Implementing Anchoring

James Hood and Antony Galton

University of Exeter, UK, {J.M.Hood, A.P.Galton}@exeter.ac.uk

Abstract. In an earlier paper we introduced Anchoring, a new approach for handling indeterminate location in a spatial information system. Here we develop it further, showing how Anchoring can be made to work in real systems. In particular, we specify a new kind of locational component for a spatial data model. We then show how that component can be implemented into an object-relational database with extensions conforming to the OpenGIS Consortium's Simple Feature Model. We argue that Anchoring, in contrast to other formalisms for handling indeterminate location, is better suited to the needs of a spatial data provider who, in supplying data to different organisations, needs to be authoritative, and thus does not want to compromise data quality by representing indeterminate data with unwarranted precision. Anchoring, in providing new ways to describe spatiality, allows that we state only what we know for certain.

Introduction

In [9] we introduced Anchoring, a new approach for handling indeterminate location in a spatial information system. There our description of Anchoring was in terms of a high-level extension to a spatial information system. This led to a partial formalisation of Anchoring as a first-order logical theory. In this paper we drop down from those heights of abstraction and start to show how Anchoring can work in real systems. In particular, we present Anchoring as a new kind of locational component for a spatial data model. We then show how that component can be implemented into an object-relational database with extensions conforming to the OpenGIS Consortium's Simple Feature Model.

1 Anchoring In Brief

Before all that, though, let us introduce Anchoring. We do this by placing it in contrast to the 'classical' model of location that we hope to replace.

1.1 The Classical Model And Its Limitations

Consider Figure 1. We can think of this as presenting the structure of a data model in a typical spatial database. In it there are spatial objects (or *features*, as they sometimes called), which are things with 'whereness'. They are the objects with which the spatial database is concerned. The spatial objects populate

a part of the spatial data model—what we call the *information space*. The information space captures and records all the various non-spatial attributes that the spatial objects have and the non-spatial relations that they stand in. By contrast, spatial attributes and spatial relationships for a spatial object are not stored explicitly in information space. Rather, they are stored implicitly through the link between a spatial object and its spatial location. Spatial locations are the regions of what we call *precise space*, a mathematical representation of space with a coordinate reference system.[1] In a typical spatial database the precise space is the coordinatised Euclidean plane representing part of the Earth's surface. Spatial objects are associated with regions on the plane, and we consider the region to which a spatial object is tied as its spatial location. The key point to note here is this. In the classical model, there is just *one way* of tying the spatial objects to regions in the precise space. As we put it, since they must be coextensive with a region in the precise space, all spatial objects must have 'exact location'.

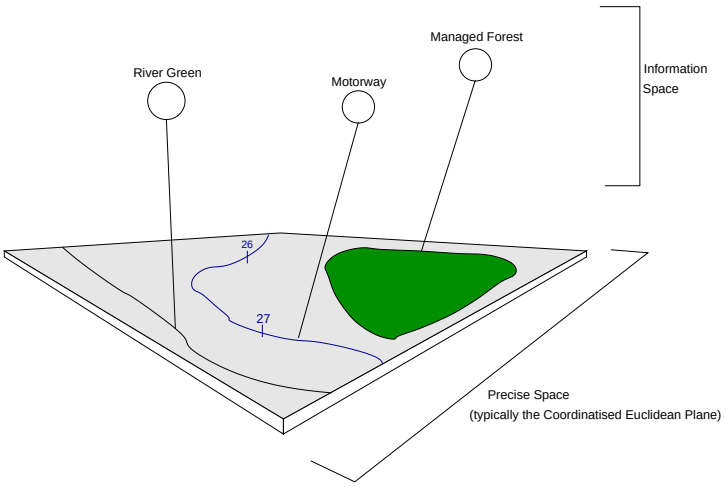

Fig. 1. Classic model of location using 'exact location'

This conclusion—that to be a 'spatial' object the object must have an exact location—means we must be able to locate the spatial object exactly on some region in a precise space. So those objects that cannot be tied to exact location cannot be represented in the data model. Hence there is no room for objects such as hills, valleys and other geomorphological features, whose location, either because of its inherent vagueness or of our lack of knowledge, cannot be specified precisely.

It may help at this point to recall some examples we presented in [9]. We will return to these later on in the paper.

[1] What we call a 'precise space' is usually just called a 'spatial reference system'.

Our first example is Greendale, in which runs the River Green. The river, as is shown by its representation in Figure 1, can be considered as having an exact location. That is, within the confines of a spatial data model, we can represent the River Green as a linear geometric construct. This, we might say, is 'good enough' for the purposes of our application. Greendale, by contrast, does not have exact location, as it is a geomorphological feature and part of the natural landscape. While a competent (human) map user would recognise such a feature not so much as an object but rather as implicit in a certain configuration of contour lines, the classical spatial data model is restricted to tying objects to regions. Consequently, Greendale, if it is to be represented at all, must be identified with some precise region. But since Greendale is lacking a precise boundary, any region we choose will invariably be a limited approximation.

Moving away from vagueness, our second example deals with uncertainty. The Highway Management Office in Devon County Council uses a spatial database to keep track of, among other things, road accidents. Road accidents are 'spatial' objects in virtue of their association with geometric points that represent accident locations. Using the classical model, though, accidents cannot be recorded in the database unless they can be given an exact location. But such information may not be available. For example, suppose we want to record information about an accident for which all we know about its location is that it happened 'somewhere' between junctions 26 and 27 of the M5 motorway. Such information cannot be entered into our database unless we can associate the accident with some region. Perhaps one way of overcoming this might be to relax the constraint that accidents must be located on points and allow them to be located on extended geometric regions instead. That way the accident could be located on the region occupied by the particular stretch of motorway, and so could be entered into the database. However, that would lead to some anomalous answers to queries since, to the information system, the accident would have to be considered as coextensive with that region. The accident might, for example, have other accidents located within it! Now we, as competent map users, knowing what accidents are, would understand what was really meant by having the accident associated to the region in this way. But the spatial database would not, since for it there is just one kind of locational relation between objects and regions.

Our final example features an odd mix of uncertainty and vagueness. It is loosely based on a real-life example. Consider an area of managed forest which falls within the confines of a national park. The managed forest is home to a rare species of butterfly. This population is, of course, located in space, but we can see that its spatiality is different from the kind possessed by other features in the spatial data model such as rivers, buildings and accidents (provided we do know where such accidents happen). Because observations have been made there, the population is definitely known to inhabit two particular areas in the forest. However, these are not the only places that the butterflies inhabit, and it is taken as given that the population is scattered all over the forest. What the users want is to loosely associate the location of the butterfly population with the entire

extent of the managed forest, but without thereby implying that every point within that extent is at every, or any, time occupied by a butterfly. As a makeshift compromise the users have added a textual 'note' onto the forest's database record which states the existence there of the butterfly population. Needless to say, though, such a note lacks any logical structure and so is unusable for the spatial information system.

1.2 Enter Anchoring ...

Uncertainty and vagueness are very different. On the one hand, under a common sense view, vagueness seems to do with the entity we are trying to represent, while on the other, uncertainty seems to do with our (lack of) knowledge about the entity. That the location of Greendale, for example, is indeterminate seems to be something intrinsic to Greendale itself, and not to us; while the indeterminacy of the accident is to do with us and our lack of knowledge, and nothing to do with the accident. Despite their differences, though, vagueness and uncertainty have enough commonality in the representational problems they raise to justify our treating them together, at least in some respects. In both cases we find a kind of *indeterminacy* which prevents us from using the classical model—that is, we cannot record in the database information about the location of the spatial object. That fact is our motivation for Anchoring, a new kind of representational framework in which we treat vagueness and uncertainty together.

Anchoring gives us more than one way of relating objects in information space to regions in precise space. We spoke earlier of there being just one locational relation between the two spaces, namely exact location. Anchoring in effect supplements exact location with other locational relations. The two most important locational relations are **anchored in** and **anchored over**. Their meanings are as follows. We say

- a spatial object **o** is *anchored in* some region R if and only if, whatever may be the nature of **o**'s location, we can certainly say that **o** is located inside R.
- a spatial object **o** is *anchored over* some region R if and only if, whatever may be the nature of **o**'s location, we can certainly say that it contains the whole of R.

In addition, for compatibility with the classical model, we stipulate that an object has exact location (in the sense of the classical model) if and only if there is some region which the object is both anchored in and anchored over.

The result of modelling our examples with the anchoring relations is shown in Figure 2. In our accident example, we now say that the accident is 'anchored in' the stretch of motorway between junctions 26 and 27 of the M5. That is to say, though we know that the accident is indeed located exactly at some particular location, at present we don't know where that location is, only that it is somewhere within the stretch of motorway. Accidents may still be constrained

to be points, but, as it happens, the exact point on which this particular accident is located is unknown to us, and so the spatiality of the accident must be represented in the spatial information system in a different way.

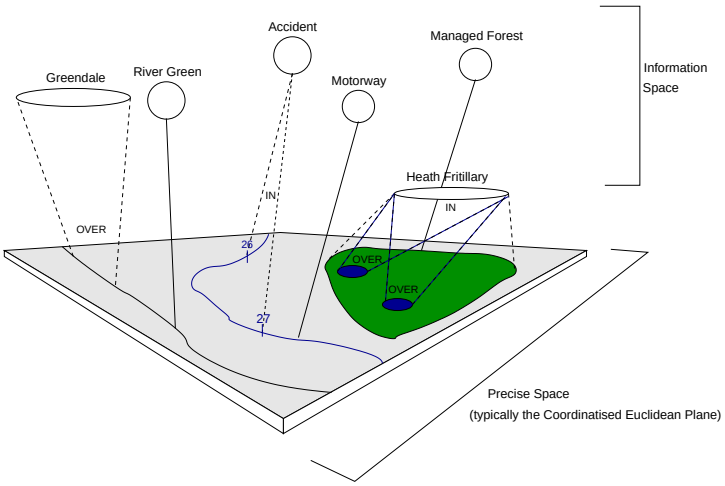

Fig. 2. Locational model using anchoring relations

The other two examples are handled similarly. Greendale is now 'anchored over' a designated stretch of Green River, while the butterfly population is now anchored in the region occupied by the national park and anchored over two specified areas where observations have been made. Each one of the anchorings provides a different piece of locational information, and thus contributes to the spatiality of the spatial object in question.

With our Anchoring approach, we are saying, in short, that there may be many ways in which we can consider objects in information space to be 'spatial'. Exact location, where we identify the location of an object with a precisely specified region, is just one way. Other ways are our two anchoring relations. In short, the job now is to set up a framework in which it is possible to specify a spatial object by a whole bag of different pieces of locational information. It is like describing the location of some object to someone we know. We might say, 'It's definitely within this bit', 'It's twenty miles north of so-and-so', or 'It covers such-and-such'. As we speak the listener is building up some idea of where we are talking about. Our task with Anchoring is to simulate such understanding in our spatial database.

2 Anchorings, Relative Locations and Location Sets

In this section we flesh out the Anchoring approach in terms of a conceptual specification for a new kind of locational component for a spatial data model.

The specification will be independent of any particular data model architecture (hence our use of 'conceptual'), but will nevertheless be geared towards actual implementation in real systems. (Indeed, in section 3 we demonstrate the feasibility of this assertion by implementing the approach as an extension to an object-relational data model.) In what follows, we write

- bold lowercase letters (e.g., $\mathbf{a}$) to denote objects in information space.
- italic uppercase letters (e.g., P) to denote elements of the precise space.

In addition, we let the elements of precise space be points or various kinds of sets of points (regions) which can function as precise spatial locations.

2.1 Location Sets

In the classical model, what makes an object 'spatial' is its being assigned an exact location in a precise space. We would like now to change this definition. An object is now a spatial object if and only if that object has a **location set**. For a given object $\mathbf{o}$ in information space, $loc(\mathbf{o})$ denotes its location set. $\mathbf{o}$ is therefore spatial if and only if $loc(\mathbf{o}) \neq \perp$.[2] A location set embodies all the information about the location of the object that has been entered into the database. Such locational information falls into two sorts: anchorings and relative locations.

2.2 Anchorings

An **anchoring** is an ordered pair (t, X), where X is a region in precise space and t is the type of anchoring (currently restricted to just IN and OVER). The anchorings in the location set $loc(\mathbf{o})$ of some object $\mathbf{o}$ must satisfy the following constraints.

- Any region an object is anchored over is part of any region it is anchored in.

  T1 $\quad (\text{IN}, X) \in loc(\mathbf{o}) \wedge (\text{OVER}, Y) \in loc(\mathbf{o}) \rightarrow Y \subseteq X$

- No two regions that an object is anchored in are disjoint.

  T2 $\quad (\text{IN}, X) \in loc(\mathbf{o}) \wedge (\text{IN}, Y) \in loc(\mathbf{o}) \rightarrow X \cap Y \neq \emptyset$

We are now in a position to describe our accident example in terms of anchorings and location sets. Let $\mathbf{a}$ be the object in information space representing the accident and M the region in precise space representing the location of the stretch of motorway between junctions 26 and 27. We describe our current knowledge about where the accident is in terms of its location set as follows:

$$loc(\mathbf{a}) = \{(\text{IN}, M)\}$$

[2] The location set may be empty, meaning that while the object has a location, we know nothing about it. This case, $loc(o) = \emptyset$, is to be distinguished from the case $loc(o) = \perp$, which means that no location set (not even an empty one) has been assigned.

2.3 Relative Locations

But what if we want in our location set to refer not to regions in precise space, but rather to other objects in information space? The region with which an object is associated may change, and so anchorings which are linked to that region may under particular circumstances also need to change. For example, consider once again our butterfly population **b** and the area of managed forest **f**. Recall that the population is definitely known to inhabit, and so is anchored over, the two regions in which it has been observed (let us call them S_1 and S_2), while it is definitely contained, and thus anchored in, the area of the managed forest. Let F be the region occupied by the managed forest (so **f** is exactly located on F). The location of b described in terms of its location set is therefore

$$loc(\mathbf{b}) = \{(\text{OVER}, S_1), (\text{OVER}, S_2), (\text{IN}, F)\}$$

But what if the region occupied by the managed forest changes? For example, suppose the local authority had dubious respect for the natural environment and allowed the development of a housing estate whose extent infringed directly upon the managed forest. Let F' denote the new region that the managed forest occupies after the development. Since the butterfly location will now also change, we need to update its location set too. After the update the new location of the butterfly population will be

$$loc(\mathbf{b}) = \{(\text{OVER}, S_1), (\text{OVER}, S_2), (\text{IN}, F')\}$$

In high-level representational frameworks of the kind we are describing there is usually no consideration about the practicalities of updating our knowledge. Such 'non-monotonic' aspects, if they are considered at all, are usually part of a dedicated theory. In designing a spatial data model, by contrast, such practicalities are of the utmost importance. So given that the Anchoring approach is intended for real implementation, we introduce now the notion of a **relative location**, which is the second type of locational information a location set may contain. Like an anchoring, a relative location is a pair whose first element is either IN or OVER. But whereas an anchoring is a relation between an object and a region, a relative location is a relation between two objects. In a sense a relative location in a object's location set relates that object to the location of another object *whatever the location happens to be at any given time*.[3] Furthermore, it does this without the introduction of any explicit temporal framework. Thus, we can describe the location of the butterfly population as follows:

$$loc(\mathbf{b}) = \{(\text{OVER}, S_1), (\text{OVER}, S_2), (\text{IN}, \mathbf{f})\}$$

(that is, by a location set comprising two anchorings and one relative location).

[3] A relative location is thus in some ways similar to Donnelly's notion of a 'relative place' [3]

2.4 Reasoning With Location Sets

As presented thus far, our representational framework is of limited use. We can capture location information pertaining to a particular spatial object in a location set, and have that information described in terms of either anchorings or relative locations. Anchorings, through their connection with a precise space, allow for drawing inferences about various regions in precise space. But relative locations allow for no such inference. Our anchoring rules, alluded to in the first section and introduced in our previous paper, provided a mechanism for deriving further information. The task now is to incorporate those rules into our framework.

At this point it is important to emphasise the distinction between a location set and the information which may be inferred from it. A location set, as stated earlier, embodies just the information about the location of the object that has been entered into the database. It is therefore not the same as the information that can be *inferred* from this data. The elements of a location set are the *givens*, the 'information atoms' that the database agent has to work with. Confronted with a query or an assertion, the database agent, like a competent map user, should therefore base its response on an understanding of how the relevant location sets relate the objects in question to each other, to other objects and to the spatial frame in general: in short, it should know what the location sets *mean*. To this end we now give rules specifying the meaning of location sets.

The first two rules may seem obvious, but nonetheless need to be stated. Together the rules state that if a location set contains an in-anchoring or an over-anchoring for a given region, then we can infer that the object is anchored in or anchored over that region, respectively. (As in [9], for the anchored-in relation we write '$\lhd$', and for the anchored-over relation we write '$\rhd$'.)

- If an object's location set contains an in-anchoring for some region, then we can infer that the object is anchored in that region.

  A1 $\qquad (\mathrm{IN}, R) \in loc(\mathbf{o}) \rightarrow \mathbf{o} \lhd R$

- If an object's location set contains an over-anchoring for some region, then we can infer that the object is anchored over that region.

  A2 $\qquad (\mathrm{OVER}, R) \in loc(\mathbf{o}) \rightarrow \mathbf{o} \rhd R$

The importance of these rules is in the use of the conditional, not the biconditional. That is, just because we could infer, in our chain of reasoning, that an object is anchored somehow to some region, we are not thereby licenced to infer that that information is explicitly contained in the location set. A location set is, after all, finite and contains just those 'information atoms' which have been recorded.

The next two rules extend our constraints about what anchorings a location set may or may not contain to more general constraints on the anchoring relation itself.

– Any region an object is anchored over is part of any region it is anchored in.

A3 $$\mathbf{o} \lhd R \wedge \mathbf{o} \rhd S \rightarrow S \subseteq R$$

– No two regions that an object is anchored in are disjoint.

A4 $$\mathbf{o} \lhd R \wedge \mathbf{o} \lhd S \rightarrow S \cap R \neq \emptyset$$

In terms of a logical theory, the constraints T1 and T2 introduced above now emerge as consequences of the axioms A1–A4.

Next, rules A5 and A6 show how existing anchoring information leads to new anchoring information.

– An object anchored in region R is anchored in any region containing R.

A5 $$\mathbf{o} \lhd R \wedge R \subseteq S \rightarrow \mathbf{o} \lhd S$$

– An object anchored over some region R is anchored over any part of R.

A6 $$\mathbf{o} \rhd R \wedge S \subseteq R \rightarrow \mathbf{o} \rhd S$$

Together, rules A3–A6 characterise the primitive anchoring relations.

Notice so far that there has been no mention of relative location. The anchoring relations are 'primitive'—which is to say, their meaning is given not explicitly by definition, but implicitly through their function in the rules as previously stated. By contrast, we now going to explicitly define the relative location notion of 'located within'. We will then see that this leads to some interesting consequences, both practically and philosophically, for the relation and the Anchoring approach in general.

Let $\mathbf{o}$ and $\mathbf{p}$ be objects in information space. We read $\mathbf{o} \sqsubseteq \mathbf{p}$ as '$\mathbf{o}$ is located within $\mathbf{p}$'. The relation $\sqsubseteq$ is defined by the following rule.

– $\mathbf{o}$ is located within $\mathbf{p}$ if and only if there is some region which $\mathbf{o}$ is anchored in and $\mathbf{p}$ is anchored over.

Def$_{\sqsubseteq}$ $$\mathbf{o} \sqsubseteq \mathbf{p} =_{\text{def}} \exists R(\mathbf{o} \lhd R \wedge \mathbf{p} \rhd R)$$

By analogy with A1 and A2, we can state rules for relative location as follows.

A7 $$(\text{IN}, \mathbf{p}) \in loc(\mathbf{o}) \rightarrow \mathbf{o} \sqsubseteq \mathbf{p}$$

A8 $$(\text{OVER}, \mathbf{p}) \in loc(\mathbf{o}) \rightarrow \mathbf{p} \sqsubseteq \mathbf{o}$$

By these axioms we can now derive a series of theorems expressing constraints on what relative locations a location set may or may not contain.

– If $\mathbf{p}$ is located within $\mathbf{o}$, then any region that $\mathbf{p}$ is anchored in must overlap any region that $\mathbf{o}$ is anchored in.

T3 $(\text{IN}, \mathbf{o}) \in loc(\mathbf{p}) \wedge (\text{IN}, R) \in loc(\mathbf{p}) \wedge (\text{IN}, S) \in loc(\mathbf{o}) \rightarrow R \cap S \neq \emptyset$

– If **p** is located within **o**, then any region that **p** is anchored over must be contained within any region that **o** is anchored in.

T4 $(\text{IN}, \mathbf{o}) \in loc(\mathbf{p}) \wedge (\text{OVER}, R) \in loc(\mathbf{p}) \wedge (\text{IN}, S) \in loc(\mathbf{o}) \rightarrow R \subseteq S$

T5 and T6 are similar to T3 and T4, but govern the relative location $(\text{OVER}, \mathbf{o})$, as opposed to $(\text{IN}, \mathbf{o})$.

T5 $(\text{OVER}, \mathbf{o}) \in loc(\mathbf{p}) \wedge (\text{IN}, R) \in loc(\mathbf{o}) \wedge (\text{IN}, S) \in loc(\mathbf{p}) \rightarrow R \cap S \neq \emptyset$

T6 $(\text{OVER}, \mathbf{o}) \in loc(\mathbf{p}) \wedge (\text{OVER}, R) \in loc(\mathbf{o}) \wedge (\text{IN}, S) \in loc(\mathbf{p}) \rightarrow R \subseteq S$

Some interesting further consequences follow from these rules. At first sight, $\sqsubseteq$ may seem to be a good candidate for the merological parthood relation. However, our hopes for having the full power of mereology are dashed as soon as we try to infer the first merological axiom, namely that parthood is reflexive (everything is part of itself). From $\text{Def}_{\sqsubseteq}$, reflexivity of $\sqsubseteq$ requires that, for every object **o**,

$$\exists R(\mathbf{o} \lhd R \wedge \mathbf{o} \rhd R)$$

i.e., every object has some region which it is both anchored in and anchored over. But that implies that all objects must have exact location, which goes counter to the base tenet of the Anchoring approach. Hence making $\sqsubseteq$ reflexive precludes the representation of objects with indeterminate boundary, and so we cannot therefore stipulate such a property. Because it is not reflexive, $\sqsubseteq$ is not 'parthood' in the mereological sense.

The axiom of anti-symmetry also cannot be proved. However, attempting to prove the axiom brings to light some consequences for Anchoring in its relation to ostensibly similar approaches. Recall that parthood anti-symmetry states that, for any two objects, if they are both part of each other, then they are equal. Let **o** and **p** be spatial objects. Suppose **o** is located within **p** and **p** is located within **o**. The definition of $\sqsubseteq$ implies the existence of regions R and S such that **o** is anchored in R and anchored over S, and **p** is anchored in S and anchored over R. By rule A3, it follows that $R \subseteq S$ and $S \subseteq R$, and so $R = S$. So **o** and **p** are each both anchored in and anchored over the same region S. But since that doesn't imply that $\mathbf{o} = \mathbf{p}$, anti-symmetry is not proved. However, from assuming that **o** and **p** are each located within the other, we have inferred that **o** and **p** are exactly located on one (and the same) region. Furthermore, in the case where **o** does equal **p**, we conclude that if **o** is located within itself, then it possesses exact location.

This is an interesting conclusion, since vaguely located objects, like Green Dale, do not have exact location, and so do not have regions which they are simultaneously both anchored in and anchored over. So, from the result in the previous paragraph, we infer that such vaguely located objects cannot be located within themselves. However, far from this being damaging for the Anchoring approach, we are of the opinion that it demonstrates a virtue which is in fact absent from other frameworks.

Consider, for example, supervaluation semantics [6], which was proposed as a way of handling vague predicates in logical models of language. Recently it has been mooted as a possible way of handling indeterminate location in spatial information systems [1]. In supervaluation theory, a property may be ascribed to a vaguely described object just so long as that property is possessed under all possible *precisifications* of that description, that is, all precise descriptions of the object to which the vague description may be regarded as being in some sense an approximation. Given this, note that every precisely described object is indeed located within itself, and so, by the tenets of supervaluation theory, a vaguely described object must be located within itself too. This, however, really only makes sense on the assumption that a vaguely described object is in some way to be equated with the totality of its possible precisifications. This runs directly contrary to the spirit of Anchoring, which prefers to recognise a kind of irreducible vagueness for which the notion of precisification is simply inappropriate.

3 Anchoring in an Object-Relational Database

We return to earth now with an examination of the practicalities of incorporating Anchoring into spatial data models.

The OpenGIS Consortium (OGC)'s Simple Feature Specification for SQL [4] details how their conceptual spatial data model (what they call a 'Simple Features' model) can be implemented into a relational or object-relational database. As we mentioned in the introduction, the Anchoring approach stipulates only the locational component of a spatial data model and, as such, we want it not so much to replace but to extend existing spatial data models. In this section we demonstrate how we could integrate Anchoring into an object-relational data model conforming to the OGC specification.

Currently, with reference to our accident example, a specification for a table maintaining information about accidents may, in accordance with the OGC specification, look something like the following.[4]

```
CREATE TABLE Accidents (
id VarChar(20),
location Point );
```

Here accident location is declared to be of type Point, a subclass of type Geometry. In object-relational terms, Geometry is an example of a 'user-defined type', a custom data type which has an accompanying set of related operations. In declaring a field as Geometry, or one of its subclasses, the user then can make use of the various spatial operations that are provided for Geometry. This raises the level of abstraction in communicating with the database. For example, a

[4] This is not quite true. A real table specification would need to set up the necessary connections between the point object and a spatial reference system (what we call a precise space) which sits in its own row in another table. We have omitted such connections because they are not part of the substance of our argument. But in a real system the connections would need to be added.

query to return the ids of all accidents located within our now familiar stretch of motorway between junctions 26 and 27 of the M5 may be written as

```
SELECT a.id
FROM Accidents a, Motorway-Segments m
WHERE within(a.location, m.location)
AND m.start = 26 AND m.end = 27 AND m.name = 'M5'
```

which, in virtue of the use of the within operation, should be decipherable even for those not familiar with SQL. In implementing Anchoring into the object-relation model, we will make full use of the benefits of user-defined types. Ideally, after the work is complete, the changes required to have accident location described in terms of location sets rather than points will be no more than is shown in

```
CREATE TABLE Accidents (
id VarChar(20),
location LocationSet );
```

Furthermore, since data type operations can be 'overloaded', there will be no need to make changes to queries such as the one above: in the case of our query, for example, it will return the id of any accident located within the stretch of motorway regardless of whether the location of the accident is described by a location set or by a point.

The reminder of this section will go as follows. First, we look at various ways that a location set can be structured as a user-defined type. Secondly, after selecting the most appropriate, we then encode into the user-defined type the constraints T1–T6 that we gave in §2.4. Finally, we extend the spatial operation 'within' so that it works with location sets or a mix of location sets and Geometries.

3.1 Location Set as a User-Defined Type

Described below are three ways of constructing a location set as a user-defined type. Our butterfly population example, because it includes both anchorings and relative locations, is used to illustrate each of the three.

The result of a first attempt to place a location set in an object-relation framework is depicted in Figure 3. As we see, the location of the butterfly, which in a classical model would have had to be a geometric primitive or construct therefrom, is now a location set. The location set is pretty much the same as how we described it in §2.3—which is to say, it contains three anchorings and one relative location. The only difference between how it was then and how it is now is in the reference of the the relative location. In §2.3 the reference was to the actual managed forest object. But now the reference is to the object's location set. The reason for the change is the nature of the object-relational model, specifically the fact that table rows do not have object identifiers and so cannot have object reference. However, having the reference changed in this way is easily handled: where before we read $\mathbf{b} \sqsubseteq \mathbf{f}$ as '$\mathbf{b}$ is located within $\mathbf{f}$', now we

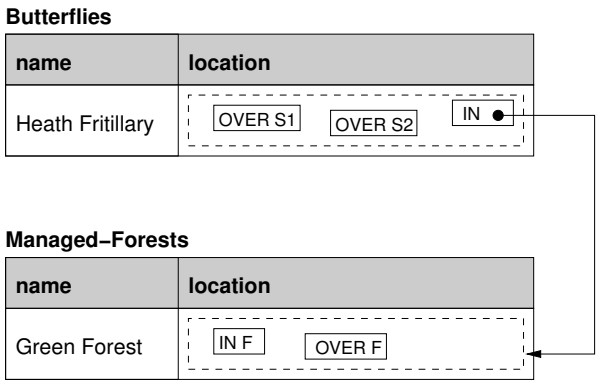

Fig. 3. Location set as user defined type (version 1)

just read it as '**b** is located within the location as described by **f**'s location set'. The change makes absolutely no difference to the workings of the theory.

Location sets are complex. They may contain arbitrary many instances of up to two different types of information and may also have to maintain strings of cross-references to other location sets. Though collection-based user-defined types are possible in an object-relation model (arrays, sets, multisets are all part of the SQL standard), the complexity of a location set may entail that we should maintain its content in separate tables outside of it, as it were. Such a possibility is illustrated in Figure 4. Here anchorings and relative locations are

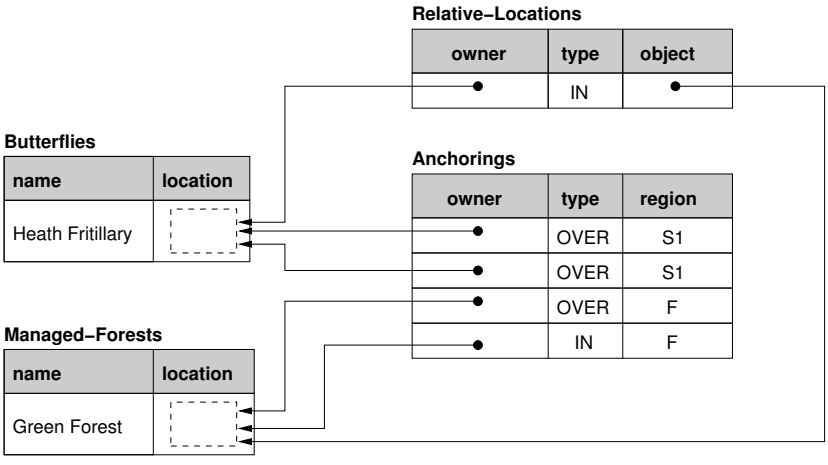

Fig. 4. Location set as user-defined type (version 2)

maintained in the tables Anchorings and Relative-Locations, while location set cross-referencing is maintained through object reference.

Our final implementation of a location set is illustrated in Figure 5. Here object references, which have played an important role thus far, have been removed and replaced with location set identifiers. We do this because object references, though stipulated since SQL 99, have not come into widespread use—e.g, they are still not implemented in the popular open-source database PostgreSQL. This final implementation is what we use below in specifying in SQL the location set constraints and the extended 'within' operation.

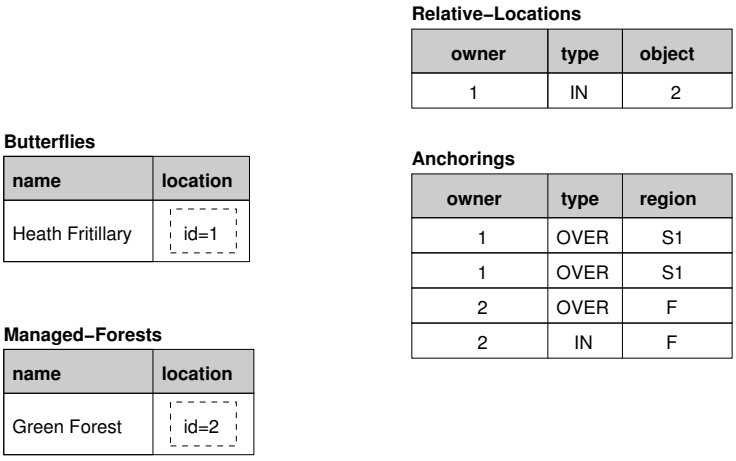

Fig. 5. Location set as user-defined type (version 3)

3.2 Anchoring Constraints

In an object-relational database implementation of Anchoring, the constraints T1–T6 should be implemented at the table level so as to constrain the kinds of relative locations and anchorings that a location set may or may not contain.

First, the definition of the Anchorings table, which maintains the constraints T1 and T2, may look something like the following:

```
CREATE TABLE Anchorings (
owner LocationSetId NOT NULL,
type CHAR NOT NULL CHECK(type IN ('IN', 'OVER')),
region GEOMETRY NOT NULL,

CONSTRAINT T1
CHECK(NOT EXISTS(SELECT x.id
                 FROM Anchorings x, Anchorings y
                 WHERE x.owner = y.owner
```

```
                AND x.type = 'IN' AND y.type = 'OVER'
                AND NOT within(y.region, x.region))),

CONSTRAINT T2
CHECK(NOT EXISTS(SELECT x.id
                FROM Anchorings x, Anchorings y
                WHERE x.owner = y.owner
                AND x.type = 'IN' AND y.type = 'IN'
                AND disjoint(y.region, x.region))),
);
```

while the definition of Relative-Locations (omitting, for the sake of space, T5 and T6) will be

```
CREATE TABLE Relative-Locations (
owner LocationSetId NOT NULL,
type CHAR NOT NULL CHECK(type IN ('IN', 'OVER')),
object LocationSetId NOT NULL,

CONSTRAINT T3
CHECK(NOT EXISTS(SELECT o.id
                FROM Relative-Locations o, Anchorings pa, Anchorings oa
                WHERE o.type = 'IN'
                AND pa.owner = o.object AND pa.type = 'IN'
                AND oa.owner = o.owner AND oa.type = 'IN'
                AND disjoint(oa.region, pa.region))),

CONSTRAINT T4
CHECK(NOT EXISTS(select o.id
                FROM Relative-Locations o, Anchorings, pa, Anchorings oa
                WHERE o.type = 'IN'
                AND pa.owner = o.object AND pa.type = 'IN'
                AND oa.owner = o.owner AND oa.type = 'OVER'
                AND NOT within(oa.region, pa.region)))
);
```

3.3 An Extended 'Within'

Before we leave our discussion of the implementation of anchoring, we show an example of one of the extended spatial operations. The spatial operation we use is 'within', which, under [4], is defined as a two-place boolean operation on Geometry and its various subclasses. The operation returns True if the first Geometry is within the second; otherwise it returns False. We now extend this operation to take either two location sets (if both locations we want to relate are described by location sets) or a location set and a Geometry (if one location is described by a location, but the other, keeping to the classical model, is described by a Geometry).

The extended within taking two location sets is described immediately below. Note that we make use here of the (often confused) fact that a Boolean in SQL

does in fact take one of three, not two, values. These are 1, 0 or NULL. Consequently we can use SQL to form a three-valued logic if we are careful. However, since the interpretation of NULL is non-standard (does it mean 'unknown', 'not applicable', or something else), its use is inevitably problematic, and there are supporters and opponents of its use in this way. Here we do not put ourselves in either camp: we just acknowledge the fact that for a proper implementation of Anchoring, a three-valued logic is required.

```
CREATE FUNCTION within(lsA LocationSet, lsB LocationSet) returns Boolean AS
# Is the location described by lsA within the location described by lsB?
IF EXISTS(SELECT id FROM RelativeLocations
          WHERE type='IN' AND owner=lsB.id AND object=lsA.id) THEN
 RETURN 1; # yes

# Is there a region which the object described by lsA is anchored in
# and the object described by lsB is anchored over
ELSE IF exists(SELECT a.region
          FROM Anchorings a, Anchorings b
          WHERE a.type='IN' AND b.type='OVER'
          AND within(a.region, b.region) THEN
 RETURN 1; # yes

# Are the locations of the objects described by lsA and lsB disjoint or
# partially overlapping. Note: disjoint and poverlap will also have to have
# extended defintions.
ELSE IF disjoint(lsA, lsB) or poverlap(lsA, lsB) THEN
 RETURN 0; # no

ELSE
 RETURN NULL; # maybe
END IF;
```

The second extension of within (which follows immediately below) is here to support backwards compatibility with the classical model. After taking a Geometry and a location set, the function first simply casts the Geometry to a new location set whose contents are an in-anchoring and an over-anchoring on the region described by the Geometry. It then passes the two location sets as arguments to the previous function.

```
CREATE FUNCTION within(ls LocationSet, r Geometry) RETURNS Boolean AS
RETURN within(ls, LocationSet(('IN',r), ('OVER',r)))
```

4 Setting Anchoring in Context

Within the field of GIScience the issue of how to handle uncertainty and vagueness is not new. Fuzzy set theory, supervaluation semantics, rough set theory and logical calculi such as, for example, Cohn and Gotts' Egg Yolk theory [2] have all at various times been put forward as solutions to the problem. So why is Anchoring any different? What does it add that the other formalisms lack?

Our answers to these questions have in some ways already been addressed in our previous paper [9], in which we provided a critical analysis of these frameworks in the light of Anchoring. Though we shall not repeat that analysis here, we should note one common criticism, which can be levelled across all these other frameworks. In these other frameworks, there is too much emphasis on introducing *arbitrary* boundaries out of precisely delineated lines which, were it not for the use of the formalism, would not exist. Why, for instance, is one fuzzy membership function better than another? Who stipulates where the inner or outer parts of the egg yolk or rough set are placed? While there may be methods that provide a reasonable basis for drawing out such information (for example, [5, 7]), it remains true that this information was not present in the original entity prior to its straitjacketing by the formalism. So, what we sought in Anchoring was a means to say just what we know *for certain* about the entity we seek to capture. Having the perspective in mind of the spatial data provider (such as the UK Ordnance Survey), who deals with many different organisations and users, we developed Anchoring accordingly to fit this picture. Other frameworks, we felt, because of their necessary forced precision, are just not appropriate for a spatial data provider who seeks to be authoritative.

But the discussion does not end here. Since writing our previous paper, we have come to the view that Anchoring is *complementary* to the other formalisms we have mentioned, not opposed to them. That this is the case comes, we think, from identifying two components of a spatial information system: (1) the database management system (DBMS) and (2) the modelling system. The DBMS sits at the heart of any spatial information system. It is here that a spatial data 'feed', such as Ordnance Survey's MasterMap, enters.[5] The various other applications which form part of a spatial information system depend on the DBMS for their data, posing queries and making assertions to it. The DBMS is a *repository* of information. The DBMS needs to store facts about spatial location, but at the moment all it has is the classical model. Replacing this with Anchoring provides more flexibility in the definition of spatiality.

Existing formalisms for handling vagueness and uncertainty operate, by contrast to Anchoring, in the modelling system. The modelling system is used to predict the future, explain the past, or to explore 'what if' scenarios [8]. The modelling system draws on the data in the DBMS and, say, after running experiments, updates that data too. In the modelling system, inference going beyond what we only know for certain is acceptable (and often necessary). Hence the usefulness of these other formalisms we have mentioned. But it follows that the DBMS would then need to have a different kind of locational model to accommodate this representation of vagueness and uncertainty. Perhaps this could have been provided by implementing any one of the other formalisms within the DBMS. However, having the perspective of a data provider in mind has alerted us to the full implications of approximating unnecessarily. A data provider such as Ordnance Survey simply must have the means to represent vagueness appro-

[5] `www.ordnancesurvey.co.uk/mastermap`

priately for its needs; to state only *what is known for certain*. Anchoring provides this.

5 Summary

In this paper we presented Anchoring on a number of different levels relating to its representation within a spatial information system. First, there was the conceptual idea of what anchoring is about in general. This is perhaps best seen in the pictures featuring the precise space and information space. Though not technical in content, the pictures give us a view of a spatial data model that highlights some major shortcomings of how location is handled in the classical model. Second, with the introduction of location sets, we showed how the locational element of the data model can be described by a series of *loc* equations: each one describing the location of an object in terms of the contents of the object's location set. Next, with the addition of the logical theory of §2.4, we saw how the contents of location sets and what information we can infer from that content can rest on a secure, mathematical foundation. Finally, in turning to extending the Simple Feature model of the OGC, we showed that Anchoring is a genuine possibility for real systems.

References

1. Brandon Bennett. What is a forest? On the vagueness of certain geographic concepts. *Topoi*, 20:189–201, 2001.
2. A. G. Cohn and N. M. Gotts. The 'egg-yolk' representation of regions with indeterminate boundaries. In Peter A. Burrough and Andrew U. Frank, editors, *Geographic Objects with Indeterminate Boundaries*, GISDATA 2, pages 171–187. Taylor & Francis, 1996.
3. Maureen Donnelly. Relative places. *Applied Ontology*, 1(1):55–75, 2005.
4. Keith Ryden (editor). OpenGIS implementation specification for geographic information — simple feature access — part 2: SQL option. Technical report, Open Geospatial Consortium Inc, 2005. Available for download at `http://www.opengeospatial.org/specs`.
5. FangjuWang and G. Brent Hall. Fuzzy representation of geographical boundaries in GIS. *International Journal of Geographical Information Systems*, 10(5):573–590, 1996.
6. Kit Fine. Vagueness, truth, and logic. *Synthese*, 30:263–300, 1975.
7. Peter Fisher, Tao Cheng, and Jo Wood. Where is Helvellyn? Multiscale morphometry and the mountains of the English Lake District. *Transactions of the Institute of British Geographers*, (29):106–128, 2004.
8. Antony Galton. Dynamic collectives and their collective dynamics. In A. G. Cohn and D. M. Mark, editors, *Spatial Information Theory*, Lecture Notes in Computer Science, pages 300–315, Berlin, 2005. Springer-Verlag. Proceedings of COSIT 2005.
9. Antony Galton and James Hood. Anchoring: A new approach to handling indeterminate location. In A. G. Cohn and D. M. Mark, editors, *Spatial Information Theory*, Lecture Notes in Computer Science, pages 1–13, Berlin, 2005. Springer-Verlag. Proceedings of COSIT 2005.