology, OTB, GIS technology
The Netherlands
(2) Landcentre – Department of Natural Resources and Mines
Australia

ABSTRACT

The "Regular Polytope" has shown some promise in providing a rigorous representation of geometric objects in 2D and 3D (Thompson 2005), in a form that is computable using the finite arithmetic available on digital computers. This is in contrast to the current practise where geometric algorithms are based on infinite precision mathematical axioms, which do fail (in exceptional cases) in the finite digital computers.

It has also been shown theoretically to provide two alternative definitions of "connectivity" (Thompson, van Oosterom and Pullar 2006). A convex polytope is defined as the intersection of any finite number of half spaces. A polytope representation is then defined as the union of a finite set of convex polytopes.

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 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 the implementation, and discusses some of the practical considerations that arose as a result. This gives an indication of the requirements of a full implementation, and what further development is needed.

INTRODUCTION

This paper will describe the Regular Polytope representation, with a re-statement of the definition of the components that are used to define it. A brief definition of the concepts of connectivity, and of pseudo-rational numbers is included as background.

This is followed by a discussion of the rationale for the prototype development work, and a description of the form that the implementation took. This is followed by an indication of the results of testing the approach, and by a summary and suggestion of further work to be undertaken.

Definition of domain-restricted rational numbers and points

Given two (large) integers M' and M", a dr-rational number r can be defined as an ordered pair of computational integers r = (I,J), (–M"≤I≤M", 0<J≤M'), interpreted as having a value of I/J.[1] The dr-rational numbers do not form a field[2] (Weisstein 2005), in contrast to the true rational numbers, and therefore cannot span a vector space (by the definition of a vector space) (Weisstein 1999). There are a number of other counter-intuitive properties of dr-rational numbers, for example that the sum of dr-rational numbers may not be dr-rational.

A dr-rational point is an ordered tuple of dr-rational numbers 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 of a line between two dr-rational points is a dr-rational point.

In this paper, uppercase non italic letters such as A, B, X, Y will be used for computational integers, lowercase non-italic letters such as x, y, z will be used for dr-rational numbers.

Half Space Definition

In 3D a half space H(A,B,C,D) is defined as the set of all points P(x,y,z), -M≤x,y,z≤M where:

(A.x + B.y + C.z + D) > 0 or
[(A.x + B.y + C.z + D) = 0 and A> 0] or
[(B.y + C.z + D) = 0 and A=0 and B>0] or
[(C.z + D) = 0 and A=0, B=0 and C=0]
Where M is the range allowed for point representations.

Two special half spaces are defined,

$H_\varphi = H(0,0,0,-1)$ ('empty' i.e. points for which –1 > 0).
$H_\infty = H(0,0,0,1)$ ('universal' i.e. points for which 1 > 0).

The complement of a half space $H = (A,B,C,D)$ is defined as $\overline{H} = (-A,-B,-C,-D)$.

Convex Polytope Definition

A convex polytope C is defined as the intersection of a finite set of half spaces[3] $C = \mathop{\mathrm{I}}_{i=1..n} H_i$.

See *Figure 1* for a 2D example of a bounded and not completely bounded convex polytope.

[1] The reason for the name "dr-rational" is that the values of I and J are constrained to a restricted domain of possible values.

[2] The set of true rational numbers **Q** obey the field axioms, including the closure axioms (e.g. a ∈ **Q**, b ∈ **Q** ⇒ a.b ∈ **Q**). **T**his is not the case for dr-rational numbers

[3] In this paper, the term "halver" will be used generically to indicate half space or half plane depending on whether a 3D or 2D geometry is being considered. Most of the illustrations are in 2D for ease of drawing.

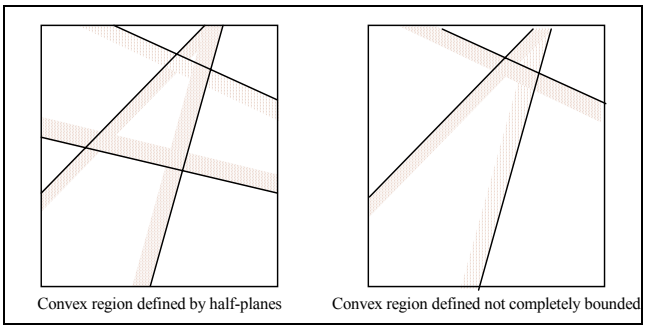

Figure 1: Convex Polytopes Defined by Half Planes.

Regular Polytope Definition

A regular polytope O is defined as the union of a finite set of convex polytopes $O = \bigcup_{i=1..m} C_i$;

see *Figure 2* for example (however a polytope needs not be connected, as this example is).
It is simple to show that the set of regular polytopes obeys the axioms of a topological space (Gaal 1964), based on the natural definitions of union and intersection (Thompson 2005b).

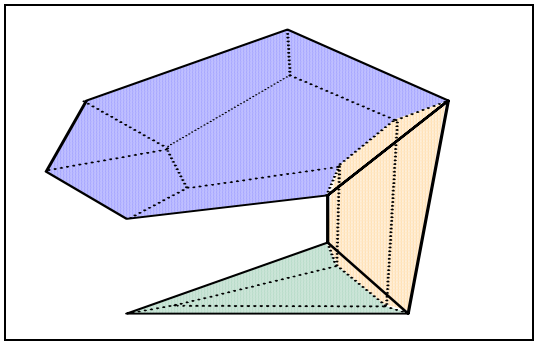

Figure 2: Definition of Regular Polytope from Convex Polytopes.
(The dashed lines are "hidden edges", included to give the impression of 3D).

RATIONALE FOR THE APPROACH TAKEN

It was felt that, while the Regular Polytope can clearly be used to represent various geometric constructs, it is only by applying the representation to a practical problem that its true worth can be verified. For that reason, it was decided to load 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 secondary interests, and several 3D parcels. It consists of properties associated with residential, light commercial, light industrial and recreational land usage.

The base cadastral parcels are known as 2D parcels, but as pointed out by Stoter (2004), the property right is actually to a volume of space, with the height and depth restrictions not explicitly stated. Thus it is the definition of the property that is 2D, not the property itself. The regular polytope, since it does not need to be bounded on all sides is a natural representation for a mix of 2D and 3D parcels. It is quite meaningful in this context, to ask whether a 2D base parcel intersects with a 3D volumetric parcel. For example in *Figure 3*, parcels A and B do not intersect C, but D and a section of road do.

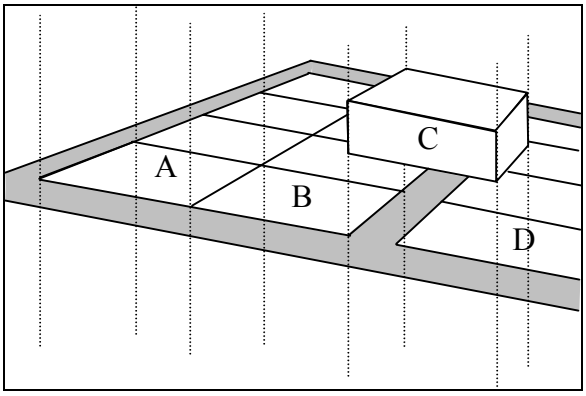

Figure 3: Mixing 2D and 3D Cadastre.

DESCRIPTION OF THE JAVA CLASSES

The Java classes as developed parallel the definitions of the components of the regular polytope.

The Halver interface is implemented by classes:

- HalfSpace – is defined by the parameters A, B, C and D.
- HalfPlane – is defined by the parameters A, B, and D.

This is augmented by the "Face" classes. While these are redundant to the half space/plane objects and are not stored in the database, they provide for faster manipulation of the constructs (in particular, for the calculation of connectivity). There is a further redundant linkage, between a PlaneFace, and the Half Spaces that adjoin and define it.

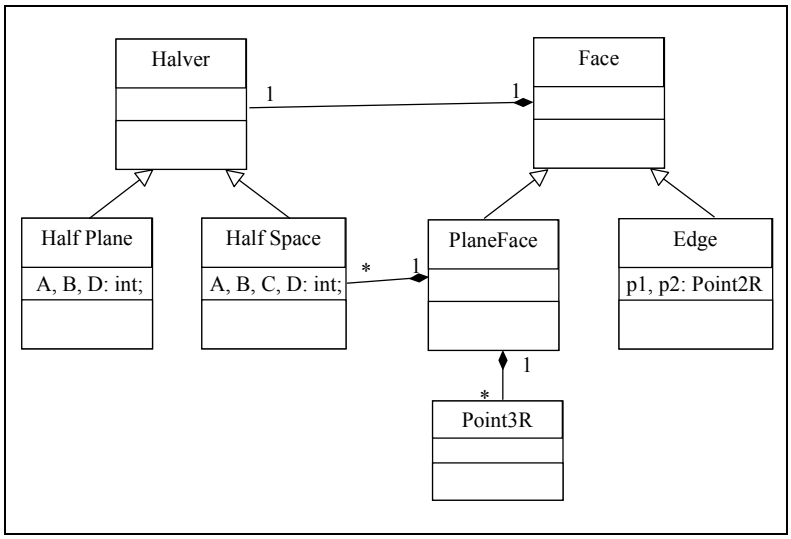

Figure 4: The Half Space, Half Plane and Face Classes.

Point2R and Point3R are domain-restricted rational point classes. They consist of a tuple of dr-rational numbers, each number consisting of a pair Java BigIntegers. BigInteger is a convenient method of dealing with the large precision required by this approach, but is not strictly necessary, since the precision requirements, though large, are not unlimited.

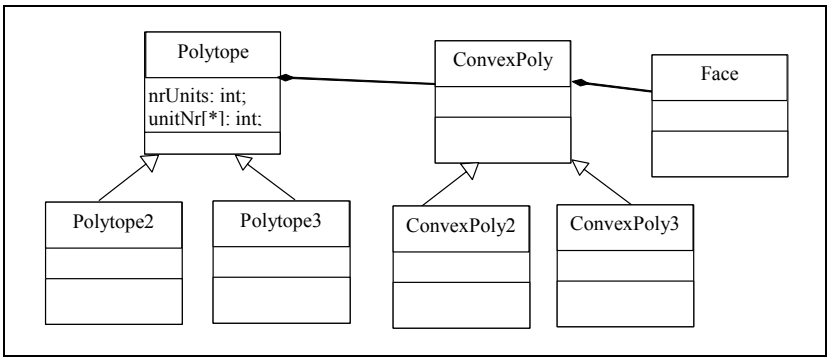

Figure 5: The Polytope and Convex Polytope Classes.

The Convex Polytope (*Figure 5*) is implemented by the ConvexPoly class, which contains a collection of Face objects. A convex poly must contain all 2D or all 3D faces (and is sub-classed as ConvexPoly2 or ConvexPoly3 respectively).

The Regular Polytope is characterised by the Polytope class. This contains a collection of ConvexPoly objects. All ConvexPoly objects in a particular Polytope object must be 2D or all must be 3D. The nrUnits, and unitNr attributes are used for the calculation of connectivity. A unit in this context is a connected set of convex polytopes.

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, and the structure of the regular polytope. For the purpose of the demonstration, and to assist with development, encoding was via a simple text string (see Appendix I), but in a final system, a more sophisticated storage mechanism would be used.
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, Cui and Cohn 1992) – see Table 1.

Operation	Description	Implementation
C(p, p_1)	p connects to p_1 (implementing C_a).	Polytope.connectsTo(Polytope)
DC(p,p_1)	p is disconnected from p_1.	Polytope.connectsTo(Polytope)
P(p,p_1)	$p \subseteq p_1$.	Polytope.isWithin(Polytope)
PP(p,p_1)	$p \subset p_1$.	Polytope.properPartOf(Polytope)
EQ(p,p_1)	$p = p_1$.	Polytope.equals(Polytope)
O(p,p_1)	$p \cap p_1 \neq \varphi$	Polytope.intersects(Polytope)
EC(p,p_1)	p is externally connected to p_1. $C(p, p_1) \wedge \neg O(p, p_1)$	Polytope.externallyConnectedTo(Polytope)
TPP(p,p_1)	p is a tangential proper part of p_1. $p \supset p_1 \wedge C(p, \overline{p}_1)$	Polytope.tangentialProperPartOf(Polytope)
NTPP(p,p_1)	p is a non-tangential proper part of p_1. $\neg C(p, \overline{p}_1)$	Polytope.nonTangentialProperPartOf(Polytope)
PO(p,p_1)	$p \cap p_1 \neq \varphi$, $p \not\subset p_1$, $p_1 \not\subset p$, $p \neq p$	Polytope.properOverlap(Polytope)

Table 1: The Basic Relations of RCC theory.

```
   /** 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));  }
   /** 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);  }
```

Figure 6: Implementation of "equals", and "within" Relations.

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"), but most are simply implemented as their definition suggests. e.g. see *Figure 6*.

Function	Description	Implementation
Complement	$\neg p$	Polytope.inverse()
Union	$p \cup p_1$	Polytope.union(Polytope)
Intersection	$p \cap p_1$	Polytope.intersection(Polytope)

Table 2: Topological Functions on Regular Polytopes.

RESULTS

Approximately one thousand parcels were selected from the Queensland Cadastre. The area chosen was the region surrounding the "Gabba" cricket grounds in Woolloongabba Brisbane.

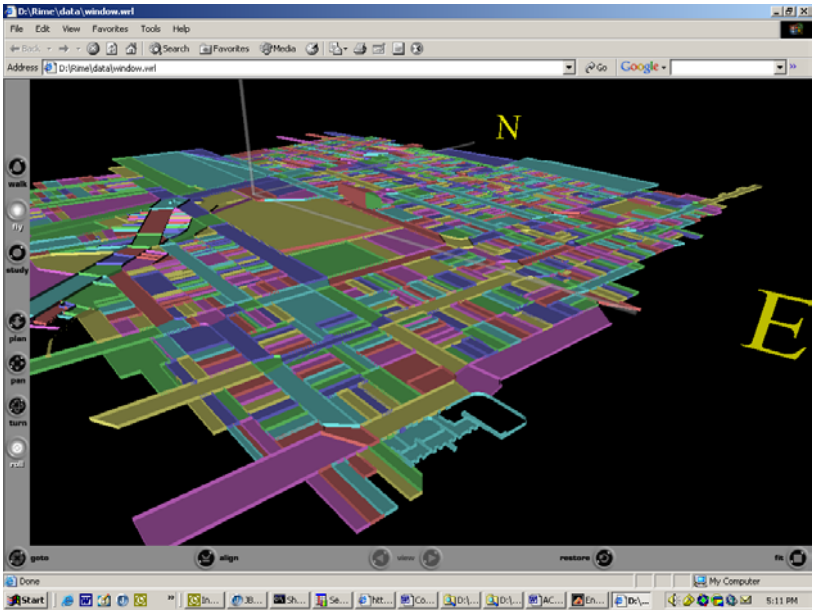

Figure 7: Overview of the Test Region.

Figure 7 shows the data in question. The 2D regular polytopes 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 colour. The parcels obtained from the database are 2D, but do include secondary interests. Thus overlapping 2D parcels exist. There were several 3D parcels in the region. Two associated with the cricket stadium, and one with a restaurant were hand encoded. *Figure 8* shows a detail of some of the 3D parcels (two of which abut without overlapping).

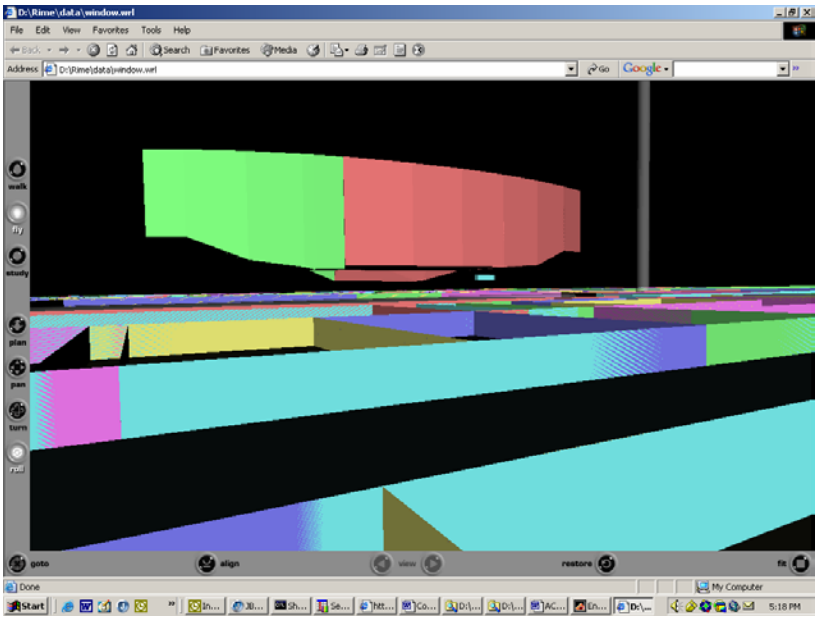

Figure 8: Detail of two 3D parcels (red and green) with a third (blue) in the background.

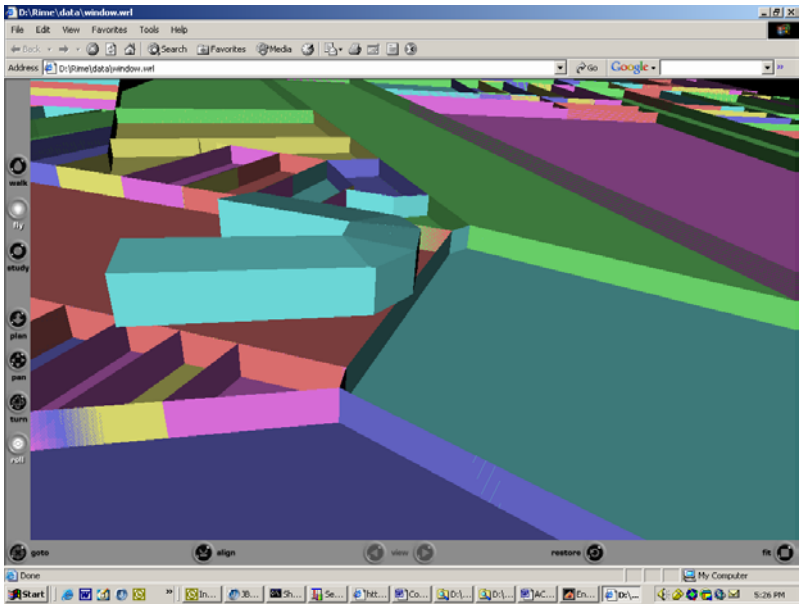

Figure 9: Another 3D object – A Restaurant Overhanging the Footpath.

In *Figure 9* it is not immediately obvious that the 3D parcel overhangs the footpath, and exactly abuts the 2D parcel (in mauve) directly below the open space partially enclosed by it.

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, and this will be the subject of further investigation. The parcels in the test region required the following representation:

	2D Case	3D Case
Average Convex Polytopes per Regular Polytope	1.3	2
Average Halvers per Regular Polytope	5.3	9.25
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
Avg bytes to encode a regular polytope in binary	120.4	457
Worst case to encode a regular polytope in binary	973	620

Table 3: Average and Worst Case Complexity of Semi-urban Parcels.

Algorithmic Complexity

Java is not a suitable 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, but the overall impression was slow but workable. 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 are discussed.

ConvexPoly.compareWith(ConvexPoly)

This is probably the most critical method, since it is used in nearly all other operations (multiple times). Inspection of the code shows that this operation will execute in $O(f^2p^2)$ time, where f is the number of half spaces or planes in the complex polytope, and p is the number of vertices in a face. In 2D, $p = 2$, so this becomes in $O(f^2)$. In 3D, it could be expected that the number of vertices on a face would be proportional to the number of faces, so this becomes $O(f^4)$. 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 convex polytopes.

If the convex polytope is restricted to a specified maximum number of halvers, this routine is $O(c)$ (i.e. constant) in complexity. The cost of this simplification is an increase in the complexity of the Regular Polytope.

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. Since each convex polytope is a well defined geometric object, and is convex, it should be relatively easy to optimise this operation, making use of limiting bounding regions, and/or scan line algorithms.

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 to $O(n^2m^2)$. In practice, however this is not as bad as it would seem, since most of the intersection convex polytopes will be empty. In practice, it is expected that the operation will be no worse than $O(l^2)$, where l is the number of convex polytopes in the final regular polytope.

Polytope.inverse()

This operation involves the calculation of the inverse of each convex polytope, and the calculation of the intersection of the resultant polytopes. This appears to be of high complexity, but again the vast majority of the resultant intersection convex polytopes will be empty. Again, it is expected in practice that the operation will be of $O(l^2)$.

Other Regular Polytope Operations

All of the other regular polytope operations (as shown in Tables 1 and 3) are simple combinations of other operations (e.g. see Table 2). Therefore the worst cases will be of no higher complexity than Polytope.inverse or Polytope.intersection.

BigInteger Arithmetic

One of the advantages of using Java was the availability of the BigInteger object class. This makes available a complete set of arithmetical operations on an integer representation with effectively unlimited size of operands. The disadvantage is the slow speed of these operations, and the fact the speed is dependant on the magnitude of the numbers.

In order to implement this software in a language other than Java (e.g. C), some of this functionality will need to be implemented, but fortunately not all. It is not necessary to allow for potentially infinite operands – although large numbers are involved, they are constrained. (In order to give millimetre precision in a region the size of Queensland, 96 bit integers are needed in 2D and 160 bit in 3D). Not all arithmetic operations need be implemented – negation, addition and multiplication are needed, but division is not (a considerable simplification).

LEARNING AND FUTURE RESEARCH

The approach is practical, and could be implemented as a large-scale database. While some more optimization of 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 compare and manipulate the data readily (detecting and correcting overlaps) using all the standard RCC and topological functions, and the combination and nesting of functions gave completely predictable results.

It is expected that, as described above, restricting the complexity of convex polytopes will lead to practical speeds. In the case of 2D polytopes, several thousand half planes per convex polytope would be practicable. In 3D, the number is probably several hundred. This is quite

appropriate for cadastral data, where the parcels with large numbers of points (more than say 2000) required in their definitions generally occur in rural areas, and are all 2D.

The overwhelming advantage of the Regular Polytope approach is in the rigorously correct logic that it supports, and this justifies the 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.

REFERENCES

Gaal, S. A. (1964) *Point Set Topology,* Academic Press, New York.

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.

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

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

Thompson, R. J. (2005b) Proofs of Assertions in the Investigation of the Regular Polytope, *GDMC, Delft University of Technology*, http://www.gdmc.nl/publications/reports/GISt41.pdf, 2006.

Thompson, R. J., P. van Oosterom and D. Pullar, D. (2006) Connectivity in the Regular Polytope Representation, *submitted to GICON 2006, Spatial Data Handling*, Vienna.

Weisstein, E. W. (1999) Vector Space, *Math World -- A Wolfram Web Resource*, http://mathworld.wolfram.com/VectorSpace.html.

Weisstein, E. W. (2005) Field of Rationals, *Math World -- A Wolfram Web Resource*, http://mathworld.wolfram.com/FieldofRationals.html, 2005 (23/May/05).

APPENDIX 1 ENCODING OF THE REGULAR POLYTOPE

```
Encoding                Meaning

n                       Dimensionality (2 or 3)
{                       Start of regular polytope
ccc                     Number  of  convex  polytopes  in  this  regular
          [
          hhh           Number of halvers in this convex polytope
                (A
                aaa     Value of A
                B
                bbb     Value of B
                C       For 3D half spaces only
                ccc     Value of C, for 3D half spaces only
                D
                dddd    Value of D
                )
          ]
          uu            The unit number
}                       End of regular polytope
```

Figure 10: Encoding of the Regular Polytope.

```
2{3[4(A1000000000B-123291173D-69575189961077544)
(A-1000000000B-26136927D58393007284472408)
(A1000000000B31431823D-57996749042819608)
(A1000000000B34920016D-57735712615021376)]
1[3(A1000000000B26175247D-58388182445082504)
(A-1000000000B-41425581D57246948141295320)
(A-1000000000B-19410078D58894463521045944)]
1[4(A22044640B-1000000000D-76164657185192560)
(A-1000000000B-34920016D57735712615021376)
(A-20797546B1000000000D76091499563579888)
(A1000000000B19410078D-58894463521045944)]1}
```

Figure11: Sample of a 2D Regular Polytope with 3 Convex Polytopes.

```
3{4[5(A209106769B-1000000000C0D-87474526385019632)
(A-1000000000B96742947C0D67619782016733008)(A154141516B1000000000C0D65542411417979120)
(A0B0C1000000000D-1620000000000)(A0B0C-1000000000D3369000000000)]
1[9(A1000000000B-96742947C0D-67619782016733008)
(A124297655B-1000000000C0D-82353864426437184)(A42902139B-1000000000C0D-77439217084724288)
(A-1000000000B-485290904C0D24057595685639176)(A154151941B1000000000C0D65541781852024560)
(A0B0C-1000000000D3369000000000)(A321910248B312154508C1000000000D3926298596179034)
(A167386483B162313705C1000000000D20410138218737 28)
(A55882393B-39480156C1000000000D-633041539597006 6)]
1[9(A1000000000B485290904C0D-24057595685639176)
(A-39306006B-1000000000C0D-72475406303075040)(A-122497336B-1000000000C0D-67452131525276536)
(A-203536187B-1000000000C0D-62558733116505200)(A-1000000000B305424379C0D83245526342290960)
(A154151941B1000000000C0D65541781852024560)(A0B0C-1000000000D3369000000000)
(A-9290258B-60270964C1000000000D-3951380303692924)
(A-2668660B-67894450C1000000000D-4921817835264868)]
1[6(A-1000000000B305424379C0D83245526342290960)(A154151941B1000000000C0D65541781852024560)
(A0B0C-1000000000D1150000000000)(A-49838883B-1000000000C414957265D-71840804779952464)
(A-111385607B-1000000000C414957265D-68124465446020072)
(A-160311958B-1000000000C555632101D-65170280390171696)]1}
```

Figure 12: Sample of a 3D Regular Polytope with 4 Convex Polytopes (100/CP900152).

CVS OF THE AUTHORS

Rod Thompson, B.Sc. (Maths), Grad Dip Comp Sc, M. Eng Sc has worked in the computer industry since the early 1970's in various fields including Insurance, Shipping, and Local Government. His primary focus in the last twenty years has been on spatial databases. He led the development of the Queensland Digital Cadastre Data Base (DCDB) – one of the earliest to adopt the approach of including spatial data within the relational database. More recently he has provided technical advice in the construction of the Resource Information Management Environment (RIME).
He is currently undertaking research with the Delft University of Technology on the definition of rigorous spatial logic based on finite precision (computational) arithmetic.

Prof.dr.ir. Peter van Oosterom obtained a MSc in Technical Computer Science in 1985 from Delft University of Technology. In 1990 he received a PhD from Leiden University for this thesis "Reactive Data Structures for GIS" (updated version published by Oxford University Press, 1993).
From 1985 until 1995 he worked at the TNO-FEL laboratory in The Hague, the Netherlands as a computer scientist. From 1995 until 2000 he was senior information manager at the Netherlands' Kadaster, were he was involved in the renewal of the cadastral (geographic) database.
Currently Van Oosterom is head of the section 'GIS Technology' at the Delft University of Technology, OTB. As from 2005 on he and his staff members are involved in a large number of projects that are carried out within the framework of the Dutch 'Space for Geo-information Programme'. In 2005 he became a member of the core drafting team on data specification (INSPIRE). His main research themes are spatial database management systems, GIS architectures, generalization, spatial analysis, querying and presentation, Internet/interoperable GIS and cadastral applications.

CO-ORDINATES

Rodney James Thompson, MSc
Delft University of Technology
OTB, GIS technology
Jaffalaan 9
2628 BX Delft
The Netherlands
Landcentre – Department of Natural Resources
and Mines
Australia
E-mail : Rod.Thompson@nrm.qld.gov.au

Prof.dr. Peter van Oosterom
Delft University of Technology
OTB, GIS technology
Jaffalaan 9
2628 BX Delft
The Netherlands
E-mail : oosterom@otb.tudelft.nl
Website : www.gdmc.nl