

Querying Vague Spatial Objects in Databases with VASA

Alejandro Pauly and Markus Schneider

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. We introduce the appropriate concepts for enabling database querying that exploits 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 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 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

Existing concepts relevant to this work can be divided into two categories: 1) concepts that provide the foundation for the work presented in this paper; 2) concepts that are defined with goals similar to those of the work in this paper.

Related to the former we are interested in crisp spatial concepts that define the crisp spatial data types for *points*, *lines*, and *regions* [25]. We are also interested in the relationships that can be identified between instances of these types. Topological relationships between spatial objects have been the focus of much research and we concentrate on the concepts defined by the *9-intersection model* originally defined in [10] for simple regions, and later extended for simple regions with holes in [11]. The complete set of topological relationships for all type combinations of complex spatial objects is defined in [25] on the basis of the 9-intersection model.

With respect to concepts for handling spatial vagueness, we further categorize them by their mathematical foundation. Approaches that utilize existing exact (crisp) models for spatial objects include the *broad boundaries* approach [6, 7], the *egg-yolk* approach [8], and the *vague regions* concept [12]. 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*). VASA, our concept for handling spatial vagueness (Section 3) is based on exact models for crisp spatial objects. Although fundamentally different to the exact based approaches, rough set theory [22] provides tools for deriving concepts with a close relation to what can be achieved with exact models. Rough set theory based approaches include early work by Worboys in [26], the concepts for deriving *quality measures* presented in [4], and the concept of *rough classification* in [1].

One of the advantages of fuzzy set theory is the ability of handling *blend-in* type boundaries (such as that between a mountain and a valley). Approaches in this category include earlier *fuzzy regions* [3], the formal definition of *fuzzy points*, *fuzzy lines* and *fuzzy regions* in [23], and an extension of the rough classification from [1] to account for fuzzy regions [2]. A recent effort for the definition of a *spatial algebra* based on fuzzy sets is presented in [9]. Finally, probabilistic approaches [13] focus on an *expected* membership to an object which 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.

Concepts even closer to that dealt with in this paper, namely querying with vagueness are discussed in [17] where it is proposed that vagueness does not necessarily only appear in the data being queried, but can also be part of the query itself. The work in [24] proposes classifications of membership values in order to group sets of values together (near fuzzy concepts). For example, a classification could assign the term “mostly” to high membership values (e.g., 0.95-0.98). In the context of databases in general, the approaches in [15, 18, 19, 16] all propose extensions to query languages on the basis of an operator that enables vague results under different circumstances. For example, in [15] 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 [18] 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 is 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 ν 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^\neg) [25]. We also assume a definition of the geometric set operations union (\oplus), intersection (\otimes), difference (\ominus), and complement (Ξ) between crisp spatial objects such as that from [14].

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

$$\nu(\alpha) = \alpha \times \alpha$$

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

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

We call $w \in \nu(\alpha)$ a (two-dimensional) *vague spatial object* with *kernel part* w_k and *conjecture part* w_c . Further, we call $w_o := (w_k, w_c)$ the *outside part* of w . For $\alpha = \textit{point}$, $\nu(\textit{point})$ is called a *vague point* object and denoted as *vpoint*. Correspondingly, for *line* and *region* we define $\nu(\textit{line})$ resulting in *vline* and $\nu(\textit{region})$ 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 1 illustrates instances of a vague point, a vague line, and a vague region as objects of the data types defined above.

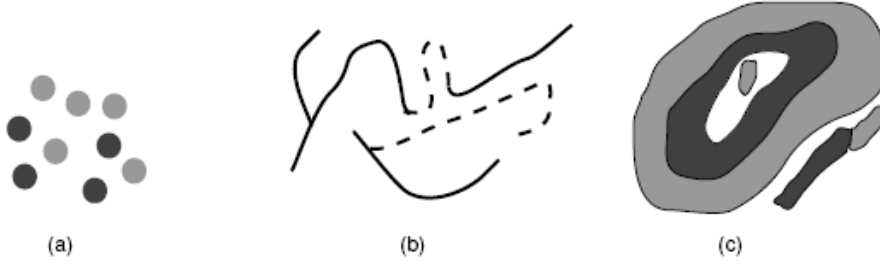


Figure 1: A vague point object (a), a vague line (b) and a vague region (c). Kernel parts are symbolized by dark gray points, straight lines, and dark gray areas. Conjecture parts are symbolized by light gray point, dashed lines, and light gray areas.

3.2 Vague Spatial Operations

For the definition of the vague spatial set operations that compute the *union*, *intersection*, and *difference* between two vague spatial objects, 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 $h: \mathcal{V}(\alpha) \times \mathcal{V}(\alpha) \rightarrow \mathcal{V}(\alpha)$. The complement operation is defined as $complement: \mathcal{V}(\alpha) \rightarrow \mathcal{V}(\alpha)$. 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.

<i>union</i>	<i>k</i>	<i>c</i>	<i>o</i>	<i>intersection</i>	<i>k</i>	<i>c</i>	<i>o</i>	<i>difference</i>	<i>k</i>	<i>c</i>	<i>o</i>	<i>complement</i>	<i>k</i>	<i>c</i>	<i>o</i>
<i>k</i>	<i>k</i>	<i>k</i>	<i>k</i>	<i>k</i>	<i>k</i>	<i>c</i>	<i>o</i>	<i>k</i>	<i>o</i>	<i>c</i>	<i>k</i>	<i>k</i>	<i>o</i>	<i>c</i>	<i>k</i>
<i>c</i>	<i>k</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>c</i>	<i>o</i>	<i>c</i>	<i>o</i>	<i>c</i>	<i>c</i>				
<i>o</i>	<i>k</i>	<i>c</i>	<i>o</i>	<i>o</i>	<i>o</i>	<i>o</i>	<i>o</i>	<i>o</i>	<i>o</i>	<i>o</i>	<i>o</i>				

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

- (i) u **union** w $:= (u_k \oplus w_k, (u_c \oplus w_c) \ominus (u_k \oplus w_k))$
- (ii) u **intersection** w $:= (u_k \otimes w_k, (u_c \otimes w_c) \oplus (u_k \otimes w_c) \oplus (u_c \otimes w_k))$
- (iii) u **difference** w $:= (u_k \otimes (\boxminus(w_k \oplus w_c)), (u_c \otimes w_c) \oplus (u_k \otimes w_c) \oplus u_c \otimes (\boxminus(w_k \oplus w_c)))$
- (iv) **complement** u $:= (\boxminus(u_k \oplus u_c), u_c)$

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 $A \in \nu(\alpha)$, and $B \in \nu(\beta)$ and the set $T_{\alpha\beta}$ of all crisp topological predicates between objects of types α and β [25], 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:

$$p(A_k, B_k) \wedge q(A_k \oplus A_c, B_k) \wedge r(A_k, B_k \oplus B_c) \wedge s(A_k \oplus A_c, B_k \oplus B_c)$$

We define the set $V_{\alpha\beta}$ of all vague topological predicates between objects of types $\nu(\alpha)$ and $\nu(\beta)$. 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:

$$(overlap(A_k, B_k), disjoint(A_k, B_k \oplus B_c), disjoint(A_k \oplus A_c, B_k), disjoint(A_k \oplus A_c, B_k \oplus B_c)).$$

In this example the implications of $overlap(A_k, B_k) \Rightarrow A_k^\circ \cap B_k^\circ \neq \emptyset$ and $disjoint(A_k, B_k \oplus B_c) \Rightarrow A_k^\circ \cap (B_k \oplus B_c)^\circ = \emptyset$ clearly show a contradiction.

In [21], 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). Each BSCN is tested for *path-consistency* which is used to check, via constraint propagation, that all original constraints are consistent; otherwise the inconsistency indicates an invalid 4-tuple.

For each type combination of *vpoint*, *vline*, and *vregion*, possibly thousands of predicates are recognized. 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.

4 QUERYING WITH VASA

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 presented in Section 4.2 extends SQL to enable handling of 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 :

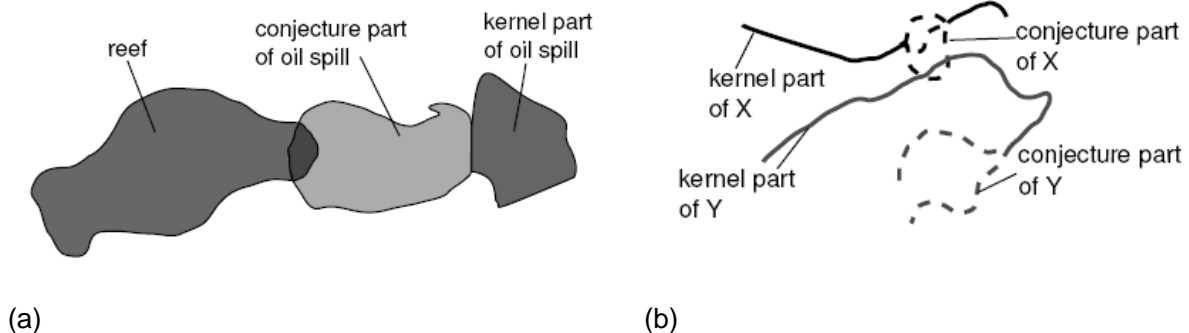


Figure 2: (a) A representation of an ecological scenario using vague regions. (b) Scenario illustrating the use of vague lines to represent routes of suspected terrorists X and Y .

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 2(a), 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 2(b), 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.

```
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 3 where the data can be stored in table *animals*(*id* : *INT* , *name* : *STRING*, *roam area* : *VREGION* , *mig route* : *VLINE* , *drink spot* : *VPOINT*).

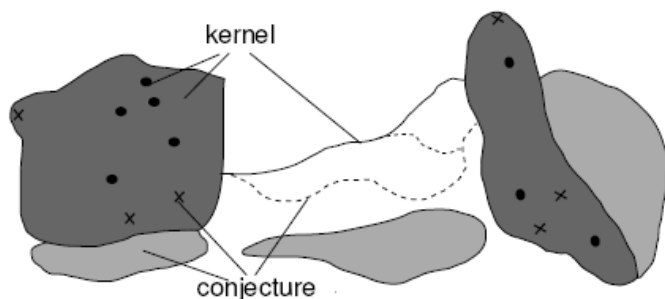


Figure 3: The vague spatial object representation of an animal's roaming areas, migration routes and drinking spots.

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 recognize animal species with low counts that could be extinct due to 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.avgsz<40 AND l.avgsz>80 AND s.count<100;
```

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 for Vague Queries on Vague Spatial Data

We analyze the approaches introduced in Section 2 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  $\sim$ 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() $\sim$ 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 dealing 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() $\sim$ 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 seem 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 design of VASA.

ACKNOWLEDGEMENTS

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

References

- [1] O. Ahlqvist, J. Keukelaar, and K. Oukbir. *Rough Classification and Accuracy Assessment*. Int. Journal of Geographical Information Sciences, 14:475–496, 2000.
- [2] O. Ahlqvist, J. Keukelaar, and K. Oukbir. *Rough and Fuzzy Geographical Data Integration*. Int. Journal of Geographical Information Sciences, 17:223–234, 2003.
- [3] D. Altman. *Fuzzy Set Theoretic Approaches for Handling Imprecision in Spatial Analysis*. Int. Journal of Geographical Information Systems, 8(3):271–289, 1994.
- [4] T. Beaubouef and F. Petry. *A Rough Set Foundation for Spatial Data Mining Involving Vague Regions*. In IEEE Int. Conf. on Fuzzy Systems, pp. 767 – 772. 2002.
- [5] P. A. Burrough and A. U. Frank, editors. *Geographic Objects with Indeterminate Boundaries*. Taylor & Francis, 1996.
- [6] E. Clementini and P. Di Felice. *A Spatial Model for Complex Objects with a Broad Boundary Supporting Queries on Uncertain Data*. Data & Knowledge Engineering, 37:3, 285–305, 2001.
- [7] E. Clementini and P. Di Felice. *An Algebraic Model for Spatial Objects with Indeterminate Boundaries*, pages 153–169. In Burrough and Frank [5], 1996.
- [8] A. G. Cohn and N. M. Gotts. *The ‘Egg-Yolk’ Representation of Regions with Indeterminate Boundaries*, pages 171–187. In Burrough and Frank [5], 1996.
- [9] A. Dilo, R.A. de By, and A. Stein. *A System of Types and Operators for Handling Vague Spatial Objects*. International Journal of Geographical Information Science, 21:4, 397 – 426.
- [10] M. J. Egenhofer. *A Formal Definition of Binary Topological Relationships*. In Int. Conf. on Foundations of Data Organization and Algorithms, pages 457–472. Springer-Verlag, 1989.
- [11] M.J. Egenhofer, E. Clementini, and P. Di Felice. *Topological Relations between Regions with Holes*. Int. Journal of Geographical Information Systems, 8:128–142, 1994.
- [12] M. Erwig and M. Schneider. *Vague Regions*. In 5th Int. Symp. on Advances in Spatial Databases, pages 298–320. Springer-Verlag, 1997.
- [13] J. T. Finn. *Use of the Average Mutual Information Index in Evaluating Classification Error and Consistency*. Int. Journal of Geographical Information Systems, 7(4):349–366, 1993.
- [14] R. H. Guting and M. Schneider. *Realm-Based Spatial Data Types: The ROSE Algebra*. VLDB Journal, 4:100–143, 1995.
- [15] T. Ichikawa and M. Hirakawa. *ARES: A Relational Database with the Capability of Performing Flexible Interpretation of Queries*. IEEE Trans. on Software Engineering, 12:624–634, 1986.
- [16] J. Kung and J. Palkoska. *Vague Joins – An Extension of the Vague Query System VQS*. In 9th Int. Workshop on Database and Expert Systems Applications, pp. 997–1001, 1998.
- [17] D.H. Lee and M.H. Kim. *Accommodating Subjective Vagueness Through a Fuzzy Extension to the Relational Data Model*. Information Systems, 18:363–374, 1993.
- [18] A. Motro. *VAGUE: A User Interface to Relational Databases that Permits Vague Queries*. ACM Trans. on Information Systems, 6:187.214, 1988.

- [19] J. Palkoska and J. Kung. *VQS-A Vague Query System Prototype*. In Int. Workshop on Database and Expert Systems Applications, pages 614–618, 1997.
- [20] A. Pauly and M. Schneider. *Vague Spatial Data Types, Set Operations, and Predicates*. In East-European Conf. on Advances in Databases and Information Systems, pages 379–392, 2004.
- [21] A. Pauly and M. Schneider. *Topological Reasoning for Identifying a Complete Set of Topological Predicates between Vague Spatial Objects*. In FLAIRS Conference. 2006.
- [22] Z. Pawlak. *Rough sets*. International Journal of Computer and Information Sciences, pages 341–356, 1982.
- [23] M. Schneider. *Uncertainty Management for Spatial Data in Databases: Fuzzy Spatial Data Types*. In Int. Symp. on Advances in Spatial Databases, pp. 330–351. Springer-Verlag, 1999.
- [24] M. Schneider. *Fuzzy Topological Predicates, Their Properties, and Their Integration into Query Languages*. In ACM Symp. on Geographic Information Systems, pp. 9–14, 2001.
- [25] M. Schneider and T. Behr. *Topological Relationships between Complex Spatial Objects*. ACM Trans. on Database Systems (TODS), 31:39–81, 2006.
- [26] M. Worboys. *Imprecision in Finite Resolution Spatial Data*. GeoInformatica, 2(3):257–279, 1998.