

# Applying Spatial Reasoning to Topographical Data with a Grounded Geographical Ontology

David Mallenby and Brandon Bennett

School of Computing,  
University of Leeds,  
Leeds,  
LS2 9JT, UK

<davidm,brandon>@comp.leeds.ac.uk

**Abstract.** Grounding an ontology upon geographical data has been proposed as a method of handling the vagueness in the domain more effectively. In order to do this, we require methods of reasoning about the spatial relations between the regions within the data. This stage can be computationally expensive, as we require information on the location of points in relation to each other. This paper illustrates how using knowledge about regions allows us to reduce the computation required in an efficient and easy to understand manner. Further, we show how this system can be implemented in co-ordination with segmented data to reason about features within the data.

**Key words:** Grounding, Ontology, RCC, Spatial Reasoning, Vagueness

## 1 Introduction

Geographic Information Systems are becoming increasingly popular methods of representing and reasoning with geographical data. In order to do this, we require methods of reasoning logically about geographical features and the relations that hold between them, including spatially. Ontologies have been cited as a method to perform this reasoning [1-3], but existing methodologies do not handle the inherent vagueness adequately. Features are often dependant on the context in which they are made, with local knowledge affecting definitions. Geographical objects are often not a clearly demarcated entity but part of another object [1, 4]. The individuation of entities is therefore more important to geographical domains than to others.

One approach proposed to improve the handling of vagueness is to ground the ontology upon the data [5, 6], making an explicit link between the ontology and the data, thus allowing reasoning to be made within the context of the particular data. So we require approaches that will allow spatial reasoning such as Region Connection Calculus (RCC) [7] to be used. RCC is a powerful representation of the principal relations between regions, but it can also be computationally expensive.

In this paper we examine developing the system introduced in [6], which takes topographical data as input and segments into polygons with attached

attributes. The data to be looked at is of the Hull Estuary<sup>1</sup>, with the aim being to obtain a method of reasoning about the hydrological features implicit in the data. We examine how this segmented data can be stored effectively, and what is required in order to reason about the RCC relations between given polygons. Finally, we look at how these can be expanded to allow first order logic definitions of inland water features to be entered, with the appropriate regions returned. We do this by applying an example definition to see what results are returned.

## 2 Motivation

Vagueness is inherent to the geographical domain, with many features being context dependant, as well as lacking precise definitions and boundaries. Vagueness is not a defect of our language but rather a useful and integral part. It is a key research area of the Ordnance Survey<sup>2</sup>, where it has been noted that GIS does not handle multiple possible interpretations well. Rather than attempting to remove vagueness, we should allow the user to make decisions about vague features. So rather than segmenting or labelling the image in advance, we require a mechanism for entering logical queries that may incorporate vagueness and can segment accordingly.

With GIS, we need to deal with several layers. We have our initial data level, which represents the points and polygons that make up a topographical map for example. An additional layer is the ontology level, whereby we define features and relations between the data. The ontology level is usually seen as separate to the data level; we reason within the ontology, and return the data that matches our queries. Thus the ontology is devoid of the data context. This has a clear impact upon handling vagueness, where context is important. A proposed improvement to this is to ground the ontology upon the data [5]. By grounding the ontology, we make an explicit link between the ontology and the data, thus allowing reasoning to be made within the context of the particular data.

The symbol grounding problem as proposed in [8] suggests that computers do not actually understand knowledge they are provided, as meanings are merely symbols we attach to objects. There have been no adequate solutions to this problem as yet and it remains an open problem [9]. Ontology grounding does not solve the problem. Rather, it allows the user to decide the meaning of concepts to some extent.

Grounding the ontology upon the data allows reasoning with the data in particular context, thus achieving our previously mentioned requirement of allowing the user control over the features generated. To ground the ontology upon the

---

<sup>1</sup> Landsat ETM+ imagery. Downloaded from the Global Landcover Facility (GLCF). Image segmented into water and land then vectorised.  
<http://glcfapp.umiacs.umd.edu:8080/esdi/index.jsp>

<sup>2</sup> Ordnance Survey Research Labs: Modelling fuzzy and uncertain features  
[http://www.ordnancesurvey.co.uk/oswebsite/partnerships/research/research/data\\_fuzzy.html](http://www.ordnancesurvey.co.uk/oswebsite/partnerships/research/research/data_fuzzy.html)

data, we need to work at both the data level and the ontology level. In [6] linearity was shown as an example of such an attribute, and it was shown the work required on both levels to use such an attribute. To expand the system, we are required to implement approaches to generate polygons based upon the spatial relations between regions, such as if they are connected or disconnected.

Spatial reasoning can be computationally expensive, as we require information on the location of all points in relation to a given region. Previous work has looked at the problem at an abstract level [10]. By looking at how the relations are calculated, we can determine methods of reducing the calculations required based upon simpler observations. So instead of explicitly requiring every point location be determined, we could use other information to infer what relations are possible and reduce down our scope until we have our solution.

By implementing an RCC based system, we allow quantitative data to be reasoned with qualitatively. This significantly improves the expressiveness of GIS. This also allows for the individuation of features.

### 3 The Region Connection Calculus

The Region Connection Calculus (RCC) was introduced in [7]. RCC assumes an initial primitive relation  $C(x,y)$ , which is true if  $x$  and  $y$  share a common point). From this initial connected relation, we can derive other relations that hold between two regions. A list of the basic key relations as listed in [11] follows:

$$DC(x, y) \equiv_{df} \neg C(x, y) \quad (1)$$

$$P(x, y) \equiv_{df} \forall z [C(z, x) \rightarrow C(z, y)] \quad (2)$$

$$PP(x, y) \equiv_{df} [P(x, y) \wedge \neg P(y, x)] \quad (3)$$

$$EQ(x, y) \equiv_{df} [P(x, y) \wedge P(y, x)] \quad (4)$$

$$O(x, y) \equiv_{df} \exists z [P(z, x) \wedge P(z, y)] \quad (5)$$

$$DR(x, y) \equiv_{df} \neg O(x, y) \quad (6)$$

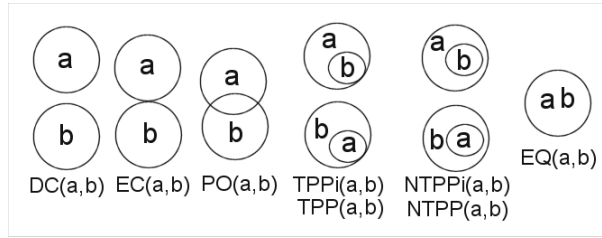
$$PO(x, y) \equiv_{df} [O(x, y) \wedge \neg P(x, y) \wedge \neg P(y, x)] \quad (7)$$

$$EC(x, y) \equiv_{df} [C(x, y) \wedge \neg O(x, y)] \quad (8)$$

$$TPP(x, y) \equiv_{df} PP(x, y) \wedge \exists z [EC(x, z) \wedge EC(y, z)] \quad (9)$$

$$NTPP(x, y) \equiv_{df} PP(x, y) \wedge \neg \exists z [EC(x, z) \wedge EC(y, z)] \quad (10)$$

RCC-8 consists of eight of these relations: DC, EQ, PO, EC, TPP, TPPi, NTPP, NTPPi, where TPPi and NTPPi are the inverses of TPP and NTPP respectively. Fig. 1 shows graphically the RCC-8 set. This set is both jointly exhaustive and a pairwise disjoint set of base relations, such that only one can ever hold between two given regions [7]. RCC has previously been proposed as a method of spatial reasoning that could be applicable to GIS, for example in [12] where it was noted that the same set of relations have independently been identified as significant for GIS [13, 14].



**Fig. 1.** The RCC-8 relations.

An additional property that we would like to express is the notion of self-connected regions, such that a region is self-connected if it is not divided into a number of DC parts. This is important, as in our system we will start with an initial set of segmented polygons, and wish to connect them to form larger regions that satisfy given properties. To do this, we first define a formula  $sum(x,y)$  which represents the spatial sum or union of two regions. From this we can define self-connectedness to be equal to the sum of a set of connected regions [11]:

$$\forall xyz[C(z, sum(x, y)) \leftrightarrow [C(z, x) \vee C(z, y)]] \quad (11)$$

$$CON(x) \equiv_{df} \forall yz[EQ(x, sum(y, z)) \rightarrow C(y, z)] \quad (12)$$

So equation 11 states that  $z$  represents the spatial sum of regions  $x$  and  $y$  if all parts of  $z$  are either connected to either  $x$  or  $y$ . This spatial sum is then used in 12 to define self-connectedness (CON); if  $x$  is self-connected, any two regions whose spatial sum is equal to  $x$  must be connected to each other. Thus  $x$  is a single connected region; if we imagine standing in any part of  $x$  it would be possible to travel to any other part of  $x$  without actually leaving the region.

## 4 Vagueness in Geography

Vagueness is ubiquitous in geographical concepts [15]. Both the boundaries and definitions of geographical concepts are usually vague, as well as resistant to attempts to give more precise definitions. For example, the definition of a river as given by the Oxford English Dictionary [16] is:

A large natural flow of water travelling along a channel to the sea, a lake, or another river.

The most obvious example of vagueness is 'large', though other aspects may also be vague such as the boundary between respective channels. But this isn't the only definition for a river; some may differ entirely, others may be more or less restrictive. In comparison, OpenCyc<sup>3</sup> is the open source version of Cyc, which is intended to be the largest and most complete general knowledge base in the world. The definitions of river and stream in OpenCyc are:

<sup>3</sup> OpenCyc <http://www.opencyc.org/>

A *River* is a specialisation of *Stream*. Each instance of *River* is a natural stream of water, normally of a large volume.

A *Stream* is a specialisation of *BodyOfWater*, *InanimateObject-Natural*, and *FlowPath*. Each instance of *Stream* is a natural body of water that flows when it is not frozen.

Again, these are vague and also do not include the restrictions of the water flowing into a particular feature. Yet at the same time, both definitions are perfectly valid within a given context to describe rivers.

The sorites paradox can be easily adapted to illustrate vagueness in geography [3, 17], showing that an explicit boundary may not always exist between definitions, such as between rivers and streams. Geographical definitions are also dependant on the context in which they are made. For example, whilst UK rivers are defined usually as permanent flows, in Australia this is not necessarily the case, and thus temporal aspects enter the definition [18]. The application of UK based definitions in Australia could therefore fail to classify some rivers due to their temporal nature, whilst Australian based definitions may overly classify things as rivers when applied in the UK.

The principal approaches for handling vagueness at present are fuzzy logic and supervaluation theory. It is usually the case that the two are presented as opposing theories. However, this in part assumes that vagueness can only take one form, which as discussed in [19] is not true. Rather, there are instances where it is more appropriate to use fuzzy logic and instances where supervaluation theory is better.

The suitability of the two approaches to the proposed system were discussed in [6], where it was noted that supervaluation theory was more applicable as crisp boundaries were produced. This means that we use *precisifications* to represent user decisions and to set contexts.

## 5 Data Segmentation

In [6], an initial polygon representing the inland water network extending from the Hull estuary was segmented based upon linearity thresholds. This was done by first finding the medial axis of the polygon using a voronoi diagram based approach VRONI [20]. The medial axis of a polygon as first proposed in [21] is defined as the locus of the centre of all the maximal inscribed circles of the polygon. Here, a maximal inscribed circle is a circle that cannot be completely contained within any other inscribed circle in the polygon [22].

However, the medial axis is extremely sensitive to noise and variation along the edge of the input polygon. We want to be able to prune off arcs such that the remaining arcs still represent the topology of the polygon effectively, without disconnecting parts or removing arcs we wish to keep. The approach used to prune the medial axis skeleton here was contour portioning [23, 24], which satisfies these requirements. The contours used here are manually defined; whilst an automatic approach is desirable (and work has been done in this area), it is beyond the scope of this project.

The results of using contour partitioning are shown in Fig. 2, where we see the remaining skeleton retains the topology whilst removing unnecessary arcs. This skeleton easily translates into a graph.



**Fig. 2.** The results of contour partitioning to reduce the medial axis of the Hull Estuary to a simplified skeleton whilst retaining topology of the shape.

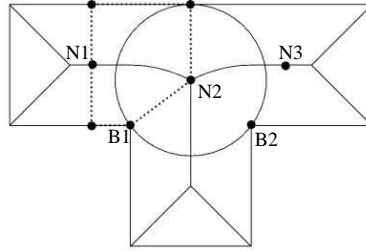
The next stage was to use this skeleton to determine linearity. In [6] this was done using a scale invariant approach looking at the variation in widths across stretches of the skeleton. From this we can determine linear lines in the skeleton

To generate a polygon from this, we determine a left and right side for the skeleton, and combine this with the end points to create a simple polygon. For each side, we use the two boundary points closest to both end nodes, which we already know as these are the points that the maximal inscribed circle at each point touches the boundary.

Once these two points are selected, we find the shortest path between the two along the boundary. This is done by representing the boundary as a graph, and thus a path between is easy to calculate. If no path exists or the length of the path is too great in relation to the distance between the points, then we simply use a single edge with the points as end nodes. An example of this is shown in Fig. 3. This approach guarantees a unique polygon for each line that is simple. We can also use the technique on sets of lines to generate larger polygons.

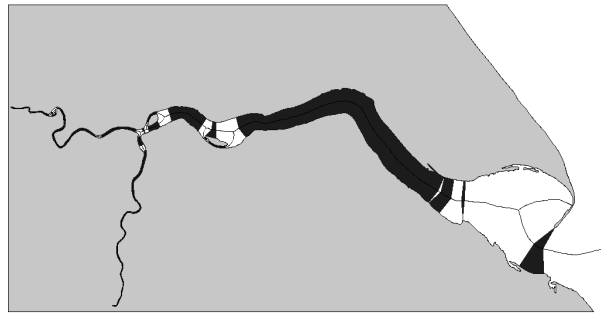
The initial results of this segmentation stage is a series of segmented polygons, with a label attached representing whether the polygon is linear or non-linear. Further attributes could be used to generate further polygons and labels, such as different linearity measures or size and distance measurements. Some may require further segmentation of the data, whilst others can be performed without segmenting.

The initial results of marking all linear polygons is shown in Fig. 4. Although most parts we would like marked as linear are marked as such, there are some cases that are not. These may be rectified with alternative or refined definitions of linearity. For example, the mouth of the river does not seem linear, because although the width is not varying, the difference in the two banks is significant.



**Fig. 3.** An example of how a line is translated into a corresponding polygon. For this shape, we have taken the line  $N1-N2$ . For  $N1$  we just use both tangent points. For  $N2$  we choose  $B1$  as it is closer to  $N1$ . We then trace along the boundary using the shortest path, with the dotted line representing the resultant polygon. Had the line been  $N1-N2-N3$ , depending on the length around the boundary between  $B1$  and  $B2$  we may replace the path with edge  $B1-B2$  instead.

Therefore a refined linearity definition may be that a polygon is required to also be linear in relation to its edges, in that the length of the edges should not vary too great from the length of the stretch.



**Fig. 4.** The results of marking linear stretches, with the original skeleton once again shown. Black sections represent polygons marked as linear with respect to the width.

However, there are always likely to be discrepancies in our data, because of variations in actual data in comparison to our abstract notion of a river as a constant line. So we would like a mechanism that can flag up such small discrepancies so that they can be filled in. A method for this was discussed in [6], where the discrepancies were referred to as gaps. To avoid confusion we have introduced the term *interstretch* to represent these features. So using a closeness threshold we can determine which polygons could be 'filled in' at a higher level to generate connected stretches.

## 6 Data storage and Querying

Our initial system allows us to segment our data into a series of linear or non-linear polygons, as well as identify *interstretches*. However, we would like to reduce the amount of pre-computed features used, as the aim is to allow a user to generate their own definitions. For example, rather than explicitly calculating *interstretch*, we would like to be able to identify these parts based upon first order logic. So example definitions of stretch and *interstretch* are:

$$\text{stretch}[l](x) \leftrightarrow \text{CON}(x) \wedge P(x, \text{WATER}) \wedge \text{linear}[l](x) \quad (13)$$

$$\begin{aligned} & \wedge \forall y (P(y, \text{WATER}) \wedge \text{linear}[l](y) \\ & \wedge P(x, y) \rightarrow \text{EQ}(x, y)) \end{aligned}$$

$$\text{interstretch}[c, l](x, y, z) \leftrightarrow \text{stretch}[l](x) \wedge \text{stretch}[l](y) \quad (14)$$

$$\wedge P(z, \text{WATER}) \wedge \text{EC}(x, z) \wedge \text{EC}(y, z)$$

$$\wedge \text{CON}(z) \wedge \forall w (PP(w, z)$$

$$\rightarrow (\text{close} - \text{to}[c](w, x) \wedge \text{close} - \text{to}[c](w, y)))$$

Here, we use the form  $p[v](x)$ , where the predicate  $p$  is true for a given variable or *precisification*  $v$  for a given variable  $x$ . So, our previous definition of something being linear if the variation in widths is low translates to  $\text{linear}[l](x)$ , or  $x$  is linear for a given *precisification*  $l$ . Equation 13 defines stretch as a maximal self-connected region that is water and linear for a given *precisification*. Equation 14 defines an *interstretch* as a self-connected region of water that is connected to two stretches, such that all parts of the *interstretch* are close to the two stretches.

Now, instead of having *interstretch* as a primitive, we have a primitive *close-to*, representing the notion that they are close if the distance between is insignificant. As with linearity, this can be treated as a *precisification*. From these definitions, we wish to define water-channels to be maximal self-connected regions that are made up of stretches or *interstretches*. An initial attempt at representing this logically is:

$$\text{waterchannel}[c, l](x) \leftrightarrow \text{CON}(x) \wedge P(x, \text{WATER}) \wedge \forall w (PP(w, x) \quad (15)$$

$$\rightarrow \exists s (\text{stretch}[l](s) \wedge P(w, s) \wedge \text{TPP}(s, x)) \vee$$

$$\exists d, e, f (\text{interstretch}[c, l](d, e, f) \wedge P(w, f)$$

$$\wedge \text{TPP}(d, x) \wedge \text{TPP}(e, x) \wedge \text{TPP}(f, x))$$

So a water-channel is a self-connected region of water such that all proper parts of the region are either part of a stretch or an *interstretch*, that is also proper parts of the channel.

This is not the only way such a query could be formed, and there may be further refinements required in order to capture exactly our intended definition. In order to represent a query such as water-channel, we require several stages of work. First, we need a data representation that allows more effective querying.

We then need to consider how we can test for RCC relations. Finally, we then need a method of producing the union of resultant polygon sets to produce the final water-channel result.

Our aim at each stage is to find a balance between simplicity and computational complexity. The system is intended to use logic definitions that may not be known at this stage. So to accommodate for this, our design should be reasonably easy to understand and adapt, whilst remaining reasonably efficient.

## 6.1 Data storage

The winged edge structure [25] and variations such as the half-edge winged structure [26] offer a more effective representation of polygons as opposed to simply storing the corner points. Our initial polygon data can be easily translated into such a structure. We can easily gain a list of all polygons, edges and points in Prolog.

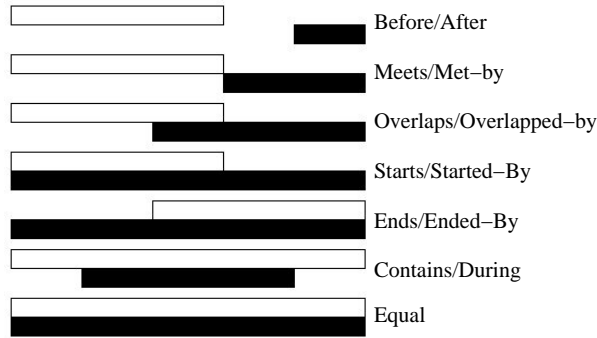
## 6.2 Calculating the RCC relations

We now move on to encoding RCC relations such that we can query the system to find the relations between polygons, thus allowing qualitative and quantitative queries. However, this move from an abstract level to the data level is computationally expensive. We therefore wish to reduce the calculations required at each stage in order to speed up the reasoning process. Previous work on an abstract level was done in [10], which illustrated the process could be broken down into a hierarchical tree, reducing the calculations required.

We first reduce down the potential relations that can occur between two polygons. A first approximation is to compare the bounding boxes of each polygon, hereby defined as the smallest rectangle that can entirely contain its polygon. This significantly reduces the initial calculations, and allows us to eliminate relations that are not possible. We do this using an approach similar to Allen's interval Algebra [27]. The algebra represents 13 different relations (hereby referred to as Allen relations) that can occur between two time intervals, as shown in Fig. 5.

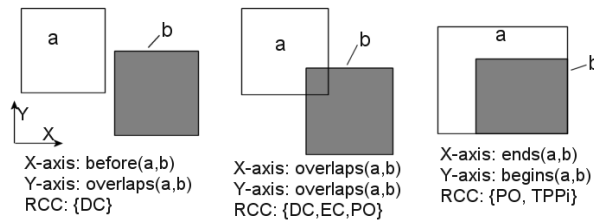
If we treat the X-Axis and Y-Axis as separate dimensions, we can determine the Allen relations between two polygons in each axis. We then compare the resulting pair of Allen relations and determine what possible RCC relations these allow. Determining the Allen relations is straightforward; for a given axis we find the minimum and maximum values of the two polygons and represent as two lines. We can then sort these numbers and determine what Allen relation they correspond to. We repeat for the other axis, so each operation is only working in a single dimension.

This results in a pair of Allen relations, which in turn represent a set of possible RCC relations, as shown in 6. In these examples, we have quickly determined that the first example shows two disconnected polygons, thus no further computation is required. With the other two examples we are left with a set of possible relations. However, we can use these reduced sets to determine the most



**Fig. 5.** A graphical representation of the 13 different Allen relations. With the exception of the final relation Equals, the other 12 are in fact 6 pairs of duals. So the first relation represents both white before black and black after white.

effective approaches to take next, thus tailoring our deductions to each pair of polygons.



**Fig. 6.** Examples of how the bounding boxes of two polygons  $a$  and  $b$  may be related spatially, and what possible RCC relations these represent. We obtained the Allen relations for the X- and Y-axis, then compared these to see what the set of possible RCC relations are for the polygons.

Theoretically there are 169 different combinations, but in fact there are only 14 different possible combinations, listed in table 1. So we now have a method of reducing the possible RCC relations quickly; we can for example quickly determine polygons which are definitely disconnected.

Our calculations for RCC relations are based upon the locations of the corners of the polygons to be compared, and whether they are inside, outside or on the boundary of the other polygon in question. In addition, we add to our set all points of intersection between the two polygons that are not already a corner point of a polygon. Table 2 shows the RCC-8 relations defined in terms of the tests that are required in order to make decisions as to the RCC relation

**Table 1.** The possible relations as a result of comparing the Allen relations between the X- and Y-axis. Starred relations also have versions replacing TPP/NTPP with TPPi/NTTPi

Possible RCC combinations from previous stage
DC
DC, EC
DC, EC, PO
DC, EC, PO, TPP *
EC, PO
EC, PO, TPP, NTPP *
EC, PO, TPP, TPPi, EQ
PO
PO, TPP *
PO, TPP, NTPP *

between two regions. This is similar to [28], where the spatial domain was also restricted to polygons as opposed to arbitrary points due to the existence of efficient algorithms to handle polygons.

**Table 2.** The definitions of the RCC-8 relations used in the system. Only 6 are shown here, as TPPi and NTTPi are merely the inverses of TPP and NTPP respectively.

RCC	Definition in system
DC(X,Y)	There is no intersection between X and Y, and no point of X is inside or on the boundary of Y (and vice versa)
EC(X,Y)	There exists a point that is on the boundary of both X and Y, but there are no points of X inside Y (or vice versa)
PO(X,Y)	There exists either at least one intersection between X and Y, and there are points that are inside one polygon but outside the other
TPP(X,Y)	All points of X are either inside or on the boundary of Y
NTPP(X,Y)	All points of X are inside Y
EQ(X,Y)	All points of X and Y lie on the boundary of each other

### 6.3 Intersections

The intersections of two polygons has been studied extensively, in an attempt to improve upon the brute force approach of comparing all lines against all others. More efficient methods are based upon the sweepline approach [29]. The aim of such algorithms is to reduce the comparisons between lines. For our approach, we use our Allen relations based approach to reduce the number of intersection tests.

Using the brute force algorithm as our basis, we order the polygons into two sets of lines. We then take a line in our first set and compare against the

bounding box second set, since if an intersection exists the line must touch, intersect or be inside this bounding box. So using our Allen relations approach we can quickly test if the line falls inside the box, and thus eliminate lines that could not intersect the second polygon. For a line that satisfies this criteria, we wish to improve upon simply then comparing the line against all others. We once again use Allen relations, as lines can't intersect if they occur before/after each other in either axis. This once again eliminates many lines, leaving only a small set to be tested.

One final consideration is which of the polygons to use as our first set, as this choice can further speed up the process. Looking at the possible relations, we first see if the relation PP is possible. If so, we use the outer polygon, as our bounding box test would remove no lines if the inner polygon was used. If the relation PP is not possible, we use whichever polygon has fewer lines, as this will remove more intersections in the first part of the test.

So our intersection algorithm uses information previously calculated to speed up the calculation of all intersects whilst remaining simple to understand. Although further work is required to determine the actual efficiency of this approach in comparison to others, it has so far been successfully implemented within Prolog, where it has proven fast enough for the requirements of the project. The result of this stage is a set of points of intersection, which may include existing corner points of a polygon. These can be separated into existing and new points using set operations.

#### **6.4 Points inside**

As with our intersection tests, we wish to reduce down the number of points we test to keep computation time down. Using the bounding boxes generated previously, we can again reduce down the possible points to be all those that are inside or touching the bounding box. This subset is then tested to find which points are inside using a standard test of extending a line horizontally from the point and then counting the number of intersections with the polygon; if the number of intersections is odd the point is inside and if it is even the point is outside. We can reduce the number of intersection tests by using Allen relations to eliminate lines that could not intersect the projected line. How to handle points that lie on the boundary is often an issue for such algorithms. However, we have previously found this set of boundary points in our intersection tests and so can use this set to remove points on the boundary, leaving only points explicitly inside.

#### **6.5 Using the results with RCC**

The results of the previous stages give us a series of sets of points. We can therefore test for RCC relations using set operations on these points, as our previous definitions easily translate into set operations.

First, we find all the potential RCC relations using the Allen relations based approach mentioned earlier. From this stage we can make decisions based on

which tests to do; for example if the results of this stage is the set [DC,EC], we know we only need to test for at least one intersection to determine if the answer is DC or EC. So for each of these sets of possible relations we can order the queries to be asked so that they are optimal. We can also find other relations that are implied; if DC is not a member of the list then we know that the two regions are connected, whereas if DC and EC have both been removed we know that at least some part of one region is part of the other.

By using Prolog, we are able to allow for variations of the query. So instead of simply being able to return the relation between two polygons, we can also ask such queries as "find all polygons that are connected to X" and "find all pairs of polygons that are externally connected".

## 6.6 Building new regions based on queries

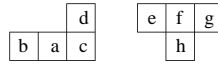
The aim of our system is to return regions that match particular queries from an ontology, so we require the system to be able to return sums of regions. For our water-channel example, we need to find all linear polygons, as well as all *interstretch* polygons that connect linear polygons together, and return the results as single connected regions.

We have previously defined self-connected as being the sum of connected regions. This is also applicable to our skeleton and the associated graph, whereby any subset of this graph can be considered self-connected if there is a path between all nodes in the subgraph generated from the subset. As illustrated in Fig. 3, the polygon generated for any given line is simple and self-connected. Thus using this technique on sets of lines is the equivalent of taking the union of all the polygons generated from all connected subsets of the set of lines. We can thus infer that the resulting polygon is self-connected if the skeleton used to generate it is a connected graph. To produce all linear polygons, we simply find the set of all linear lines and convert into a graph, then generate polygons for each maximal self-connected component, where maximal means that there does not exist an edge that is connected to our component that is not part of the component. For *interstretch*, we find the set of lines used to produce the polygons that satisfy our definition and repeat the process above (thus some non-linear polygons may have more than one *interstretch* proper part).

To generate our maximal self-connected polygons, we use an approach similar to a breadth-first search, marking neighbours of polygons as we find them. An example of this process is shown in Fig. 7.

So for our water-channel example, our criteria is that all polygons are either linear or an *interstretch* between linear polygons. We find this set, then using our breadth-first search type approach, travel through all connections until all have been visited. The result is sets of maximal self-connected regions.

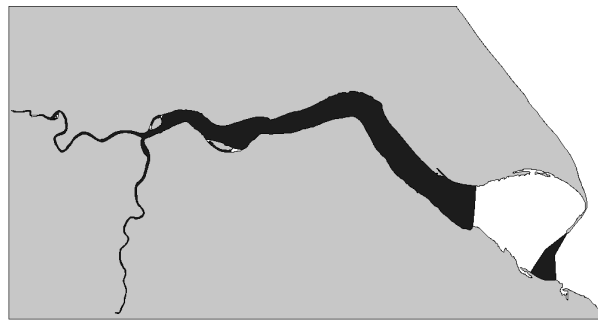
We now wish to generate the sum of these polygons, to create our new polygon representing a water-channel. For this, we can use our winged edge structure coupled with our polygon generation approach. Firstly, if we have a set of polygons that are only ever EC, we can find the union by removing all edges that are incident to two or more polygons and traveling along the remaining edges,



**Fig. 7.** An example of how self-connected regions are marked. Starting at  $a$  our breadth-first search returns the set  $a, b, c, d$ , and then finds polygons remain, so repeats to get  $e, f, g, h$ . These sets can then be spatially summed to return maximal self-connected regions.

returning a polygon when we reach our start point (if further edges remain, these are holes and we simply repeat the process until there remains no unvisited edges). If we have overlapping polygons, then we can use our polygon generation approach to create a union by combining the sets of lines that make up our polygons and generating a new polygon that represents their union. We could simply use this approach for the union of all polygons, but this is slower than the union of existing polygons.

So our sum operation combines the previous operations; we find our set of candidate polygons, find maximal self-connected sets via our breadth-first search and then form the union either through union or additional polygon generation. Further operations such as spatial difference or intersection could also be developed, but are beyond the scope of this work. The results of running our water-channel query are shown in Fig. 8, where we see stretches have successfully been joined to form larger regions.



**Fig. 8.** The results of running the water-channel query. The linear stretches were segmented first, then a query representing interstretch was run. Finally, a water-channel was defined to be the self-connected sum of these two features, such that we find the set of polygons that are either linear or an interstretch, then used our traveling algorithm to find maximal self-connected sets.

## 7 Future Work

The next key stage of the research is into further logical definitions that can be used to represent inland water features, and thus construct an ontology that represents such features. This may require further primitive functions to be implemented in addition to the linearity and closeness tests present in the system. However, the aim is to keep such primitives to a minimum, as the system is intended to be as general as possible. Thus new features should be defined in logical definitions at the ontology level.

The system has been developed in Prolog and at present is designed to use first order logic based queries. However, a possible extension would be to integrate more closely with a language such as OWL, which can be inputted into Prolog [30, 31]. By creating the ontology in OWL, we allow interaction with the semantic web, whilst retaining the segmentation level in Prolog allows us to reason with vague features and ground the ontology upon the data effectively. This is also proposed in [32], where it is shown that OWL cannot effectively handle RCC without modifying the rules of the language. However, such revisions may remove other favourable features of OWL, hence a hybrid system is more appropriate.

## 8 Related Work

The problem of combining qualitative and quantitative data has previously been discussed in [33]. Here, the combination of different levels of information are discussed, such that the intention is to bridge the gap between the primitive level of points, lines and polygons, and the object level describing the spatial relations and definitions of features.

Like the Allen relation based approach used in this paper, transitivity tables are formed representing the possible relations between different primitives. Thus, spatial relations can be calculated by deductive processes as opposed to computational geometric algorithms (or at least a reduced usage of such algorithms). In this paper, we have expanded this to show how intersection and point locations can be determined using similar approaches to reduce the computational geometry requirements.

As previously mentioned, [10] discussed a hierarchical approach to determining RCC relations. Moreover, the calculations were converted to boolean terms, such that the problem becomes one of the closure of half-planes. On the other hand, in this paper decisions are made based on both the intersection and location of points with respect to the regions. Thus a richer level of detail is deductible.

Another approach to deducing the spatial relationships is to use constraint logic programming [34], as discussed particularly in [35]. Such an approach offers an interesting alternative, but is reliant on the efficiency of the constraint logic solver used, and as discussed in [35] further work is required to improve such an approach for effective implementation.

## 9 Conclusion

In this paper we have demonstrated a method of calculating and using RCC relations on segmented topographical data, thus allowing integration with an ontology grounded upon the data. This improves the handling of vagueness within geographical features, as we can make decisions on features based upon the context in which they are made, as opposed to using predefined regions.

We have shown that although the calculation of RCC relations is computationally expensive, we can still implement the relations effectively by using other knowledge representation approaches such as Allen's interval algebra. Further, Allen's relations were adapted to provide simple but effective methods of calculating the intersections and locations of points of polygons in relation to each other, although more efficient algorithms may exist. Further work is therefore required to determine the efficiency of the approaches discussed here, or whether a hybrid approach using deductive methods in conjunction with other computational geometric algorithms, thus providing the most efficient environment overall.

We have shown how previous queries used in the system could be written in first order logic instead of being specified in the code. Although these may require further clarification, this does highlight the possibility of defining features in first order logic. We have also shown how maximal self-connected regions satisfying such queries can be generated. Finally, we have shown where the work is intended to progress and how this will improve the handling of vagueness within geographical features.

**Acknowledgments.** I am grateful to our industrial partner Ordnance Survey, for their contribution towards PhD funding. I am also grateful to Brandon Bennett and Allan Third for their support. Finally, I am extremely grateful to the reviewers for their comments that helped me prepare the final text.

## References

1. Fonseca, F., Egenhofer, M., Agouris, C., Cmara, C.: Using ontologies for integrated geographic information systems. *Transactions in Geographic Information Systems* **6**(3) (2002) 231–257
2. Guarino, N., Welty, C.: Evaluating ontological decisions with ontoclean. *Communications Of The Acm* **45**(2) (2002) 61–65 ISI Document Delivery No.: 515HL.
3. Varzi, A.C.: Vagueness in geography. *Philosophy & Geography* **4**(1) (2001) 49–65
4. Smith, B., Mark, D.M.: Ontology and geographic kinds. In Poiker, T., Chrisman, N., eds.: *Proceedings of the Tenth International Symposium on Spatial Data Handling*, Burnaby, BC, Simon Fraser University (1998) 308–320
5. Jakulin, A., Mladenić, D.: Ontology grounding. In: *Proceedings of the 8th International multi-conference Information Society IS-2005*, Ljubljana, Slovenia (2005) 170–173

6. Mallenby, D.: Grounding a geographic ontology on geographic data. In Amir, E., Lifschitz, V., Miller, R., eds.: 8th International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense 2007), Stanford, USA (2007) 101 – 105
7. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic-based on regions and connection. In: Principles Of Knowledge Representation And Reasoning: Proceedings Of The Third International Conference (Kr 92). Morgan Kaufmann Pub Inc, San Mateo (1992) 165–176
8. Harnad, S.: The symbol grounding problem. *Physica D* **42**(1-3) (1990) 335–346 ISI Document Delivery No.: DR770 0167-2789.
9. Taddeo, M., Floridi, L.: Solving the symbol grounding problem: a critical review of fifteen years of research. *Journal Of Experimental & Theoretical Artificial Intelligence* **17**(4) (2005) 419–445 Review 0952-813X.
10. Bennett, B., Isli, A., Cohn, A.G.: A system handling rcc-8 queries on 2d regions representable in the closure algebra of half -planes. In: Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA-AIE), LNCS (Berlin: Springer-Verlag) (1998)
11. Giritli, M.: Who can connect in rcc? In: *Ki 2003: Advances In Artificial Intelligence*. Volume 2821 of Lecture Notes In Artificial Intelligence. Springer-Verlag Berlin, Berlin (2003) 565–579 ISI Document Delivery No.: BX96D.
12. Bennett, B.: The application of qualitative spatial reasoning to gis. In Abrahart, R., ed.: *Proceedings of the First International Conference on GeoComputation*. Volume I. (1996) 44 – 47
13. Egenhofer, M.J.: Reasoning about binary topological relations. *Lecture Notes In Computer Science* **525** (1991) 144–160 ISI Document Delivery No.: GJ204 0302-9743.
14. Egenhofer, M.J., Franzosa, R.D.: Point-set topological spatial relations. *International Journal Of Geographical Information Systems* **5**(2) (1991) 161–174 ISI Document Delivery No.: FZ572 0269-3798.
15. Bennett, B.: What is a forest? on the vagueness of certain geographic concepts. *Topoi-An International Review Of Philosophy* **20**(2) (2001) 189–201 ISI Document Delivery No.: 480MJ Article 0167-7411.
16. Simpson, J., Weiner, E., eds.: *Concise Oxford English Dictionary*. 11 edn. Oxford University Press (2004)
17. Varzi, A.C.: Vagueness, logic, and ontology. *The Dialogue. Yearbooks for Philosophical Hermeneutics* **1** (2001) 135–154
18. Taylor, M.P., Stokes, R.: When is a river not a river? consideration of the legal definition of a river for geomorphologists practising in new south wales, australia. *Australian Geographer* **36**(2) (2005) 183–200
19. Dubois, D., Esteva, F., Godo, L., Prade, H.: An information-based discussion of vagueness. In: 10th IEEE International Conference On Fuzzy Systems, Vols 1-3 - Meeting The Grand Challenge: Machines That Serve People. IEEE PRESS (2001) 781–784 ISI Document Delivery No.: BV21J Article.
20. Held, M.: Vroni: An engineering approach to the reliable and efficient computation of voronoi diagrams of points and line segments. *Computational Geometry-Theory And Applications* **18**(2) (2001) 95–123
21. Blum, H.: Biological shape and visual science.1. *Journal Of Theoretical Biology* **38**(2) (1973) 205–287 ISI Document Delivery No.: O8126 0022-5193.

22. Ge, Y.R., Fitzpatrick, J.M.: Extraction of maximal inscribed disks from discrete euclidean distance maps. In: 1996 Ieee Computer Society Conference On Computer Vision And Pattern Recognition, Proceedings. Proceedings / Cvpr, Ieee Computer Society Conference On Computer Vision And Pattern Recognition. IEEE PRESS (1996) 556–561 ISI Document Delivery No.: BF92T.
23. Bai, X., Latecki, L.J., Liu, W.Y.: Skeleton pruning by contour partitioning. In: Discrete Geometry For Computer Imagery, Proceedings. Volume 4245 of Lecture Notes In Computer Science. Springer-Verlag Berlin, Berlin (2006) 567–579 ISI Document Delivery No.: BFG01.
24. Bai, X., Latecki, L.J., Liu, W.Y.: Skeleton pruning by contour partitioning with discrete curve evolution. IEEE Transactions On Pattern Analysis And Machine Intelligence **29**(3) (2007) 449–462 ISI Document Delivery No.: 125CY 0162-8828.
25. Baumgart, B.G.: Winged edge polyhedron representation. Technical report, Stanford University (1972) 891970.
26. Mantyla, M.: Introduction to Solid Modeling. W. H. Freeman & Co. (1988) 60949.
27. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications Of The Acm **26**(11) (1983) 832–843 ISI Document Delivery No.: RS929 Article 0001-0782.
28. Haarslev, V., Lutz, C., Möller, R.: Foundations of spatioterminological reasoning with description logics. In: Principles of Knowledge Representation and Reasoning. (1998) 112–123
29. Bentley, J.L., Ottmann, T.A.: Algorithms for reporting and counting geometric intersections. Ieee Transactions On Computers **28**(9) (1979) 643–647 ISI Document Delivery No.: HQ394 0018-9340.
30. Laera, L., Tamma, V., Bench-Capon, T., Semeraro, G.: Sweetprolog: A system to integrate ontologies and rules. In: Rules And Rule Markup Languages For The Semantic Web, Proceedings. Volume 3323 of Lecture Notes In Computer Science. Springer-Verlag Berlin, Berlin (2004) 188–193 ISI Document Delivery No.: BBF01 English.
31. Wielemaker, J.: An optimised semantic web query language implementation in prolog. In: Logic Programming, Proceedings. Volume 3668 of Lecture Notes In Computer Science. Springer-Verlag Berlin, Berlin (2005) 128–142 ISI Document Delivery No.: BDF61 English.
32. Grütter, R., Bauer-Messmer, B.: Towards spatial reasoning in the semantic web: A hybrid knowledge representation system architecture. In Fabrikant, S., Wachowicz, M., eds.: The European Information Society: Leading the Way With Geoinformation. Volume XVII of Lecture Notes in Geoinformation and Cartography. (2007) 486
33. Abdelmoty, A., Williams, M., Paton, N.: Deduction and deductive database for geographic data handling. In: Symposium on Large Spatial Databases. (1993) 443–464
34. Jaffar, J., Maher, M.J.: Constraint logic programming: A survey. Journal of Logic Programming **19/20** (1994) 503–581
35. Almendros-Jimenez, J.: Constraint logic programming over sets of spatial objects. In: Proceedings of the 2005 ACM SIGPLAN workshop on Curry and functional logic programming, Tallinn, Estonia, ACM Press (2005) 32–42 1085106 <http://doi.acm.org/10.1145/1085099.1085106>.