Topological Reasoning for Identifying a Complete Set of Topological Predicates between Vague Spatial Objects

Alejandro Pauly and **Markus Schneider***
Computer & Information Science and Engineering
University of Florida
Gainesville, Florida 32611
{apauly,mschneid}@cise.ufl.edu

Abstract

Current implementations of spatial databases are not capable of handling data that is intrinsically indeterminate or vague. The goal of the *Vague Spatial Algebra* (VASA) project is to define concepts and implementations that will address this problem. In this paper we further develop work on topological predicates between vague spatial objects. The new developments make use of spatial reasoning techniques that provide the necessary tools for the definition of a mechanism to identify a complete set of vague topological predicates. These predicates work on a definition of vague spatial data types defined previously as part of VASA and that are based on well known definitions of crisp spatial data types.

Introduction

The widespread use of spatial databases and specifically GIS has triggered an increase in research efforts on spatial reasoning within a database context. Much of this research is geared towards the optimization of queries by inferring implicit information about the relationships between spatial objects stored in the database. The types of relationships include direction, distance and topological relationships among others. Topological relationships like *overlap*, *disjoint* and *inside* characterize the relative position of spatial objects and remain unchanged under continuous transformations like translation, rotation and scaling. These relationships are purely qualitative and have proven important for many applications of spatial reasoning and spatial databases in general and so have been widely studied in the literature.

Our work currently focuses on the design of a *Vague Spatial Algebra* (VASA). The goal of VASA is to allow for uncertainty handling in spatial databases and involves the formalization of *vague topological predicates* that define topological relationships between *vague spatial objects*. The *vague data types*, their operations and predicates defined as part of VASA will provide the conceptual foundation for the implementation of a software library that will extend current database management systems to allow such types of data to be manipulated. During the definition of vague topological predicates several issues have been resolved thanks to the application of topological reasoning. This marks a new kind of objective for which topological reasoning proves useful.

The goal of this paper is to refine our mechanism that identifies vague topological predicates by including a topological reasoning method that ensures completeness of the set of predicates generated. The application of such a method represents the last step that is required to formalize our concept of vague topological predicates.

In order to fully understand the mechanism presented here, it is necessary to cover the context in which it was developed. This paper starts by exploring the necessary related work and then continues by introducing *vague spatial data types*. The original mechanism for identifying vague topological predicates is described before we describe major developments to the mechanism that allow to resolve the completeness issue stated above. The results of identifying vague topological predicates are shown before conclusions are drawn and the direction of future work is discussed in the last section.

Related Work

The contextual background necessary for the definition of vague spatial data types and vague topological predicates is defined in this section.

Spatial Data Types

We concentrate in the definitions of *simple spatial data types* and *complex spatial data types*. The types of the first kind have a simple structure as illustrated in Figure 1. A *simple point* describes an element of the Euclidean plane $\mathbb{R}^2$. A *simple line* is a one-dimensional, continuous geometric structure embedded in $\mathbb{R}^2$ with two end points. A *simple region* is a two-dimensional point set in $\mathbb{R}^2$ and topologically equivalent to a closed disk.

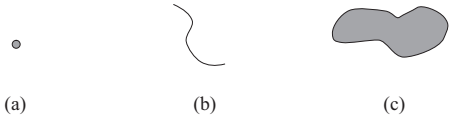

Figure 1: Examples of a simple point object (a), a simple line object (b), and a simple region object (c).

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

Lack of closure properties of the simple spatial data types as well as stronger application requirements have resulted in the definition of the more general *complex spatial data types* illustrated in Figure 2 (see (Schneider 1997) for a survey).

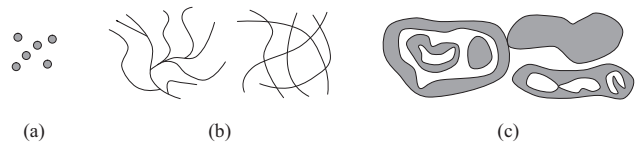

Figure 2: Examples of a complex point object (a), a complex line object (b), and a complex region object (c).

A *complex point* is a finite point collection (e.g., ATM locations in the city of Cartago). A *complex line* is an arbitrary, finite collection of one-dimensional curves, i.e., a spatially embedded network possibly consisting of several disjoint connected components (e.g., the roads in The Hague). A *complex region* permits multiple areal components, called *faces*, and holes in faces (e.g., Italy with its mainland and offshore islands as components and with the Vatican as a hole). The original simple spatial data types turn out to represent special cases of their complex counterparts. Thus the complex spatial data type specification is sufficient to cover all cases also covered by simple spatial data types. This formal and complete definition of complex points, complex lines and complex regions can be found in (Schneider & Behr 2006).

Topological Predicates

Topological relationships between spatial objects have been a focus of interdisciplinary research and have also been widely studied in the spatial database related literature. This type of relationships is purely qualitative and describes 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. Due to its suitability for our purposes, we concentrate 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, Clementini, & Di Felice 1994). The complete set of topological relationships for all type combinations of complex spatial objects is defined in (Schneider & 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 & 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 better usability of the predicates identified.

Vague Spatial Data Types

Naturally occurring phenomena in space often (if not always) cannot be precisely defined because of the intrinsic uncertainty of their features. The location of animal refuges might not be precisely known, and the path of rivers might be uncertain due to water volume fluctuations and changing land characteristics. The extension of lakes can also change and thus have uncertain areas. All these are examples of so-called *vague spatial objects*. The animal refuge locations can be modeled as a *vague point* object where the precisely known locations are called the *kernel point* object and the assumed locations are denoted as the *conjecture point* object. The river paths can be modeled as *vague line* objects. Some segments or parts of the path, called *kernel line* objects, can be definitely identified since they are always part of the river. Other paths can only be assumed, and these are denoted as *conjecture line* objects. Knowledge about the extension of lakes can be modeled similarly with *vague regions* formed by *kernel* and *conjecture* parts. Figure 3 gives some illustrations. Grey shaded areas, straight lines, and grey points indicate kernel parts; areas with white interiors, dashed lines, and white points refer to conjecture parts.

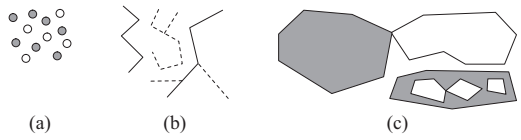

Figure 3: Examples of a (complex) vague point object (a), a (complex) vague line object (b), and a (complex) vague region object (c). Each collection of components forms a single vague object.

For the definition of vague points, vague lines, and vague regions we leverage the well known data types *point* for crisp points, *line* for crisp lines, and *region* for crisp regions (Schneider & Behr 2006). These types are closed under the geometric set operations *union*, *intersection* , *difference*, and

complement. The use of an exact model for constructing vague spatial data types leads to the benefit that existing definitions, techniques, data structures and algorithms need not be redeveloped but can simply be used or in the worst case slightly modified or extended as necessary.

A vague spatial object is defined by a pair of two *disjoint* or *meeting* crisp complex spatial objects. Hence, the same generic definition is applicable to all vague spatial data types. That is, the extension of a crisp spatial data type to a corresponding vague type is given by a type constructor v as follows:

$$v(\alpha) \quad = \quad \alpha \times \alpha \qquad\qquad \forall\, \alpha \in \{\textit{point}, \textit{line}, \textit{region}\}$$

This means that for $\alpha = \textit{region}$ we obtain $v(\textit{region}) = \textit{region} \times \textit{region}$, which we also name *vregion*. Accordingly, the data types *vline* and *vpoint* are defined. For a vague spatial object $A = (A_k, A_c) \in v(\alpha)$, the first crisp spatial object A_k, called the *kernel part*, describes the determinate component of A, that is, the component that definitely and always belongs to the vague object. The second crisp spatial object A_c, called the *conjecture part*, describes the vague component of A, that is, the component from which we cannot say with any certainty whether it or subparts of it belong to the vague object or not. *Maybe* the conjecture part or subparts of it belong to the vague object, *maybe* this is not the case. Since the kernel part and the conjecture part of the *same* vague spatial object may not share interior points, we define:

$$\forall\, \alpha \in \{\textit{point}, \textit{line}, \textit{region}\}$$
$$\forall\, A = (A_k, A_c) \in v(\alpha) : A_k^\circ \cap A_c^\circ = \emptyset$$

More details, in particular about the semantics of vague spatial data types as well as the definition of vague spatial operations, can be found in (Pauly & Schneider 2004).

A Method for Identifying Vague Topological Predicates

Topological predicates between vague spatial objects (or *vague topological predicates*) can be used to identify the topological relationships between two vague spatial objects and should take into account the objects' vague properties (provided in our definition by the *conjecture part*). For example, it is interesting to known whether two lakes represented by vague regions overlap, and more so, to known whether this overlap includes conjecture parts or not. A primary goal of the definition of *vague spatial data types* has been the leveraging of well defined notions of *crisp spatial data types*. We follow the same idea and define vague topological predicates on the basis of well defined notions of topological predicates between *crisp spatial objects*. The mechanism originally presented in (Pauly & Schneider 2005b) is illustrated in Figure 4.

Vague topological predicates are characterized by way of the crisp topological predicates that hold between the pairs (A_k, B_k), $(A_k, B_k \oplus B_c)$[1], $(A_k \oplus A_c, B_k)$ and $(A_k \oplus A_c, B_k \oplus B_c)$. The four pairs, which are composed of crisp

[1] $\oplus$ denotes the spatial union operation as defined in (Schneider 1997)

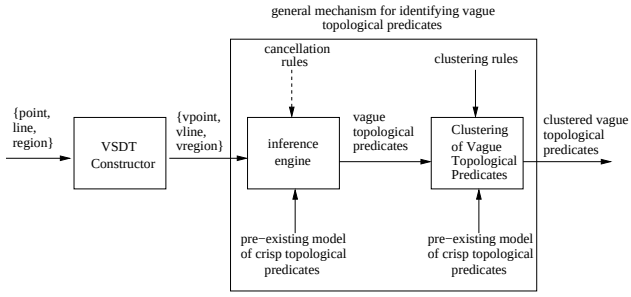

Figure 4: Overview of the derivation of vague topological predicates

spatial objects, are precisely those that are used to describe the kernel and conjecture components of vague spatial objects. In other words, we can say that the vague topological predicate between two vague spatial objects is represented by a 4-tuple of crisp topological predicates between crisp spatial component objects. The main task of the identification mechanism is to find the 4-tuples that represent valid vague topological predicates. Note that not all 4-tuple permutations of crisp topological predicates result in valid vague topological predicates. If one element in the tuple directly contradicts any other element in the tuple, it makes the tuple *topologically inconsistent* and thus invalid. Take for example a 4-tuple for vague regions A and B: ($\textit{overlap}(A_k, B_k), \textit{disjoint}(A_k, B_k \oplus B_c), \textit{disjoint}(A_k \oplus A_c, B_k), \textit{disjoint}(A_k \oplus A_c, B_k \oplus B_c)$). In this example $\textit{overlap}(A_k, B_k) \Rightarrow A_k^\circ \cap B_k^\circ \neq \emptyset$ and $\textit{disjoint}(A_k, B_k \oplus B_c) \Rightarrow A_k^\circ \cap (B_k \oplus B_c)^\circ = \emptyset$. 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 $\textit{disjoint}(A_k, B_k \oplus B_c)$. The elimination of invalid tuples is done by an invalidation process and represents the first step in the mechanism that identifies all vague topological predicates. After this step, each remaining tuple will (or should) represent a unique vague topological predicate. The invalidation of inconsistent 4-tuples is done by an inference mechanism that works on the basis of so-called *cancellation rules*. The cancellation rules must be manually defined for all different combinations of vague spatial data types (Pauly & Schneider 2005b).

Even though many (in most cases the majority) of the tuples are invalidated, the number of valid tuples representing vague topological predicates can remain rather large and difficult to handle. For this reason the mechanism from (Pauly & Schneider 2005a; 2005b) introduces the notion of *clustering* in which tuples can be clustered into sets so that only a small number of (clustered) vague topological predicates needs to be handled. These clusters can be predefined or customized by the user depending on her needs. Due to the uncertainty that is handled through the conjecture parts of the spatial objects involved, the clusters are defined as predicates in a three-valued logic (*true,false,maybe*). Each clustered vague topological predicate results in *true* when

the predicate definitely holds, *false* when it definitely does not hold, and *maybe* when the inclusion of the conjecture part makes it impossible to determine whether the predicate holds or not.

An example set of clustered vague topological predicates is defined in (Pauly & Schneider 2005b). The clusters in this set are named *Disjoint*, *Overlap*, *Meet*, *Contains*, *Inside*, *Covers*, *CoveredBy* and *Equal* and can be regarded as the vague counterparts of the eight original topological predicates between crisp simple regions. The semantics definition of the clustered vague predicates is based on point set topological concepts. In order to use the clustered vague topological predicates for querying in databases, the reference shows a transformation from the three-value logic to boolean logic. Sample queries illustrate all concepts.

Completeness of Vague Topological Predicates

Completeness is an important issue with the original definition of the mechanism that employs cancellation rules. How can we be sure that all tuples that need to be invalidated are covered by the current set of cancellation rules? Proving correctness of each rule can be done as shown in (Pauly & Schneider 2005b); hence it is known that tuples were correctly invalidated. But this only proves the correctness of the mechanism and does not say anything about completeness. In this section we show how the cancellation rules can be replaced with a more general inference engine. The engine combines concepts from topological reasoning and binary constraint networks in order to invalidate tuples and generates the complete set of vague topological predicates. We proceed by introducing these concepts and then applying them to our problem.

Topological Reasoning

Spatial reasoning in general is important in a variety of areas due to the fact that it provides new spatial information that is currently not explicitly stored and available to the user. We refer to topological reasoning as the specific type of spatial reasoning that generates information solely from the topological relations that are currently known. In spatial databases topological relationships are accessed through topological predicates like those reviewed in the *Related Work* section. A common method for topological reasoning involves the derivation of the composition of two (or more) topological relationships. A valid composition as noted in (Egenhofer 1994; Abdelmoty & El-Geresy 1995) is formally defined for general relations in (Tarski 1941). Given a (topological) relationship P_1 between (spatial) objects A and B and a (topological) relationship P_2 between (spatial) objects B and C, the composition $P_1; P_2$ (at least partially) provides the (topological) relationship P_3 that holds between A and C. It is said that P_3 is only partially defined if it is impossible to derive complete information from the given pair of relationships. In such cases, P_3 is considered a disjunction of relationships which in the worst case (when nothing of P_3 can be inferred) covers the complete set of relationships defined between the object types of A and C. As an example of this extreme case we derive the topological relationship between simple regions A and C. Assume that it is known that *overlap*(A, B) and *overlap*(B, C) hold. The only determination that the composition *overlap*(A, B); *overlap*(B, C) can achieve is that the topological predicate between A and C can be any of the 8 predicates defined between simple regions (i.e., the derivation cannot infer any unique values for the derived 9-intersection matrix):

$$\begin{aligned} \textit{overlap}; \textit{overlap} = \{&\textit{overlap}, \textit{disjoint}, \textit{meet}, \textit{contains}, \\ &\textit{inside}, \textit{covers}, \textit{coveredBy}, \textit{equal}\} \end{aligned}$$

Based on ideas of set inclusion and containment of point sets, (Egenhofer 1994) gives a set of detailed rules that can be applied to 9-intersection matrices (for crisp topological predicates). The result of this composition is a new 9-intersection matrix that represents the derived predicate (or set of predicates). The composition results in a partial derivation if not all 9 values of the matrix can be certainly inferred. The authors in (Abdelmoty & El-Geresy 1995) define a very similar but more general set of rules and apply them to a topological predicate model that is similar to the 9-intersection. They also cover issues regarding composition of topological predicates between objects of different types and issues involving their own definition of complex spatial objects which highly differs from that shown in the *Related Work* section. Figure 5 provides a sample composition of predicates between simple regions from (Egenhofer 1994). The desired result of both papers is in the form of composition tables that provide all results for all possible combinations of two topological predicates with one object in common.

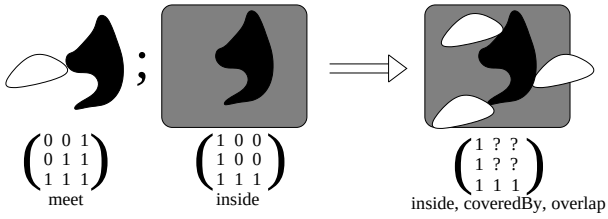

Figure 5: Sample composition resulting in partial derivation

Binary Constraint Networks

Binary constraint networks (BCN) (Ladkin & Maddux 1994) are well known structures used to represent the widely studied binary constraint problems. A BCN is a labeled digraph where each node represents a variable and each ordered pair of nodes is connected by a directed edge labeled with the relations between the variables. Let M be the adjacency matrix (also called *constraint matrix*[2]) representation of a BCN with n variables. The edge between two variables Q and R is annotated by a set of relationships $\{r_1, r_2, ..., r_z\}$ such that:

$$(r_1(Q, R)) \vee r_2(Q, R)) \vee \ ... \ \vee r_z(Q, R)) = \textit{true}$$

[2]A constraint matrix must have identity relations on the main diagonal. Clearly $M_{ii} = M_{ik}; M_{ki}$

M is said to be *path-consistent* if $M_{ij} \subseteq M_{ik}; M_{kj}$ for all $i, j, k < n$. Path-consistency refers to consistency that not only holds directly for every individual triple of vertices and their edges in the graph but also holds in general for any cycle path in the BCN. Generally, M is path-consistent if, and only if, for every finite set of indices $k_1, k_2, ..., k_m < n$ and all $i, j < n$ it holds that:

$$M_{ij} \subseteq M_{ik_1}; M_{k_1k_2}; M_{k_2k_3}; ...; M_{k_{m-1}k_m}; M_{k_mj}$$

Given a BCN of n vertices in which some relationships are well known while others are only partially defined, the path-consistency concept can be applied to derive specific information about the partially defined relationships. Iterative verification of path-consistency for all M_{ij} where $i, j < n$ can remove any relationships that are *path-inconsistent* until no inconsistent edges exist in the BCN. In a single iteration the consistency of all directly related variables is assured. Any inconsistent relation is removed and another iteration must take place to ensure that anything that was consistent by the removal of the inconsistent relationship, remains consistent.

In (Smith & Park 1992), the theory of BCN and path-consistency is applied to spatial knowledge and used as a reasoning system that besides inferring new information is able to check for consistency of the existing spatial knowledge. A *binary spatial constraint network* (BSCN) is a BCN in which the variables are spatial objects and the edges refer to the topological relationships between the objects. It is important to observe that given three spatial objects A, B and C and the topological relationships between (A, B) and (B, C) we can represent this as a triangle BSCN and the path-consistency can be applied to infer the relationships (A, C). Similarly, if all three relationships were given, the path-consistency method can be used to determine whether all three relationships are consistent.

Solving the Completeness Issue

Recall that the invalidation of a 4-tuple occurs only when there is a contradiction between two elements of a tuple. Because such a contradiction amounts to topological inconsistency we can easily apply the concept of path-consistency to the invalidation of 4-tuples. Each 4-tuple can be represented as a BSCN from (Smith & Park 1992) and the compositions required to verify the path-consistency of a tuple can be derived using the rules from (Egenhofer 1994).

The first step needed to verify each 4-tuple is to construct the BSCN which we illustrate in Figure 6. Four crisp spatial objects are represented as vertices and labeled A_k, B_k, A for $A_k \oplus A_c$ and B for $B_k \oplus B_c$. The 4-tuple directly provides 8 edges (the 4 given predicates and their converses). We label these edges as t_{kk} for the predicate between A_k and B_k, t_{kB} for the predicate between A_k and $B_k \oplus B_c$, t_{Ak} for the predicate between $A_k \oplus A_c$ and B_k, and t_{AB} for the predicate between $A_k \oplus A_c$ and $B_k \oplus B_c$ The converse for a predicate (or set of predicates) p is denoted as $\overline{p}$. Given a vague spatial object Q, we define P_Q as the set of crisp topological predicates that operate between two objects of the same type as the crisp components of Q. The identity relationship of P_Q is defined as $I(P_Q)$ (e.g. the *equal* predicate in the case of simple regions). The rest of the edges in the BSCN can be inferred from the fact that $Q_k^\circ \subseteq (Q_k \oplus Q_c)^\circ$ for any vague spatial object Q. We denote the relationship between Q_k and $Q_k \oplus Q_c$ as $in(P_Q)$ such that for two spatial objects E and F holds:

$$\forall q \in in(P_Q) : q(E, F) \Rightarrow E^\circ \subseteq F^\circ$$

Notice that $in(P_Q)$ can possibly refer to more than one topological predicate (i.e., a set of topological relationships. For example: $\{$*inside*, *coveredBy*, *equal*$\}$ in the simple region case). Once the BSCN is constructed, the path-consistency procedure that is detailed in (Smith & Park 1992) can be applied to determine if there is an inconsistency in the 4-tuple.

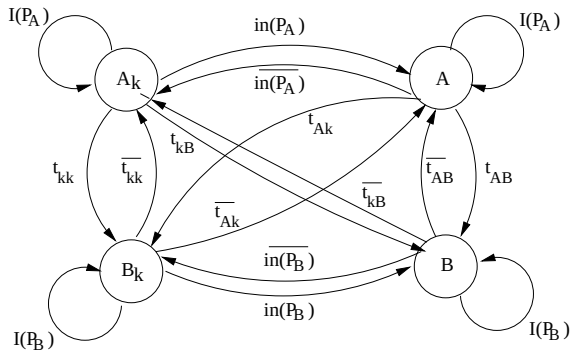

Figure 6: BSCN for 4-tuple consistency check

To identify all vague topological predicates we create an inference engine that replaces the cancellation rules and is based on the algorithm in Figure 7. This algorithm iteratively generates the tuples for given vague spatial data types and determines whether each tuple is valid or not. All valid 4-tuples represent vague topological predicates that are later handled by the clustering mechanism.

Besides the obvious advantage of proven completeness by this method, we also remove the problem of manually generating the sets of cancellation rules. Such sets were data type combination dependent and had to be defined for every single data type combination. By using path-consistency as the basis of the inference engine we just need the composition tables which can be generated for all simple and complex spatial data types (from the *Related Work* section) by using the derivation mechanism provided in (Egenhofer 1994).

The Complete Set of Vague Topological Predicates

With all the pieces in place, we have been able to implement a program we call *Composer* that derives the composition of topological predicates between complex spatial objects. This implementation uses the complex spatial data types and their topological predicates as defined in (Schneider & Behr 2006). The composition tables are generated using the inference rules from (Egenhofer 1994) and are fed into another program that implements the path-consistency procedure from (Smith & Park 1992) and the algorithm from

Algorithm *IdentifyVTP*
input:
(i) Vague spatial data types $v(\alpha)$ and $v(\beta)$,
(ii) complete set $P_{\alpha\alpha}$ of crisp topological predicates between two objects of type α
(iii) complete set $P_{\beta\beta}$ of crisp topological predicates between two objects of type β
(iv) complete set $P_{\alpha\beta}$ of crisp topological predicates between two objects of types α and β respectively. If $\alpha = \beta$ then inputs (ii), (iii) and (iv) represent the same set of predicates.
output:
Complete set of vague topological predicates
begin
for each predicate $t_{kk} \in P_{\alpha\beta}$ **do**
for each predicate $t_{kB} \in P_{\alpha\beta}$ **do**
for each predicate $t_{Ak} \in P_{\alpha\beta}$ **do**
for each predicate $t_{AB} \in P_{\alpha\beta}$ **do**
M = Generate BSCN with 4-tuple $(t_{kk}, t_{kB}, t_{Ak}, t_{AB})$ and sets $in(P_{\alpha\alpha}), in(P_{\beta\beta}), I(P_{\alpha\alpha}), I(P_{\beta\beta})$
if (M is path-consistent)
add $(t_{kk}, t_{kB}, t_{Ak}, t_{AB})$ to output
endfor;
endfor;
endfor;
endfor;
end *IdentifyVTP*

Figure 7: Algorithm for identifying the complete set of vague topological predicates for type combination $(v(\alpha), v(\beta))$

Figure 7. The program is successful in invalidating all necessary 4-tuples and resulting in the complete set of vague topological predicates. The number of predicates for each vague data type combination of $v(\textit{point})$, $v(\textit{line})$ and $v(\textit{region})$ is rather large (as seen in Table 1). Thus the clustering mechanism from (Pauly & Schneider 2005b) still proves useful in generating an accessible set of predicates for the user.

	$v(\textit{point})$	$v(\textit{line})$	$v(\textit{region})$
$v(\textit{point})$	51	974	166
$v(\textit{line})$	974	471650	74916
$v(\textit{region})$	166	74916	55880

Table 1: Number of identified vague topological predicates on the basis of complex spatial data types.

Conclusion

In this paper a mechanism for the complete definition of vague topological predicates is provided. To achieve this definition, we have modified a previously presented mechanism that identified vague topological predicates but lacked the critical property of completeness. The modifications include concepts from several topics in topological reasoning and binary constraint networks. Besides solving the issue of completeness that existed in the previous version of the mechanism, the new version represents a more robust mechanism in which the identification of vague topological predicates is data type independent. The new ideas presented also provide an example of the application of binary constraint networks and topological reasoning.

Future work involving VASA includes the implementation of a software library that integrates the concepts established here and in previous papers into a fully usable database extender. This implementation will be the last step in developing the Vague Spatial Algebra.

References

Abdelmoty, A. I., and El-Geresy, B. A. 1995. A general method for spatial reasoning in spatial databases. In *ACM Int. Conf. on Information and Knowledge Management (CIKM)*, 312–317.

Egenhofer, M.; Clementini, E.; and Di Felice, P. 1994. Topological relations between regions with holes. *Int. Journal of Geographical Information Systems* 8(2):128–142.

Egenhofer, M. J. 1989. A formal definition of binary topological relationships. In *3rdInt. Conf. on Foundations of Data Organization and Algorithms*, LNCS 367, 457–472. Springer-Verlag.

Egenhofer, M. J. 1994. Deriving the composition of binary topological relations. *Journal of Visual Languages and Computing* 2:133–149.

Ladkin, P., and Maddux, R. 1994. On binary constraint problems. *Journal of the Association for Computing Machinery* 41:435–469.

Pauly, A., and Schneider, M. 2004. Vague spatial data types, set operations, and predicates. In *8thEast-European Conf. on Advances in Databases and Information Systems*, 379–392.

Pauly, A., and Schneider, M. 2005a. Identifying topological predicates for vague spatial objects. In *ACM Symp. for Applied Computing (ACM SAC)*, 587–591. ACM.

Pauly, A., and Schneider, M. 2005b. Topological predicates between vague spatial objects. In *9th Int. Symp. on Spatial and Temporal Databases (SSDT)*, Lecture Notes in Computer Science, 418–432. Springer.

Schneider, M., and Behr, T. 2006. Topological relationships between complex spatial objects. *ACM Trans. on Database Systems (TODS)*. Accepted for Publication.

Schneider, M. 1997. *Spatial Data Types for Database Systems - Finite Resolution Geometry for Geographic Information Systems*, volume LNCS 1288. Berlin Heidelberg: Springer-Verlag.

Smith, T., and Park, K. 1992. An algebraic approach to spatial reasoning. *Int. Journal of Geographical Information Systems* 6:177–192.

Tarski, A. 1941. On the calculus of relations. *Journal of Symbolic Logic* 6:73–89.