QUERYING VAGUE SPATIAL OBJECTS IN DATABASES WITH VASA

Alejandro Pauly and Markus Schneider

Computer and Information Sciences and Engineering Department
University of Florida
Gainesville, FL USA
apauly@cise.ufl.edu, mschneid@cise.ufl.edu

KEY WORDS: spatial vagueness, imprecision, uncertainty, VASA, query

ABSTRACT:

Recent years have been witness to the increasing efforts of scientists to design concepts and implementations that can adequately handle the vagueness and imprecision that is widespread in spatial data. Plenty of spatial entities, especially those that occur naturally such as mountains and biotopes, contain intrinsic vague attributes that make their representation as crisply bounded entities ineffective and far from exact. Other examples of such spatial objects include population density, pollution clouds, oil pouches, and even lakes and rivers whose water levels are not determinate but rather can change depending on the pluvial activity. In the context of spatial databases, retrieval and handling of vague spatial objects through querying is critical in exploiting the functionality of the database. Thus, it is important that query mechanisms are able to handle the vagueness that is included in the data models used to represent the objects that are stored. In this paper we present a *Vague Spatial Algebra* (*VASA*) capable of handling spatial vagueness. VASA is defined on the basis of exact models for crisp spatial objects and includes the definitions of vague spatial data types as well as the operations and predicates needed to effectively manipulate instances of these data types. Using VASA as a basis, we introduce the appropriate concepts for enabling database querying that can help exploit the power of the vague spatial data model.

1 INTRODUCTION

Many man-made spatial objects such as buildings, roads, pipelines and political divisions have a clear boundary and extension. In contrast to these crisp spatial objects, most naturally occurring spatial objects have an inherent property of vagueness or indeterminacy of their extension or even of their existence. Point locations may not be exactly known, paths or trails might fade and become uncertain at intervals. The boundary of regions might not be certainly known or simply not be as sharp as that of a building or a highway. Examples are lakes (or rivers) whose extension (or path) depends on pluvial activity, or the locations of oil fields that in many cases can only be guessed. This inherent uncertainty brings to light the necessity of more adequate models that are able to cope with what we will refer to as *vague spatial objects*.

Existing implementations of Geographic Information Systems (GIS) and Spatial Databases assume that all objects are crisply bounded. With the exception of few domain specific solutions, the problem of dealing with spatial vagueness has no widely accepted practical solution. Instead, different conceptual approaches exist for which researchers have defined formal models that can deal with a closer approximation of reality where not all objects have are crisp. For the treatment of vague spatial objects, our *Vague Spatial Algebra* (VASA), which can be embedded into databases, encompasses data types for *vague points*, *vague lines*, and *vague regions* as well as for all operations and predicates required to appropriately handle objects of these data types. The central goal of the definition of VASA is to leverage existing models for crisp spatial objects, resulting in robust definitions of vague concepts derived from proven crisp concepts.

In order to fully exploit the power of VASA in a database context, users must be able to pose significant queries that will allow retrieval of data that is useful for analysis. In this paper, we provide an overview of VASA and the capabilities it provides for handling vague spatial objects. Based on these capabilities, we describe how users can take full advantage of an implementation of VASA by proposing meaningful queries on vague spatial objects. We use sample scenarios to explain how the queries can be posed with a moderate extension of the standard database query language SQL.

This paper starts in Section 2 by summarizing related work that covers relevant concepts from crisp spatial models as well as other concepts for handling spatial vagueness. In Section 3 we introduce the VASA concepts for data types, operations, and predicates. Section 4 shows how a simple extension to SQL will be of great benefit when querying vague spatial data. Finally, in Section 5 we derive conclusions and expose future work.

2 RELATED WORK

In Section 2.1 we provide an overview of the spatial concepts that are necessary for later sections. Section 2.2 details the main features of current approaches for dealing with spatial vagueness.

2.1 Crisp Spatial Concepts

Central to spatial data handling are the data types *point*, *line*, and *region*. In their simple version, *point* objects are defined as single points represented by a pair of coordinates. Simple *line* objects are connected and non self-intersecting curves. Objects of type *region* as made up of a single areal component topologically equivalent to a closed disk. Stronger application requirements and failure to fulfill closure properties under the basic spatial set operations resulted in the development of *complex* spatial data types. Complex *point* objects are represented by a collection of single points, each represented by a pair of coordinates. A complex *line* object is made up of several disconnected block components, each equivalent to a simple *line*. The concept of complex *regions* allows for areal objects composed of several disconnected areas called faces, each possibly with holes. Figure 1 illustrates simple and complex points, lines, and regions.

Topological relationships between spatial objects have been the focus of interdisciplinary research and have also been widely studied in the literature related to spatial databases. Relationships of

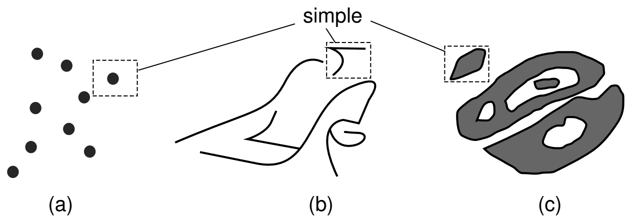

Figure 1: Example complex point (a), complex line (b), and complex region (c). The components inside the dashed squares represent simple versions of each type.

this type are purely qualitative and describe the relative position of spatial objects towards each other. Topological relationships remain unchanged under continuous transformations like translation, rotation and scaling. Such relationships are provided as so-called topological predicates that are commonly used as part of querying systems to test whether a topological relationship holds between a given pair of spatial objects. We focus on the topological predicates defined through the *9-intersection model*. This model has been defined for simple region objects in (Egenhofer, 1989) and later extended to simple regions with holes in (Egenhofer et al., 1994). The complete set of topological relationships for all type combinations of complex spatial objects is defined in (Schneider and Behr, 2006) on the basis of the 9-intersection model.

The 9-intersection model characterizes a topological relationship between two spatial objects A and B by exploring the emptiness of the point sets resulting from the 9 intersections between all the combinations of the *interior* ($^\circ$), *boundary* (∂) and *exterior* ($^-$) from both objects. Each topological relationship can be represented by a *9-intersection matrix* whose values are determined as shown here:

$$\begin{pmatrix} A^\circ \cap B^\circ \neq \emptyset & A^\circ \cap \partial B \neq \emptyset & A^\circ \cap B^- \neq \emptyset \\ \partial A \cap B^\circ \neq \emptyset & \partial A \cap \partial B \neq \emptyset & \partial A \cap B^- \neq \emptyset \\ A^- \cap B^\circ \neq \emptyset & A^- \cap \partial B \neq \emptyset & A^- \cap B^- \neq \emptyset \end{pmatrix}$$

Each valid 9-intersection matrix uniquely represents a single topological relationship. Following this model, the resulting relationships turn out to be mutually exclusive (i.e. one and only one topological relationship holds between each pair of spatial objects). The eight well known and commonly used predicates originally defined for simple regions include *disjoint, overlap, meet, equal, cover, contain, inside* and *coveredBy*. The number of predicates defined in (Schneider and Behr, 2006) for all complex type combinations is larger (e.g., 33 between two complex regions, 82 between two complex lines). Hence, naming of each predicate is not considered a good option. Instead, an alternative method of clustering is used to provide an identifiable set of predicates.

2.2 Approaches for Handling Spatial Vagueness

In this section, we introduce the four main categories of approaches for dealing with spatial vagueness and imprecision. The approaches in these categories can be distinguished due to their ability to deal with mainly two kinds of vagueness. The first kind is inherent to the object and is commonly denoted as *fuzziness*. The second is denoted as *uncertainty* and it commonly appears due to the impossibility to determine values to represent the features of an object (Schneider, 1999). Uncertainty can appear indirectly due to *imprecision* and even due to *inaccuracy* of the collected data.

2.3 Extension of Exact Models into Three-Valued Logics

The most popular approaches to handling spatial vagueness that utilize existing exact models for crisp spatial objects are the *broad boundaries* approach (Clementini and Felice, 1996), the *egg-yolk* approach (Cohn and Gotts, 1996), and the *vague regions* concept (Erwig and Schneider, 1997). These models extend the common assumption that boundaries of regions divide the plane into two sets (the set that belongs to the region, and the set that does not) with the notion of an intermediate set that is not known to certainly belong or not to the region. Thus we say that these models extend crisp models that operate on the Boolean logic (*true*, *false*) into models that handle uncertainty with a three-valued logic (*true*, *false*, *maybe*).

The regions with broad boundaries approach (Clementini and Di Felice, 2001) leverages complex regions (i.e., composed of several components, possibly with holes) to enable the handling of spatial uncertainty. A region with a broad boundary is initially defined as a complex region whose components are enclosed in a zone considered its broad boundary. This zone is bounded by two sharp line boundaries, where the inner line expresses the minimal extension of the component, and the outer line expresses its maximal extension. The special case where both boundaries are equal, represents a crisp component.

Similar to the broad boundary approach (see Figure 2(a)), the egg-yolk model introduced in (Cohn and Gotts, 1996) leverages the *region connection calculus* (RCC) method for topological relationships between simple regions. The egg-yolk model effectively models an uncertain region where the yolk certainly belongs to the region, and the zone surrounding the yolk represents the uncertainty of the model. As a result, the egg-yolk approach and the broad boundary approach enable the handling of what we call concentric regions, where the part that certainly belongs to the region is always surrounded by the uncertain part.

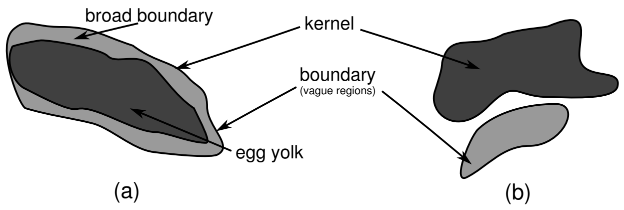

Figure 2: Examples of a broad boundary and egg-yolk region (a), and two vague regions (a,b).

Both the egg-yolk and broad boundary approaches focus on establishing the topological relationships between regions with uncertainty. The broad boundary approach recognizes 44 topological relationships between broad boundary regions based on simple regions, and 56 between those based on complex regions. The egg-yolk recognizes 46 topological relationships between regions with uncertain boundaries.

Finally, the vague regions approach (Erwig and Schneider, 1997) leverages simple regions. A vague region is represented by two simple regions, one denoted as the *kernel*, the other as the *boundary*. The kernel represents the area that certainly belongs the region. The boundary represents the area that may or may not belong to the region (see Figure 2(b)). Vague regions are closed under well defined spatial set operations (i.e., geometric intersection, union, difference, complement). This is not the case in the previous two approaches. The vague regions approach is the precursor to VASA, presented in Section 3.

2.4 Spatial Data Modeling with Rough Sets

Rough set theory was originally introduced by Pawlak in (Pawlak, 1982). A rough set is defined on the basis of a lower approximation and an upper approximation, which are both crisp sets. The lower approximation indicates the elements that certainly belong to the set, whereas the upper approximation also includes those elements that may or may not belong to the set. At the core of rough set theory, *indiscernibility relations* between attributes of elements in the set are used to define the upper and lower approximations. These relations can be used to manipulate the granularity on which the approximations are established.

Worboys in (Worboys, 1998) was one of the first to propose a rough set based alternative for dealing with spatial vagueness. His proposal is geared towards providing a basis for integrating and reasoning about multi-resolution spatial data, that is data that was captured at different resolutions but needs to be handled together.

The authors in (Beaubouef and Petry, 2002) use rough sets as the mathematical foundation to model uncertainty in topological relationships between egg-yolk regions. In comparing the expressiveness of using the RCC model versus rough set theory, some relationships become indistinguishable using rough sets, while others can in fact be specified with higher precision. The authors conclude that rough sets provide a valuable foundation for modeling uncertainty in spatial data and can leverage existing theories such as the RCC theory for topological relationships. Furthermore, rough sets can provide higher granularity than models such as the egg-yolk approach. This is due to the use of the indiscernibility relationship. In contrast, exact model based approaches do not handle such a measure for defining the partition between what is certain and what is not, instead this is assumed to be a feature of the existing data.

Another method for employing rough sets in handling uncertainty is proposed in (Ahlqvist et al., 2000) where the authors pursue the derivation of quality measures for the uncertainty or imprecision in their data. *Rough classification* is used to assign data (areas) to classes that ensure the mutual exclusivity of the lower approximations of objects but allow upper approximations to overlap through classes.

2.5 Fuzzy Set Theoretic Approaches to Spatial Data Handling

Fuzzy set based approaches attempt to model inherently vague spatial objects. These models have the ability to represent *blend-in* type boundaries such as the boundaries of a pollution cloud or the “boundary” between a mountain and a valley.

Much work has focused on the definition of data types designed for the treatment of fuzzy spatial objects. *Fuzzy regions* presented in (Altman, 1994) assign membership to some property for every coordinate point within the (fuzzy) region. A formal definition of *fuzzy points*, *fuzzy lines* and *fuzzy regions* is included in (Schneider, 1999). A recent effort for the definition of a *vague spatial algebra* based on fuzzy sets is presented in (Dilo et al., 2004).

In (Ahlqvist et al., 2003), the authors extend their rough classification concept from (Ahlqvist et al., 2000) to a classification of fuzzy regions. That is, the lower approximation and the upper approximation of a region are each represented by a fuzzy classification. According to the authors, this enables treatment of indiscernibility or imprecision by the rough set classification and fuzziness by the fuzzy classification.

2.6 Probabilistic Methods

Probabilistic approaches for handling spatial vagueness such as that in (Finn, 1993), model spatial objects on the basis of the probability of membership of an entity (i.e., point, area, object) in a set. This results in an *expected* membership which is based on the subjectively defined probability function. This can be contrasted to the membership values of fuzzy sets that are objective in the sense that they can be computed formally or determined empirically.

2.7 Querying with Vagueness

Literature provides some work that has been devoted to solving the problem of querying with vagueness. As discussed in (Lee and Kim, 1993) vagueness does not necessarily only appear in the data being queried, but can also be part of the query itself. Namely, one can pose a query to retrieve all ”small cars”. While the car data is crisp and there is no vagueness on the data, the query itself is vague due to the semantics of the term ”small” which is an obvious example of a vague concept.

Work that focuses on querying vague data has largely sided with fuzzy representations of data. In a spatial context, (Schneider, 2001) proposes classifications of membership values in order to group sets of values together. For example, a classification could assign the term ”mostly” to high membership values (e.g., 0.95-0.98). Based on this classification queries can ask for ”mostly” disjoint objects when using fuzzy topological predicates thus giving a degree of disjointness on the basis of the membership values. Most fuzzy oriented querying propositions have to deal with classifying the membership values, thus they prove ineffective for our purposes.

In the context of databases in general, the approaches in (Ichikawa and Hirakawa, 1986, Motro, 1988, Palkoska and Kung, 1997, Kung and Palkoska, 1998) all propose extensions to query languages on the basis of an operator that enables vague results under different circumstances. For example, in (Ichikawa and Hirakawa, 1986) the operator `similar-to` for *QBE* (Query-by-Example) is proposed alongside relational extensions so that related results can be provided in the event where no exact results match a query. In (Motro, 1988) the operator $\sim$ is used in a similar way to the `similar-to` operator. All these approaches require additional information to be stored as extra relations and functions about distance that allow the query processor to compute close enough results. Although some of these approaches are extended to deal with fuzzy data, the general idea promotes the execution of vague queries over crisp data.

3 VASA

In this section we describe the concepts that compose our Vague Spatial Algebra. The foundation of VASA are its data types which we specify in Section 3.1. Spatial set operations and metric operations are introduced in Section 3.2. Finally, the concept of vague topological predicates is briefly introduced in Section 3.3.

3.1 Vague Spatial Data Types

An important goal of VASA (and of all approaches to handling spatial uncertainty that are based on exact models) is to leverage existing definitions of crisp spatial concepts. In VASA, we enable a generic vague spatial type constructor v that, when applied to any crisp spatial data type (i.e., *point*, *line*, *region*), renders a formal syntactic definition of its corresponding vague spatial data type. For any crisp spatial object x, we define its composition from three disjoint point sets, namely the interior (x°), the boundary (∂x) that surrounds the interior, and the exterior (x^-) (Schneider and Behr, 2006). We also assume a definition of the geometric set operations union ($\oplus$), intersection ($\otimes$), difference ($\ominus$), and complement($\boxminus$) between crisp spatial objects such as that from (Güting and Schneider, 1995).

Definition 1 Let $\alpha \in \{point, line, region\}$. A *vague spatial data type* is given by a type constructor v as a pair of equal crisp spatial data types α, i.e.,

$$v(\alpha) \quad = \quad \alpha \times \alpha$$

such that for $w = (w_k, w_c) \in v(\alpha)$, holds:

$$w_k^\circ \cap w_c^\circ = \varnothing$$

We call $w \in v(\alpha)$ a (two-dimensional) *vague spatial object* with *kernel part* w_k and *conjecture part* w_c. Further, we call $w_o := \boxminus(w_k \oplus w_c)$ the *outside part* of w. For $\alpha = point$, $v(point)$ is called a *vague point* object and denoted as *vpoint*. Correspondingly, for *line* and *region* we define $v(line)$ resulting in *vline* and $v(region)$ resulting in *vregion*.

Syntactically, a vague spatial object is represented by a pair of crisp spatial objects of the same type. Semantically, the first object denotes the kernel part that represents what certainly belongs to the object. The second object denotes the conjecture part that represents what is not certain to belong to the object. We require both underlying crisp objects to be disjoint from each other. More specifically, the constraint described above requires the interiors of the kernel part and the conjecture part to not intersect each other. Figure 3 illustrates instances of a vague point, a vague line, and a vague region as objects of the data types defined above.

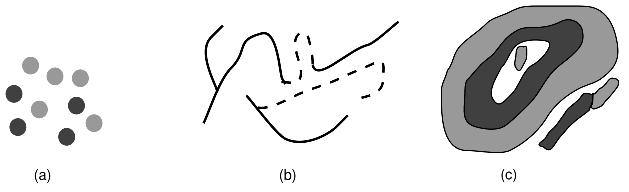

Figure 3: A vague point object (a), a vague line (b) and a vague region (c). Kernel parts are symbolized by dark gray points, straight lines, and dark gray areas. Conjecture parts are symbolized by light gray point, dashed lines, and light gray areas.

3.2 Vague Spatial Operations

We are interested in the definition of the vague spatial set operations that compute the *union*, *intersection*, and *difference* between two vague spatial objects. Following from the generic definition of vague spatial data types, we leverage crisp spatial set operations to reach a generic definition of vague spatial set operations.

We define the syntax of function $h \in \{$**intersection**, **union**, **difference**$\}$ as $h : v(\alpha) \times v(\alpha) \to v(\alpha)$. The complement operation is defined as **complement** : $v(\alpha) \to v(\alpha)$. Semantically, their generic (type independent) definition is reached by considering the individual relationships between kernel parts, conjecture parts, and the outside part (i.e., everything that is not kernel part or conjecture part) of the vague spatial objects involved in the operations. The result of each operation is computed using one of the tables in Table 1. For each operation the rows denote the parts of one object and the columns the parts of another, and we label them *k*, *c*, and *o* to denote the kernel part, conjecture part, and outside part respectively. Each entry of the table denotes the intersection of kernel parts, conjecture parts, and outside parts of both objects, and the label in each entry specifies whether the corresponding intersection belongs to the kernel part, conjecture part, or outside part of the operation's result object.

union	k	c	o
k	k	k	k
c	k	c	c
o	k	c	o

intersection	k	c	o
k	k	c	o
c	c	c	o
o	o	o	o

difference	k	c	o
k	o	c	k
c	o	c	c
o	o	o	o

complement	k	c	o
	o	c	k

Table 1: Components resulting from intersecting kernel parts, conjecture parts, and outside parts of two vague spatial objects with each other.

Each table from Table 1 can be used to generate an executable specification of the given crisp spatial operations. For each table, the specification will operate on the kernel parts and conjecture parts to result in a definition of its corresponding vague spatial operation. Following are such definitions as executable specifications of geometric set operations over crisp spatial objects:

Definition 2 Let $u, w \in v(\alpha)$, and let u_k and w_k denote their kernel parts and u_c and w_c their conjecture parts. We define:

$$\begin{array}{rll}
\text{(i)} & u\ \textbf{union}\ w & := (u_k \oplus w_k, (u_c \oplus w_c) \ominus \\
& & \quad (u_k \oplus w_k)) \\
\text{(ii)} & u\ \textbf{intersection}\ w & := (u_k \otimes w_k, (u_c \otimes w_c) \oplus \\
& & \quad (u_k \otimes w_c) \oplus (u_c \otimes w_k)) \\
\text{(iii)} & u\ \textbf{difference}\ w & := (u_k \otimes (\boxminus(w_k \oplus w_c)), (u_c \otimes w_c) \oplus \\
& & \quad (u_k \otimes w_c) \oplus (u_c \otimes (\boxminus(w_k \oplus w_c)))) \\
\text{(iv)} & \textbf{complement}\ u & := (\boxminus(u_k \oplus u_c), u_c)
\end{array}$$

We can formally prove the correspondence between executable specifications and the tables in Table 1. For readability, we introduce juxtaposition as an abbreviating notation for the intersection of two crisp spatial objects and note that intersection has a higher associativity than union and difference. To prove Definition 2(ii) where $z = u \otimes w$ we first note that for $p, q \in \alpha$ that identity $p \oplus q = pq \oplus p(\boxminus q) \oplus (\boxminus p)q$ holds. Thus, we can rewrite the conjecture part of the definition as $(u_c w_c \oplus u_c(\boxminus w_c) \oplus (\boxminus u_c)w_c) \ominus (u_k w_k \oplus u_k(\boxminus w_k) \oplus (\boxminus u_k)w_k)$. Considering that $\boxminus w_c = w_k \oplus w_o$ we derive: $(u_c w_c \oplus u_c(w_k \oplus w_o) \oplus (u_k \oplus u_o)w_c) \ominus (u_k w_k \oplus u_k(w_c \oplus w_o) \oplus (u_c \oplus u_o)w_k)$. Applying distributivity of $\otimes$ results in $(u_c w_c \oplus u_c w_k \oplus u_c w_o \oplus u_k w_c \oplus u_o w_c) \ominus (u_k w_k \oplus u_k w_c \oplus u_k w_o \oplus u_c w_k \oplus u_o w_k)$. In this term, only $u_c w_k$ and $u_k w_c$ appear in both parts of the difference; all other intersections to be subtracted have no effect at all since all intersections are pairwise disjoint due to the definition of vague spatial objects. We obtain $u_c w_c \oplus u_c w_o \oplus u_o w_c$, which corresponds to the condition required for k_c. Due to space constraints we refer the reader to (Pauly and Schneider, 2004) where the rest of the proofs can be found.

3.3 Vague Topological Predicates

For the definition of topological predicates between vague spatial objects (*vague topological predicates*), it is our goal to continue leveraging existing definitions of crisp spatial concepts, in this case topological predicates between crisp spatial objects. Topological predicates are used to describe purely qualitative relationships such as *overlap* and *disjoint* that describe the relative position between two objects and are preserved under continuous transformations.

For two vague spatial objects $A \in v(\alpha)$, and $B \in v(\beta)$ and the set $T_{\alpha\beta}$ of all crisp topological predicates between objects of types α and β (Schneider and Behr, 2006), the topological relationship between A and B is determined by the 4-tuple of crisp topological relationships (p, q, r, s) such that $p, q, r, s \in T_{\alpha\beta}$ and:

$$\begin{aligned}&p(A_k, B_k) \ \wedge \ q(A_k \oplus A_c, B_k) \ \wedge \\ &r(A_k, B_k \oplus B_c) \ \wedge \ s(A_k \oplus A_c, B_k \oplus B_c)\end{aligned}$$

We define the set $V_{\alpha\beta}$ of all vague topological predicates between objects of types $v(\alpha)$ and $v(\beta)$. Due to inconsistencies that can exist between elements within each tuple, not all possible combinations result in 4-tuples that represent valid vague topological predicates in the set $V_{\alpha\beta}$. An example is the 4-tuple: $(\mathit{overlap}(A_k, B_k), \mathit{disjoint}(A_k, B_k \oplus B_c), \mathit{disjoint}(A_k \oplus A_c, B_k), \mathit{disjoint}(A_k \oplus A_c, B_k \oplus B_c))$. In this example $\mathit{overlap}(A_k, B_k) \Rightarrow A_k^\circ \cap B_k^\circ \neq \emptyset$ and $\mathit{disjoint}(A_k, B_k \oplus B_c) \Rightarrow A_k^\circ \cap (B_k \oplus B_c)^\circ = \emptyset$. These two implications clearly contradict one another because according to the definition of the spatial union operation it holds that $B_k^\circ \subseteq (B_k \oplus B_c)^\circ$ and by the transitivity of set containment it is implied that $A_k^\circ \cap (B_k \oplus B_c)^\circ \neq \emptyset$. This directly contradicts $\mathit{disjoint}(A_k, B_k \oplus B_c)$.

In (Pauly and Schneider, 2006), we present a method for identifying the complete set of vague topological predicates. At the heart of the method, each 4-tuple is modeled as a *binary spatial constraint network* (BSCN). The modeling is done by considering A_k, B_k, $A_k \oplus A_c$, and $B_k \oplus B_c$ as variables in the network, and p, q, r, and s as constraints over the variables. The implicit containment relationship between A_k and $A_k \oplus A_c$, and between B_k and $B_k \oplus B_c$ are also modeled as constraints. Each BSCN is tested for *path-consistency* which is used to check, via constraint propagation, that all original constraints are consistent.

For each type combination of *vpoint*, *vline*, and *vregion*, possibly thousands of predicates are recognized (see Table 2). As a result a further step of predicate clustering is needed to make these predicates accessible for querying.

	vregion	*vline*	*vline*
vpoint	51	974	166
vline	974	471650	74916
vregion	166	74916	55880

Table 2: Number of identified 4-tuples on the basis of complex spatial data types.

Sets of 4-tuples are created into clustered vague topological predicates. Clusters can be defined by the user who specifies three rules for each cluster; one rule is used to determine whether the clustered predicate certainly holds between the objects, the second to determine if the cluster certainly does not hold, and the third to determine when the cluster *maybe* holds, but it is not possible to give a definite answer. This effectively symbolizes the three-valued logic that is central to our definition of vague spatial data types.

The specification of rules for each cluster can be performed at two different levels; the first is the the crisp topological predicate level, where the criteria is specified on the basis of the values of p, q, r, s. For example, if p, q, r, s represent crisp topological predicates between simple spatial objects, we can create a cluster labeled $\mathit{Disjoint}$ which simulates the semantics of the underlying crisp topological predicate *disjoint*. Such a cluster can be represented by the following three rules:

$$\begin{aligned}
\mathit{Disjoint}(A,B) = \mathit{true} \quad & \Leftrightarrow \mathit{disjoint}(A_k \oplus A_c, B_k \oplus B_c) \\
& \Leftrightarrow s = \mathit{disjoint} \\
\mathit{Disjoint}(A,B) = \mathit{false} \quad & \Leftrightarrow \neg \mathit{disjoint}((A_k), (B_k)) \\
& \Leftrightarrow p \neq \mathit{disjoint} \\
\mathit{Disjoint}(A,B) = \mathit{maybe} \ & \Leftrightarrow \neg(\mathit{Disjoint}(A,B) = \mathit{true} \vee \\
& \mathit{Disjoint}(A,B) = \mathit{false}) \\
& \Leftrightarrow \neg(s = \mathit{disjoint} \ \vee \ p \neq \mathit{disjoint})
\end{aligned}$$

Clustering rules can also be specified at the point set level. That is, the rules are defined on the basis of the emptiness (or non-emptiness) of the intersection between all combinations of interior ($^\circ$), boundary (∂) and exterior ($^-$) of the crisp spatial objects that make up the vague spatial objects for which the cluster is defined. This type of specification allows for more general rules that are independent of the set of topological predicates that operate on the underlying crisp spatial objects. For example, the rules for the cluster $\mathit{Disjoint}$ can be specified more generally as:

$$\begin{aligned}
\mathit{Disjoint}(A,B) = \mathit{true} \quad & \Leftrightarrow ((A_k \oplus A_c)^\circ \cap (B_k \oplus B_c)^\circ = \varnothing) \\
& \wedge ((A_k \oplus A_c)^\circ \cap \partial(B_k \oplus B_c) = \varnothing) \\
& \wedge (\partial(A_k \oplus A_c) \cap (B_k \oplus B_c)^\circ = \varnothing) \\
& \wedge (\partial(A_k \oplus A_c) \cap \partial(B_k \oplus B_c) = \varnothing) \\
\mathit{Disjoint}(A,B) = \mathit{false} \quad & \Leftrightarrow ((A_k)^\circ \cap (B_k)^\circ \neq \varnothing) \vee \\
& ((A_k)^\circ \cap \partial(B_k) \neq \varnothing) \vee \\
& (\partial(A_k) \cap (B_k)^\circ \neq \varnothing) \vee \\
& (\partial(A_k) \cap \partial(B_k) \neq \varnothing) \\
\mathit{Disjoint}(A,B) = \mathit{maybe} \ & \Leftrightarrow \neg(\mathit{Disjoint}(A,B) = \mathit{true} \vee \\
& \mathit{Disjoint}(A,B) = \mathit{false})
\end{aligned}$$

This rule implies that the vague spatial objects A and B are truly disjoint if none of their components have intersections of interiors or boundaries between each other.

4 QUERYING WITH VASA

Now that all the VASA concepts are in place, we are interested in showing the practical use of VASA objects and their operations in database queries. We propose two ways of enabling VASA within a database query language; the first as presented in Section 4.1 works by adapting VASA to partially work with SQL, currently the most popular database query language. The second proposal presented in Section 4.2 extends SQL so that it has the necessary operators to handle vague queries.

4.1 Crisp Queries of Vague Spatial Data

One of the advantages of being able to use VASA in conjunction with popular DBMSs is the availability of a database query language such as SQL. We focus on querying with SQL as it represents the most popular and widely available database query language. SQL queries can be used to retrieve data based on the evaluation of boolean expressions. This obviously represents a problem when dealing with vague spatial objects because their vague topological predicates are based on a three-valued logic. On the other hand the current definitions of numeric vague spatial operations do not suffer from this issue because the operations return crisp values that are later interpreted by the user (e.g., the user posing a query must know that **min-length** returns the length associated with the kernel part of a vague line object). Thus, these concepts are already adapted to provide crisp results of vague data.

In the case of vague topological predicates, the first step in order to allow querying of vague spatial objects through SQL is to adapt the results of the predicates to a form understandable by the query language. The adaptation of the three-valued vague topological predicates to boolean predicates can be done with the following six transformation predicates that are defined for each vague topological predicate P that can operate over vague spatial objects A and B:

$$\begin{aligned}
True\_P(A,B) = \mathit{true} &\Rightarrow P(A,B) = \mathit{true} \\
True\_P(A,B) = \mathit{false} &\Rightarrow P(A,B) = \mathit{maybe} \vee \\
& P(A,B) = \mathit{false} \\
Maybe\_P(A,B) = \mathit{true} &\Rightarrow P(A,B) = \mathit{maybe} \\
Maybe\_P(A,B) = \mathit{false} &\Rightarrow P(A,B) = \mathit{true} \vee \\
& P(A,B) = \mathit{false} \\
False\_P(A,B) = \mathit{true} &\Rightarrow P(A,B) = \mathit{false} \\
False\_P(A,B) = \mathit{false} &\Rightarrow P(A,B) = \mathit{true} \vee \\
& P(A,B) = \mathit{maybe}
\end{aligned}$$

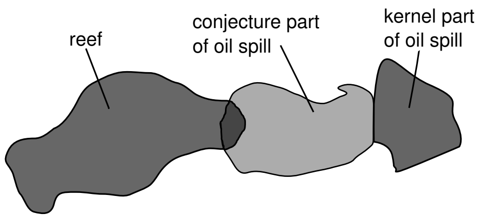

Figure 4: A representation of an ecological scenario using vague regions.

With this transformation in place, queries operating on vague spatial objects can include references to vague topological predicates and vague spatial operations. For example, for the purpose of storing scenarios such as that in Figure 4, assume that we have a table $spills(id : INT,\ name : STRING,\ area : VREGION)$ where the column representing oil spills is denoted by a vague region where the conjecture part represents the area where the spill may extend to. We also have a table $reefs(id : INT,\ name : STRING,\ area : VREGION)$ with a column representing coral reefs as vague regions. We can pose an SQL query to retrieve all coral reefs that are in any danger of contamination from an oil spill. We must find all reefs that are not certainly *Disjoint* from the *Exxon-Valdez* oil spill.

```
SELECT r.name
FROM   reefs r, spills s
WHERE  s.name = "Exxon-Valdez"
and    NOT True_Disjoint(r.area,s.area);
```

Vague topological predicates can also be used to optimize query performance. Assume that, as illustrated in Figure 5, we have data of terrorists routes represented by vague lines in the table $terrorists(id : INT,\ name : STRING,\ route : VLINE)$. We want to retrieve the minimum length of the intersections of all pairs of intersecting routes of terrorists. To do so, we choose to compute the intersection of only those pairs that are certainly not *Disjoint* and neglect the computation of the intersection of those pairs that have been determined to not certainly intersect.

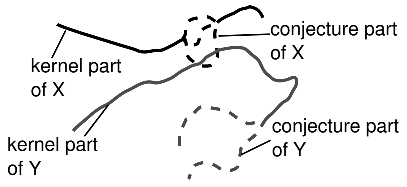

Figure 5: Scenario illustrating the use of vague lines to represent routes of suspected terrorists X and Y.

```
SELECT a.name, b.name,
       min-length(
        intersection(a.route,b.route))
FROM   terrorists a, terrorists b
WHERE  False_Disjoint(a.route,b.route);
```

Other queries can include retrieval based not only on spatial data but also based on common type data (i.e., numbers, characters) stored alongside the spatial objects. Being able to relate both data domains (spatial and non-spatial) in queries is one of the main advantages of providing VASA as an algebra that can extend current DBMSs which are well-proven to provide the necessary services for dealing with data of common types. We can provide such queries based on Figure 6 where the data can be stored in table $animals(id : INT,\ name : STRING,\ roam\_area : VREGION,\ mig\_route : VLINE,\ drink\_spot : VPOINT)$.

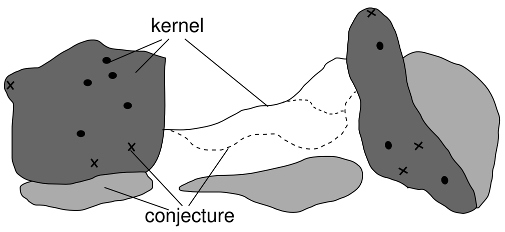

Figure 6: The vague spatial object representation of an animal's roaming areas, migration routes and drinking spots.

For example, we wish to retrieve all species of animals whose average weight is under 40 lbs., their last count was under 100 and may have roaming areas completely contained within the roaming areas of carnivore animals whose average weight is above 80 lbs. This information might be useful to recognize animal species with low counts that could be extinct due to living amongst larger predators. The extinction of the smaller species can be catastrophic even for the larger species that depends on the smaller for nutrition. This retrieval uses data elements that are both spatial and non-spatial:

```
SELECT s.name,
FROM   animals s, animals l
WHERE  s.avgsize<40
and    l.avgsize>80
and    s.count<100
and    NOT False_Inside(
       s.roam_area,l.roam_area);
```

Queries can also be posed to test elements from within single tuples in the database. For example, we would like to retrieve all animal species that do not have drinking spots that are certainly lying inside their roaming areas. For any of these species environmentalists must create artificial drinking spots where the animals can hydrate.

```
SELECT s.name,
FROM   animals s
WHERE  NOT False_Disjoint(
       s.drink_spot,s.roam_area);
```

4.2 A Vague Query Language Extension to Enable Vague Queries on Vague Spatial Data

We analyze the approaches introduced in Section 2.7 and notice that, in the context of VASA, we are not trying to solve the problem of dealing with vague queries, but we need to query vague data. Thus, we propose to extend a common query language such as SQL with the operator $\sim$. However, our data itself is vague thus we do not need the extra relations and functions of distance required by previous approaches. As a result, the semantics of the operator $\sim$ is not the same as in the existing literature where it allows for vague queries to be executed on crisp data. Instead, we will allow for the execution of vague queries over vague data.

Boolean predicates in SQL and in fact in many programming languages implicitly assume truth values unless otherwise noted, which is commonly done with the negation operators `NOT` or `!`. We propose $\sim$ to operate syntactically similar to `!` but semantically, instead of negating the result, it opens the possibility for uncertain results. For example, let us assume we have the table $tempzones(id : INT,\ name : STRING,\ shape : VREGION)$ that contains information about different temperature zones, including their representation as vague regions in the column named `shape`. We pose the following query:

```
SELECT a.name, b.name
FROM   tempzones a, tempzones b
WHERE  Overlap(a.shape,b.shape);
```

This query will return only those regions that certainly overlap. But instead we want to include in the result, all those that might overlap as well, we pose the query again as:

```
SELECT a.name, b.name
FROM   tempzones a, tempzones b
WHERE  ~Overlap(a.shape,b.shape);
```

In this case, the interpretation of $\sim$ should allow the retrieval of all temperatures that may or may not overlap in addition to those that definitely overlap. For the use of numeric values in queries, the query processor should be able to handle number ranges as an atomic data type such that we can combine the minimum and maximum area operations on vague regions into one operator and pose the following query:

```
SELECT a.name
FROM   tempzones a
WHERE  a.shape.area()~300;
```

That is, the result of this query will include all temperature zones whose area range includes 300. The inclusion of this operator and the management of number ranges those not preclude the use of exact operators that would allow to deal with crisp spatial regions. Because crisp spatial objects represent simply a specific instance of vague spatial data types, a query such as the following can still be executed with the result set including all those temperature zones that were modeled as vague regions with no conjecture, thus representing crisp regions:

```
SELECT a.name
FROM   tempzones a
WHERE  a.shape.area()=300;
```

This extension of course, would require actual reimplementation of the query language within the DBMS in order to enable handling of numeric ranges and three-valued logic operations.

5 CONCLUSIONS AND FUTURE WORK

The conceptual design of VASA which we have presented in this paper, shows the clear goal of leveraging existing crisp concepts. There is more than one reason behind this goal. The first reason is to take advantage of existing robust concepts for handling crisp spatial objects. Second, at the conceptual level, the correctness of the definitions for vague concepts largely rests on the correctness of the defined crisp concepts; thus, we reduce the chance of errors in our definitions. As an example, see Definition 2 where vague spatial operations are defined as an executable specification on the basis of crisp spatial operations. Third, the executable specification translates easily to the implementation level. Having an existing correct implementation of crisp spatial data types, their operations, and predicates, we can implement VASA by instantiating existing crisp spatial data types and executing operations on them.

In Section 2, we mentioned current approaches to handling spatial vagueness and imprecision. VASA's concepts feed from all these and thrive in providing a complete type system that includes a systematic approach to vague spatial operations, and most importantly to vague topological predicates. The main advantages of VASA include conceptual simplicity, robustness derived from existing robust crisp concepts, and viability of implementation. In contrast, VASA's main disadvantage is its inability to effectively deal with situations that would seems appropriate for fuzzy set based systems. Nonetheless, we believe that future work can be directed towards more general definitions based on exact models that would be more near to the capabilities of fuzzy set based systems but that can take advantage of existing crisp concepts.

Based on these concepts, we have proposed ideas for database querying of objects from VASA. While these ideas are simple, they are able to fully exploit the capabilities of VASA and allow the user to pose significant queries that can handle spatial vagueness. The proposed language extension and transformation mechanisms further reassure the advantages of defining VASA on the basis of existing exact spatial models. These advantages include robustness of formal concepts that can directly transfer into an implementation that also benefits from simplicity.

Other future work related to VASA stems in at least two directions that are worth following; the first involves enabling similar querying ideas to systems that attempt to handle vagueness with a higher precision, such as fuzzy set theory based systems or even systems with finite multi-valued logics (i.e., more than three values). The other direction involves the performance aspect of implementing indexes that can operate on vague spatial objects and whether it is possible to extend current indexing concepts for crisp spatial objects, thus following the general design of VASA.

ACKNOWLEDGEMENTS

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

REFERENCES

Ahlqvist, O., Keukelaar, J. and Oukbir, K., 2000. Rough Classifcation and Accuracy Assessment. Int. Journal of Geographical Information Sciences 14, pp. 475– 496.

Ahlqvist, O., Keukelaar, J. and Oukbir, K., 2003. Rough and Fuzzy Geographical Data Integration. Int. Journal of Geographical Information Sciences 17, pp. 223–234.

Altman, D., 1994. Fuzzy Set Theoretic Approaches for Handling Imprecision in Spatial Analysis. Int. Journal of Geographical Information Systems 8(3), pp. 271–289.

Beaubouef, T. and Petry, F., 2002. A Rough Set Foundation for Spatial Data Mining Involving Vague Regions . In: IEEE Int. Conf. on Fuzzy Systems, IEEE Computer Society, pp. 767 – 772.

Burrough, P. A. and Frank, A. U. (eds), 1996. Geographic Objects with Indeterminate Boundaries. Taylor & Francis.

Clementini, E. and Di Felice, P., 2001. A Spatial Model for Complex Objects with a Broad Boundary Supporting Queries on Uncertain Data. Data & Knowledge Engineering (DKE) pp. 285–305.

Clementini, E. and Felice, P., 1996. An Algebraic Model for Spatial Objects with Indeterminate Boundaries. in (Burrough and Frank, 1996), pp. 153–169.

Cohn, A. G. and Gotts, N. M., 1996. The 'Egg-Yolk' Representation of Regions with Indeterminate Boundaries. in (Burrough and Frank, 1996), pp. 171–187.

Dilo, A., de By, R. and Stein, A., 2004. Definition and Implementation of Vague Objects. In: Int. Symp. on Spatial Data Quality, pp. 139–145.

Egenhofer, M., Clementini, E. and Di Felice, P., 1994. Topological Relations between Regions with Holes. Int. Journal of Geographical Information Systems 8, pp. 128–142.

Egenhofer, M. J., 1989. A Formal Definition of Binary Topological Relationships. In: Int. Conf. on Foundations of Data Organization and Algorithms, Springer-Verlag, pp. 457–472.

Erwig, M. and Schneider, M., 1997. Vague Regions. In: 5thInt. Symp. on Advances in Spatial Databases, Springer-Verlag, pp. 298–320.

Finn, J. T., 1993. Use of the Average Mutual Information Index in Evaluating Classification Error and Consistency. Int. Journal of Geographical Information Systems 7(4), pp. 349–366.

Güting, R. H. and Schneider, M., 1995. Realm-Based Spatial Data Types: The ROSE Algebra. VLDB Journal 4, pp. 100–143.

Ichikawa, T. and Hirakawa, M., 1986. ARES: A Relational Database with the Capability of Performing Flexible Interpretation of Queries. IEEE Trans. on Software Engineering 12, pp. 624–634.

Kung, J. and Palkoska, J., 1998. Vague Joins – An Extension of the Vague Query System VQS. In: 9th Int. Workshop on Database and Expert Systems Applications, pp. 997–1001.

Lee, D. and Kim, M., 1993. Accommodating Subjective Vagueness Through a Fuzzy Extension to the Relational Data Model. Information Systems 18, pp. 363–374.

Motro, A., 1988. VAGUE: A User Interface to Relational Databases that Permits Vague Queries. ACM Trans. on Information Systems 6, pp. 187.214.

Palkoska, J. and Kung, J., 1997. VQS-A Vague Query System Prototype. In: Int. Workshop on Database and Expert Systems Applications, pp. 614–618.

Pauly, A. and Schneider, M., 2004. Vague Spatial Data Types, Set Operations, and Predicates. In: East-European Conf. on Advances in Databases and Information Systems, pp. 379–392.

Pauly, A. and Schneider, M., 2006. Topological Reasoning for Identifying a Complete Set of Topological Predicates between Vague Spatial Objects. In: FLAIRS Conference, AAAI Press.

Pawlak, Z., 1982. Rough sets. International Journal of Computer and Information Sciences pp. 341–356.

Schneider, M., 1999. Uncertainty Management for Spatial Data in Databases: Fuzzy Spatial Data Types. In: Int. Symp. on Advances in Spatial Databases, Springer-Verlag, pp. 330–351.

Schneider, M., 2001. Fuzzy Topological Predicates, Their Properties, and Their Integration into Query Languages. In: ACM Symp. on Geographic Information Systems (ACM GIS), pp. 9–14.

Schneider, M. and Behr, T., 2006. Topological Relationships between Complex Spatial Objects. ACM Trans. on Database Systems (TODS) 31, pp. 39–81.

Worboys, M., 1998. Imprecision in Finite Resolution Spatial Data. GeoInformatica 2(3), pp. 257–279.