Vague Spatial Data Types, Set Operations, and Predicates

Alejandro Pauly & Markus Schneider*

University of Florida
Department of Computer & Information Science & Engineering
Gainesville, FL 32611, USA
{apauly, mschneid}@cise.ufl.edu

Abstract. Many geographical applications deal with spatial objects that cannot be adequately described by determinate, crisp concepts because of their intrinsically indeterminate and vague nature. Current geographical information systems and spatial database systems are unable to cope with this kind of data. To support such data and applications, we introduce *vague spatial data types* for *vague points*, *vague lines*, and *vague regions*. These data types cover and extend previous approaches and are part of a data model called *VASA* (*Vague Spatial Algebra*). Their formal framework is based on already existing, general exact models of crisp spatial data types, which simplifies the definition of the vague spatial model. In addition, we obtain executable specifications for the operations which can be immediately used as implementations. This paper gives a formal definition of the three vague spatial data types as well as some basic operations and predicates. A few example queries illustrate the embedding and expressiveness of these new data types in query languages.

1 Introduction

The current mapping of spatial phenomena of the real world to exclusively crisp, i.e., precisely determined, spatial objects is an insufficient abstraction process for many geometric applications since often the feature of *spatial vagueness* or *spatial indeterminacy* is inherent to many geometric and geographic data [2]. Applications based on this kind of geometric data are so far not covered by current GIS and spatial database systems.

So far, often contrary to reality, spatial data modeling implicitly assumes that the positions of points, the locations and routes of lines, and the extent and hence the boundary of regions are precisely determined and universally recognized. The properties of the space at points, along lines, and within regions are given by attributes whose values are assumed to be constant over the whole objects. Examples are man-made spatial objects (e.g., monuments, highways, buildings) and predominantly immaterial spatial objects (e.g., countries, districts, land parcels with their political, administrative, and cadastral boundaries). We denote this kind of entities as *crisp* or *determinate spatial objects*.

On the other hand, there are many geometric applications in which positions of points are not exactly known, the locations and routes of lines are unclear, and regions

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

do not have sharp boundaries, or their boundaries cannot be precisely determined. Examples are social or natural phenomena (e.g., terrorists' refuges and escape routes, population density, unemployment rate, soil quality, vegetation, oceans, oil fields, biotopes, deserts). We denote this kind of entities as *vague* or *indeterminate spatial objects*.

This paper presents an object model for defining *vague spatial data types* for *vague points*, *vague lines*, and *vague regions*. These types are part of a data model called *VASA* (*Vague Spatial Algebra*). The model rests on "traditional" (i.e., exact) modeling techniques and extends, rather than replaces, the current theory of spatial database systems and GIS. Further, moving from an exact to a vague domain does not necessarily invalidate conventional (computational) geometry; it is merely an extension. Hence, exact object models can be considered as special cases of our vague object model. All vague spatial data types and several vague spatial operations are defined generically, i.e., without type-specific definitions. Since our vague spatial data types and operations are based on their crisp counterparts and can be expressed by them, we obtain *executable specifications* that can be directly used as an implementation. In this paper, we do not aim at developing a type system with a "complete" set of operations and predicates. The goal is more to demonstrate the power, simplicity, and expressiveness of our model.

Section 2 discusses related work. Section 3 informally introduces the concept of vague spatial objects and motivates it by giving some application examples. Section 4 gives a generic definition of vague spatial data types and vague spatial set operations. Section 5 deals with type-specific operations. Section 6 introduces some vague topological predicates. Section 7 illustrates the embedding of vague spatial data types into query languages. Finally, Section 8 draws some conclusions and addresses future work.

2 Related Work

Spatial vagueness has to be seen in contrast to spatial uncertainty resulting from either a lack of knowledge about the position and shape of an object (*positional* uncertainty) or the inability of measuring such an object precisely (*measurement* uncertainty). Much literature, which we will not consider here, has been published on dealing with positional and measurement uncertainty; it mainly proposes probabilistic models. Spatial vagueness is an intrinsic feature of a spatial object where we cannot be sure whether certain components belong to the spatial object or not. Our vague spatial data types cover both aspects of spatial uncertainty and spatial vagueness.

Three main alternatives have been proposed as general design methods. *Models based on fuzzy sets* (e.g., [1, 11]) are all based on fuzzy set theory, allow a much more fine-grained modeling of vague spatial objects, but are computationally much more expensive with respect to data structures and algorithms. *Models based on rough sets* (e.g., [13]) work with lower and upper approximations of spatial objects, which is similar to our approach. But the formal background is rather different. *Models based on exact spatial objects* (e.g., [3, 4, 10, 6] extend data models, type systems, and concepts for crisp spatial objects to vague spatial objects. The model described in this paper belongs to this latter category.

A benefit of the exact object model approach is that existing definitions, techniques, data structures, algorithms, etc., need not be redeveloped but only modified and ex-

tended, or simply used. So far, four object models have been proposed for vague regions. The first three models use some kind of zone concept, either without holes [3, 4] or with holes [10]. The central idea is to consider determined zones surrounding the indeterminate boundaries of a region and expressing its minimal and maximal extension. The zones serve as a description and separation of the space that certainly belongs to the region and the space that is certainly outside. While [3] and [4] are mainly interested in classifications of topological relationships between vague regions for which a simple model is assumed, [10] proposes a model of complex vague regions with vague holes and focuses on their formal definition. Unfortunately, the three approaches are limited to "concentric" object models and have problems with geometric closure properties. The model described in [6] also pursues the exact model approach but is much more general and much simpler than the other approaches. It is a precursor of this paper and introduces the concept of vague regions.

3 What are Vague Spatial Objects?

The central idea of *vague spatial objects* is to base their definition on already well known, geometric modeling techniques. Our concept necessitates a general object model incorporating determinate spatial data types *point*, *line*, and *region* that are closed under (appropriately defined) geometric union, intersection, difference, and complement operations. Such *crisp* type systems have, e.g., been proposed in [9, 7], and we will take them and their corresponding formal definition for granted in this paper. Informally, these models consider a *point* object as a finite set of individual points, a *line* object as a finite set of disjoint *blocks* where each block represents a finite set of curves, and a *region* object as a finite set of disjoint, connected areal components (called *faces*) possibly with disjoint holes (see Figure 1).

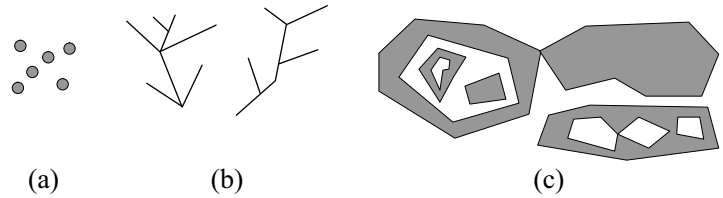

Fig. 1. Examples of a (complex crisp) point object (a), a (complex crisp) line object (b), and a (complex crisp) region object (c). Each collection of components forms a single crisp object.

As an illustrating example, we consider a homeland security scenario to introduce our concept for dealing with spatial vagueness and to demonstrate its usability. Secret services (should) have knowledge of the whereabouts of terrorists. For each terrorist, some of their refuges are precisely known, some are not and only conjectures. We can model these locations 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. Secret services are also interested in the routes a terrorist takes to move from one refuge to another. These routes can be modeled as *vague line* objects. Some

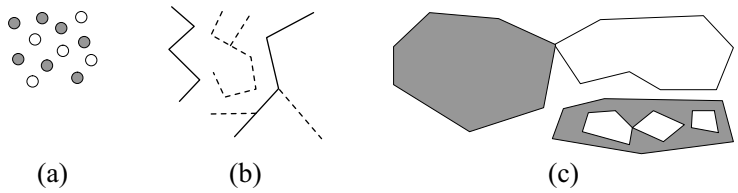

Fig. 2. Examples of a (complex) vague point object (a), a (complex) vague line object (b), and a (complex) vague region object (c). Each collection of components forms a single vague object.

routes, called *kernel line* objects, have been identified. Other routes can only be assumed to be taken by a terrorist; they are denoted as *conjecture line* objects. Knowledge about areas of terroristic activities is also important for secret services. From some areas it is well known that a terrorist operates in them; we call them *kernel region* objects. From other areas we can only assume that they are the target of terroristic activity; we denote them as *conjecture region* objects. Figure 2 gives some examples. Grey shaded areas, straight lines, and grey points indicate kernel parts; areas with white interiors, dashed lines, and white points refer to conjecture parts.

Based on this scenario and taking into account spatial vagueness, we are able to pose interesting queries. We can ask for the locations where any two terrorists have taken the same refuge. We can determine those terrorists that operated in the same area. We can compute the locations where routes taken by different terrorists crossed each other. Many further queries are possible. Vague concepts offer a greater flexibility for modeling properties of spatial phenomena in the real world than determinate concepts do. Still, vague concepts comprise the modeling power of determinate concepts as a special case.

In this sense, many scenarios can be found that could make meaningful use of the concept of vague spatial objects. They all have in common that a vague spatial object (e.g., a vague line) is described by a pair of two disjoint or adjacent crisp spatial objects (e.g., two crisp lines). The first crisp spatial object, called the *kernel part*, describes the determinate component of the vague object, that is, the component that definitely and always belongs to the vague object. The second crisp spatial object, called the *conjecture part*, describes the vague component of the vague object, 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. Or we could say that this is *unknown*.

4 A Generic Definition of Vague Spatial Data Types and Vague Spatial Set Operations

Based on the motivation in the previous section, we now give a formal definition of vague spatial data types (Section 4.1) and vague spatial set operations (Section 4.2). An interesting observation is that these definitions can be given in a generic manner, i.e., type-specific considerations are unnecessary. At the end, Section 4.3 introduces a few other generic operations as well as predicates.

4.1 Vague Spatial Data Types

For the definition of vague points, vague lines, and vague regions we make use of the data types *point* for crisp points, *line* for crisp lines, and *region* for crisp regions. All crisp spatial data types $\alpha \in \{point, line, region\}$ are assumed to have a complex inner structure as it has been defined, e.g., on the basis of point sets and point set topology in [7], or in concrete implementations in [8]. In particular, this means that a *point* object includes a finite number of single points, a *line* object is assembled from a finite number of curves, and a *region* object consists of a finite number of disjoint faces possibly containing a finite number of disjoint holes. Further, these types must be closed under the geometric set operations *union* ($\oplus : \alpha \times \alpha \to \alpha$), *intersection* ($\otimes : \alpha \times \alpha \to \alpha$), *difference* ($\ominus : \alpha \times \alpha \to \alpha$), and *complement* ($\sim : \alpha \to \alpha$). Each type α together with the operations $\oplus$ and $\otimes$ forms a boolean algebra. The identity of $\otimes$ is denoted by **1**, which corresponds to $\mathbb{R}^2$. The identity of $\oplus$ is presented by **0**, which corresponds to the empty spatial object (empty point set).

Syntactically, the extension of a crisp spatial data type to a corresponding vague type is given by a type constructor *v* as follows:

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

That is, each vague spatial data type is represented as a pair of corresponding crisp spatial data types. For example, for $\alpha = point$ we obtain $v(point) = point \times point$, which we also name *vpoint*. Accordingly, the data types *vline* and *vregion* are defined. For a vague spatial object $w = (k, c) \in v(\alpha)$, we call $k \in \alpha$ the *kernel part* of *w*, and $c \in \alpha$ denotes the *conjecture part* of *w*.

Semantically, the kernel part represents the determinate, crisp part of *w*, i.e., the area which definitely and always belongs to *w*. The conjecture part describes the vague part of *w*, i.e., the area for which we cannot say with any certainty whether it or parts of it belong to *w* or not. *Maybe* it or parts of it belong to *w*, *maybe* this is not the case. We could also say that this is *unknown* or *unclear* and thus a conjecture. To enable this intended semantics, we require:

$$\forall \alpha \in \{point, line, region\}\ \forall w = (k, c) \in v(\alpha) : disjoint(k, c)\ \vee\ meet(k, c)$$

The functions *disjoint* and *meet*, which operate on complex crisp spatial objects, denote generalized versions [12] of the well known topological predicates on simple spatial objects [5].

Let $points : v(\alpha) \to \mathbb{R}^2$ be an auxiliary function that yields the (unknown) point set of a vague spatial object. For an object $w = (k, c) \in v(\alpha)$ we can then conclude that

$$k \subseteq points(w) \subseteq k \oplus c$$

Hence, *k* can be regarded as a lower (minimal, guaranteed) approximation of *w* and $k \oplus c$ can be considered as an upper (maximally possible, speculative) approximation of *w*, which brings us near to rough set theory. Even if we do not know the exact point set of *w*, we assume and require that $points(w)$ is not arbitrary but compatible to α, i.e.,

$$points(w) \in \alpha \quad \text{and} \quad points(w) \ominus k \in \alpha$$

Using the characteristic function χ deciding about the existence or non-existence of an element in a set, we obtain $\chi(p) = 1$ for all $p \in k$, $\chi(p) = 0$ for all $p \in \mathbb{R}^2 - (k \cup c)$, $\chi(p) = 1 \vee \chi(p) = 0$ for all $p \in c - k$, and $\chi(p) = 1$ for all $p \in \mathit{points}(w) \in \alpha$. Note the deliberate use of set-theoretic operations. Especially common boundary points of k and c ($k \cap c \neq \varnothing$) are mapped to 1.

4.2 Vague Spatial Set Operations

The three vague geometric set operations **union**, **intersection**, and **difference** have all the same signature $v(\alpha) \times v(\alpha) \rightarrow v(\alpha)$. In addition, we define the operation **complement** with the signature $v(\alpha) \rightarrow v(\alpha)$. It is our goal and makes sense to define them in a type-independent and thus generic manner. In order to define them for two vague spatial objects u and w, it is helpful to consider meaningful relationships between the kernel part, the conjecture part, and the outside part of u and w. For each operation we give a table where a column/row labeled by k, c, or o denotes the kernel part, conjecture part, or outside part of u/w. Each entry of the table denotes a possible combination, i.e., 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 for the four vague geometric set operations.

The *union* (Table 1) of a kernel part with any other part is a kernel part since the union of two vague spatial objects asks for membership in either object and since membership is already assured by the given kernel part. Likewise, the union of two conjecture parts or the union of a conjecture part with the outside should be a conjecture part, and only the parts which belong to the outside of both objects contribute to the outside of the union.

The outside of the *intersection* (Table 1) is given by either region's outside because intersection requires membership in both regions. The kernel part of the intersection only contains components which definitely belong to the kernel parts of both objects, and intersections of conjecture parts with each other or with kernel parts make up the conjecture part of the intersection.

Obviously, the *complement* (Table 1) of the kernel part should be the outside, and vice versa. With respect to the conjecture part, anything inside the vague part of an object might or might not belong to the object. Hence, we cannot definitely say that the complement of the vague part is the outside. Neither can we say that the complement belongs to the kernel part. Thus, the only reasonable conclusion is to define the complement of the conjecture part to be the conjecture part itself.

The definition of *difference* (Table 1) between u and w can be derived from the definition of complement since it is equal to the intersection of u with the complement of v.

That is, removing a kernel part means intersection with the outside which always leads to outside, and removing anything from the outside leaves the outside part unaffected. Similarly, removing a conjecture part means intersection with the conjecture part and thus results in a conjecture part for kernel parts and conjecture parts, and removing the outside of w (i.e., nothing) does not affect any part of u.

Motivated by the just informally described, intended semantics for the four operations, we now define them formally. An interesting aspect is that these definitions can be based solely on already known crisp geometric set operations on well-understood exact spatial objects. Hence, we are able to give *executable specifications* for the vague geometric set operations. This means, if we have the implementation of a crisp spatial algebra available, we can directly *execute* the vague geometric set operations without being forced to design and implement new algorithms for them.

Let $u, w \in v(\alpha)$, and let u^k and w^k denote their kernel parts, u^c and w^c their conjecture parts, and u^o and w^o their outside parts. The outside of u, e.g., is defined as $u^o := \sim(u^k \oplus u^c)$. We define:

$$\begin{array}{rcl} u \textbf{ union } w & := & (u^k \oplus w^k, (u^c \oplus w^c) \ominus (w^k \oplus w^k)) \\ u \textbf{ intersection } w & := & (u^k \otimes w^k, (u^c \otimes w^c) \oplus (u^k \otimes w^c) \oplus (u^c \otimes w^k)) \\ u \textbf{ difference } w & := & (u^k \otimes (\sim w^k), (u^c \otimes w^c) \oplus (u^k \otimes w^c) \oplus (u^c \otimes (\sim w^k))) \\ \textbf{complement } u & := & (\sim u^k, u^c) \end{array}$$

We introduce juxtaposition as an abbreviating notation for the intersection of two crisp spatial objects and assign intersection higher associativity than union and difference. Hence, the above definition for u **difference** w could also be specified more concisely as $(u^k(\sim w^k), u^c w^c \oplus u^k w^c \oplus u^c(\sim w^k))$.

In a next step, we have to prove that the definitions realize the behavior specified in Table 1. For $z = u$ **union** w we have to show the three identities (1) $z^k = u^k w^k \oplus u^k w^c \oplus u^k w^o \oplus u^c w^k \oplus u^o w^k$, (2) $z^c = u^c w^c \oplus u^c w^o \oplus u^o w^c$, and (3) $z^o = u^o w^o$. The proof for (1) leverages that $\oplus$ is idempotent. We can therefore duplicate the first term $u^k w^k$. Then using the fact that $\otimes$ distributes over $\oplus$ we can factorize both u^k and w^k and obtain: $z^k = (u^k(w^k \oplus w^c \oplus w^o)) \oplus (w^k(u^k \oplus u^c \oplus u^o)))$. Since $w^k \oplus w^c \oplus w^o = \mathbf{1} = \mathbb{R}^2$ and $u^k \oplus u^c \oplus u^o = \mathbf{1}$, where $\mathbf{1}$ is the identity of $\otimes$, we get $z^k = (u^k \otimes \mathbf{1}) \oplus (w^k \otimes \mathbf{1}) = u^k \oplus w^k$, which is the definition of the kernel part of **union**.

For proving equation (2) we know that for $r, s \in \alpha$ holds: $r \oplus s = rs \oplus r(\sim s) \oplus (\sim r)s$. We can use this identity to rewrite the conjecture part definition as $u^c w^c \oplus u^c(\sim w^c) \oplus (\sim u^c)w^c \ominus (u^k w^k \oplus u^k(\sim w^k) \oplus (\sim u^k)w^k)$. Now we evaluate all complements by using that $\sim w^c = w^k \oplus w^o$ and $\sim w^k = w^c \oplus w^o$. This leads to $u^c w^c \oplus u^c(w^k \oplus w^o) \oplus (u^k \oplus u^o)w^c \ominus (u^k w^k \oplus u^k(w^c \oplus w^o) \oplus (u^c \oplus u^o)w^k)$. Applying distributivity of $\otimes$ we obtain: $u^c w^c \oplus u^c w^k \oplus u^c w^o \oplus u^k w^c \oplus u^o w^c \ominus (u^k w^k \oplus u^k w^c \oplus u^k w^o \oplus u^c w^k \oplus u^o w^k)$. In this term, only $u^c w^k$ and $u^k v^c$ appear in both parts of the difference; all other intersections to be subtracted have no effect at all since all intersections are pairwise disjoint. We obtain: $u^c w^c \oplus u^c w^o \oplus u^o w^c$ which corresponds exactly to the condition required for z^c.

For proving equation (3) we note that in a boolean lattice for $r, s \in \alpha$ holds: $\mathbf{1} \otimes s = s$, $\mathbf{1} \oplus s = \mathbf{1}$, and $\mathbf{1} = r \oplus (\sim r)$. Therefore, we know that $s = (r \oplus (\sim r))s = rs \oplus (\sim r)s$, and it follows that $r \oplus s = r \oplus rs \oplus (\sim r)s$. We also know that $r \oplus rs = r(\mathbf{1} \oplus s) = r$ so that $r \oplus s = r \oplus (\sim r)s$. Since $(\sim r)s$ is another way of denoting the difference $s \ominus r$, we get:

$r \oplus (s \ominus r) = r \oplus s$. Now we have by definition that $z^o = \sim(z^k \oplus z^c) = \sim(u^k \oplus w^k \oplus ((u^c \oplus w^c) \ominus (u^k \oplus w^k))) = \sim(u^k \oplus w^k \oplus u^c \oplus w^c)$. By commutativity and de Morgan's law this reduces to $(\sim(u^k \oplus u^c)) \otimes (\sim(w^k \oplus w^c))$ which is by the definition of complement equal to $u^o \otimes w^o$, the condition required for z^o.

Due to their lengthiness, we omit the proofs of the other three operations here whose correct behavior can be shown in a similar way.

4.3 Other Generic Operations and Predicates

Sometimes it is helpful to be able to explicitly deal with the kernel part or the conjecture part of a vague spatial object, or to swap its kernel part and conjecture part. For that purpose, we define the following generic operations for $u = (u^k, u^c) \in v(\alpha)$:

$$\begin{array}{rcl} \textbf{kernel}(u) & := & (u^k, \mathbf{0}) \\ \textbf{conjecture}(u) & := & (\mathbf{0}, u^c) \\ \textbf{invert}(u) & := & (u^c, u^k) \end{array}$$

All three operations have the signature $v(\alpha) \to v(\alpha)$[1]. The **kernel** operation especially facilitates computations exclusively with the exact part of a vague spatial object u, because the vague spatial operations, applied to vague spatial objects with an empty conjecture part, behave exactly like the corresponding crisp spatial operations. This can be easily seen from the definitions and is intended. Consequently, crisp spatial objects are a special case of their corresponding vague counterparts. The **conjecture** operation allows one to focus on the unclear, indeterminate part of u. The **invert** operation changes the role of kernel part and conjecture part of u.

It is also possible to identify generic predicates. The most obvious ones are $=, \neq$: $v(\alpha) \to \mathit{bool}$. For $u, w \in v(\alpha)$, they are defined as follows:

$$\begin{array}{rcl} u = w & := & (u^k = w^k \ \wedge \ u^c = w^c) \\ u \neq w & := & (u^k \neq w^k \ \vee \ u^c \neq w^c) \end{array}$$

5 Type-Specific Spatial Operations

In this section we describe a few operations requiring particular types as operands. The operation **boundary** with the signature *vregion* $\to$ *vline* allows us to extract the boundary of a vague region as a vague line. Its definition requires the crisp operation *boundary* : *region* $\to$ *line* which determines the boundary of a crisp region as a crisp line. Vice versa, the operation **interior** with the signature *vline* $\to$ *vregion* determines faces in a vague line object and transforms them into a vague region. Its definition requires the crisp operation *interior* : *line* $\to$ *region* which calculates the faces of a crisp line and collects them into a crisp region. For $r \in$ *vregion* in particular $\textbf{interior}(\textbf{boundary}(r)) = r$ holds, i.e., the operations **interior** and **boundary** are inverse. The operation **vertices**

[1] To really connect vague and crisp spatial data types and to map the kernel part or the conjecture part of a vague spatial object to the corresponding crisp spatial object, one could define projection functions $\pi_k, \pi_c : v(\alpha) \to \alpha$ with $\pi_k(u) = u^k$ and $\pi_c(u) = u^c$.

with the signatures *vline* $\rightarrow$ *vpoint* and *vregion* $\rightarrow$ *vpoint* collects the end points of the segments of a vague line and the end points of the segments of the boundary of a vague region respectively. Its definition is based on the crisp operation *vertices* with the signatures *line* $\rightarrow$ *point* and *region* $\rightarrow$ *point*. Let further $l \in$ *vline*. We define:

$$\begin{aligned}
\textbf{boundary}(r) &:= (\mathit{boundary}(r^k), \mathit{boundary}(r^c)) \\
\textbf{interior}(l) &:= (\mathit{interior}(l^k), \mathit{interior}(l^c)) \\
\textbf{vertices}(l) &:= (\mathit{vertices}(l^k), \mathit{vertices}(l^c)) \\
\textbf{vertices}(r) &:= (\mathit{vertices}(r^k), \mathit{vertices}(r^c))
\end{aligned}$$

We have so far described the **intersection** operation only for two vague spatial objects of the same type. We extend this definition now to all mixed type combinations with the signatures *vpoint* $\times$ *vline* $\rightarrow$ *vpoint*, *vpoint* $\times$ *vregion* $\rightarrow$ *vpoint*, *vline* $\times$ *vregion* $\rightarrow$ *vline*, *vregion* $\times$ *vpoint* $\rightarrow$ *vpoint*, *vregion* $\times$ *vline* $\rightarrow$ *vline*, and *vline* $\times$ *vpoint* $\rightarrow$ *vpoint*. For this purpose, we generalize the crisp intersection operation $\otimes$ to all corresponding crisp variants of the just mentioned signatures. These crisp variants are well defined. The already known definition for **intersection** can then also be applied in case of the mixed type combinations.

The operation **common_border** incorporates a special kind of intersection. It computes the shared boundary of two extended vague spatial objects as a vague line. Its signatures are *vline* $\times$ *vline* $\rightarrow$ *vline*, *vline* $\times$ *vregion* $\rightarrow$ *vline*, *vregion* $\times$ *vline* $\rightarrow$ *vline*, and *vregion* $\times$ *vregion* $\rightarrow$ *vline*. The definitions can be reduced to the well defined crisp spatial operation *common_border* : *line* $\times$ *line* $\rightarrow$ *line* which computes the common line parts of two crisp lines. We define for $l, m \in$ *vline* and $r, s \in$ *vregion*:

$$\begin{aligned}
\textbf{common_border}(l, m) &:= (\mathit{common_border}(l^k, m^k), \mathit{common_border}(l^c, m^c)) \\
\textbf{common_border}(l, r) &:= \textbf{common_border}(l, \textbf{boundary}(r)) \\
\textbf{common_border}(r, l) &:= \textbf{common_border}(l, r) \\
\textbf{common_border}(r, s) &:= \textbf{common_border}(\textbf{boundary}(r), \textbf{boundary}(s))
\end{aligned}$$

Finally, we discuss the vague operator **convex_hull** : *vpoint* $\rightarrow$ *vregion*. A subset S of the plane is called *convex* if and only if for any pair of points $p, q \in S$ the line segment between p and q is completely contained in S. The well known crisp operation *convex_hull* : *point* $\rightarrow$ *region* computes the smallest convex region that contains all points of a given point object. We define the convex hull of a vague point $p \in$ *vpoint* as

$$\textbf{convex\_hull}(p) := (\mathit{convex\_hull}(p^k), \mathit{convex\_hull}(p^k \oplus p^c) \ominus \mathit{convex\_hull}(p^k))$$

The smallest, guaranteed convex hull of p is given by the convex hull of its kernel part. If all points of the conjecture part of p lie inside or on the boundary of the convex hull of its kernel part, the conjecture part of the resulting vague region is $\mathbf{0}$, i.e., the empty region object. Otherwise, the convex hull involving all points both from the kernel part and the conjecture part will be larger than the convex hull of the kernel part.

6 Vague Topological Predicates

Topological predicates provide information about the relative position of spatial objects towards each other. The result type of *vague topological predicates*[2] is a value of a new vague data type named $vbool = \{true, false, maybe\}$. That is, we use a three-valued logic as the range of these predicates. The definition of the *vague logical operators* **and**, **or**, and **not** (Table 2) parallels the definition of the vague spatial operations in Section 4.2 (t, f, and m are used as abbreviations for *true*, *false*, *maybe*).

and	t	m	f
t	t	m	f
m	m	m	f
f	f	f	f

or	t	m	f
t	t	t	t
m	t	m	m
f	t	m	f

not	t	m	f
	f	m	t

Table 2. Vague logical operators (three-valued logic).

We first consider a generic definition of the vague **inside** predicate which has the signatures $vpoint \times \alpha \to vbool$ with $\alpha \in \{vpoint, vline, vregion\}$, $vline \times \beta \to vbool$ with $\beta \in \{vline, vregion\}$, and $vregion \times vregion \to vbool$. Let u be the first operand and w be the second operand according to the signatures. Then their definition is as follows:

$$u \textbf{ inside } w := \begin{cases} true & \text{if } u^k \oplus u^c \subseteq w^k \\ false & \text{if } u^k \not\subseteq w^k \oplus w^c \\ maybe & \text{otherwise} \end{cases}$$

Hence, we can safely say that u **inside** w holds if everything of u (i.e., kernel part and conjecture part) is inside the kernel part of w. If this is not the case, we cannot simply conclude that u **inside** w is *false* since this requires definite knowledge about a part of u being outside any part of v. In other words, if we can exclude *true* as the predicate result and if $u^k \subseteq w^k \oplus w^c$, we are not sure about insideness, and we define u **inside** w as *maybe*.

Next we present a generic definition of the vague **intersects** predicate which has the signature $\alpha \times \alpha \to vbool$ with $\alpha \in \{vpoint, vline, vregion\}$. We define for $u, w \in \alpha$:

$$u \textbf{ intersects } w := \begin{cases} true & \text{if } u^k w^k \neq \mathbf{0} \\ false & \text{if } u^k w^k \oplus u^c w^c \oplus u^k w^c \oplus u^c w^k = \mathbf{0} \\ maybe & \text{otherwise} \end{cases}$$

The definition means that the predicate holds if the kernel parts of u and w intersect. This is true independent from the value of $u^c w^c$. Likewise, if $u^k \oplus u^c$ and $w^k \oplus w^c$ are disjoint, we can definitely say that u and w do not intersect at all. However, if $u^k w^k = \mathbf{0}$ and $u^c w^c \neq \mathbf{0}$, we cannot be sure about the intersection of u and w and let the predicate return the value *maybe*.

[2] In this paper, we deliberately omit the discussion of a "complete" collection of vague topological predicates for which we are currently designing a comprehensive concept that will be presented in a future paper.

The predicate **on_border_of** : $vpoint \times \alpha \rightarrow vbool$ with $\alpha \in \{vline, vregion\}$ checks whether a vague point is located on a vague line and a vague region respectively. Let $p \in vpoint$, $l \in vline$, and $r \in vregion$. We use the operation **boundary** defined in Section 5 to compute the boundary of a vague region as a vague line and define:

$$\begin{array}{lcl} \textbf{on\_border\_of}(p,l) & := & \textbf{inside}(p,l) \\ \textbf{on\_border\_of}(p,r) & := & \textbf{inside}(p,\textbf{boundary}(r)) \end{array}$$

The predicate **border_in_common** has the signatures $vline \times vline \rightarrow vbool$, $vline \times vregion \rightarrow vbool$, $vregion \times vline \rightarrow vbool$, and $vregion \times vregion \rightarrow vbool$. It determines whether two extended vague spatial objects share a common border. Let $l, m \in vline$ and $r, s \in vregion$. Then

$$\begin{array}{rcl} \textbf{border\_in\_common}(l,m) & := & \begin{cases} true & \text{if } l^k m^k \neq \mathbf{0} \ \wedge \ l^k m^k \in vline \\ false & \text{if } l^k m^k \oplus l^c m^c \oplus l^k m^c \oplus l^c m^k = \mathbf{0} \ \vee \\ & \quad l^k m^k \oplus l^c m^c \oplus l^k m^c \oplus l^c m^k \notin vline \\ maybe & \text{otherwise} \end{cases} \\ \textbf{border\_in\_common}(l,r) & := & \textbf{border\_in\_common}(l,\textbf{boundary}(r)) \\ \textbf{border\_in\_common}(r,l) & := & \textbf{border\_in\_common}(l,r) \\ \textbf{border\_in\_common}(r,s) & := & \textbf{border\_in\_common}(\textbf{boundary}(r),\textbf{boundary}(s)) \end{array}$$

7 Embedding Vague Spatial Data Types into Query Languages

A few examples shall demonstrate the integration of vague spatial concepts into the relational data model and the query language VSQL (vague SQL). We do not give a full description of VSQL. Vague spatial data types are embedded as *abstract data types* into a relational schema and may be used as attribute types like standard types. That is, their internal structure is hidden from the user and can only be accessed by operations. For instance, we may specify the relation weather(climate: *string*, region: *vregion*), where the column named *region* contains vague region values for various climatic conditions given by the column *climate*, and the relation soil(quality: *string*, region: *vregion*) describing the soil quality for certain regions.

If we want to find out all regions where lack of water is a problem for cultivation, or if we are interested in bad soil regions as a hindrance for cultivation, we can pose the following queries:

select region **from** weather **where** climate = “dry”
select region **from** soil **where** quality = “bad”

Note that the result of both queries is a set of vague regions.

If we want to find out about vague regions where cultivation is impossible due to a lack of water or bad soil quality, we ask for the union of two vague region sets. Thus, we first have to cast the sets into single *vregion* objects. We therefore use the built-in, overloaded aggregation function **sum** which, when applied to a set of vague regions, aggregates this set by repeated application of **union**. We can determine regions where cultivation is impossible by:

(**select sum**(region) **from** weather **where** climate = “dry”)
union
(**select sum**(region) **from** soil **where** quality = “bad”)

As a next example, we take the self-explaining relations pollution(type: *string*, region: *vregion*) and areas(use: *string*, region: *vregion*). Pollutions are nowadays a central ecological problem and cause an increasing number of environmental damages. Important examples are air pollution and oil soiling. Pollution control institutions, ecological researchers, and geographers, usually use maps for visualizing the expansion of pollution. We can ask, for example, for inhabitable areas which are air polluted (where the kernel part of an air pollution denotes heavily polluted areas and the conjecture part only gives slightly polluted regions) as follows:

select sum(pollution.region) **intersection sum**(areas.region)
from pollution, areas
where area.use = “inhabited” **and** pollution.type = “air”

Then the kernel part of the result consists of inhabited regions which are heavily polluted, and the conjecture part consists (a) of slightly polluted inhabited regions, (b) of heavily polluted regions which are only partially inhabited, and (c) of slightly polluted and partially inhabited regions.

If we want to reach all people who live in heavily polluted areas, we need the kernel part of the intersection together with conjecture part (b) of the intersection. How can we get this from the above query? The trick is to force boundary parts (a) and (c) to be empty by restricting pollution areas to their kernel region:

select kernel(**sum**(pollution.region)) **intersection sum**(areas.region)
from . . .

A slightly different query is to find out all areas where people are definitely or possibly endangered by pollution. Of course, we have to use an intersection predicate. More precisely, we want to find those areas for which **intersects** either yields *true* or *maybe*. For this purpose we can prefix any predicate with **maybe** which causes the predicate to fail only if it returns *false*. Technically **maybe** turns a *maybe* value into *true*. So the query is:

select areas.name
from pollution, areas
where area.use = “inhabited” **and** pollution.region **maybe intersects** areas.region

This query is an example of a *vague spatial join*.

The following example is based on the the self-explaining relations resources(kind: *string*, region: *vregion*) and nature(type: *string*, region: *vregion*) and describes a situation which stresses the conflicting interests of economy and ecology. Assume on the one hand areas of animal species and plants that are worth being protected (nature reserves and national parks are the kernel parts) and on the other hand mineral resources the mining of which prospects high profits. An example for forming a difference of vague regions is a query which asks for mining areas that do not affect the living space of endangered species.

(**select sum**(region) **from** resources **where** kind = “mineral”)
difference
(**select sum**(region) **from** nature **where** type = “endangered”)

The kernel part of the result describes regions where mining should be allowed. The conjecture part consists (a) of regions where mineral resources are uncertain and (b) of resource kernels that lie in (non-kernel) regions hosting endangered species. Since national parks are generally protected by the government, it is especially regions (b) conservationists should carefully observe. We can determine these regions by:

(**select kernel**(**sum**(region)) **from** resources **where** kind = “mineral”)
intersection
(**select conjecture**(**sum**(region)) **from** nature **where** type = “endangered”)

The result is a vague region with an empty kernel part and a conjecture part that just consists of the intersection of the mineral resource kernel part and the endangered nature conjecture part.

Next, we consider an example from biology and assume living spaces of different animal species stored in a relation animals(name: *string*, region: *vregion*). The kernel part describes places where they normally live, and the conjecture part describes regions where they can be found occasionally(e.g., to hunt for food or to migrate from one kernel area to another through a corridor). We can search for pairs of species which share a common living space. This query asks for regions which have a non-empty intersection kernel:

select A.name, B.name
from animals **as** A, animals **as** B
where A.region **intersects** B.region

8 Conclusions and Future Work

We have defined a data model of points, lines, and regions that is capable of describing many different aspects of spatial vagueness. It is a canonical extension of determinate spatial data models, which facilitates the treatment of vague and exact objects in one model. Since our approach is based on exact spatial modeling concepts, it allows to build upon existing work and simplifies many definitions. In particular, we can leverage already existing implementations of crisp spatial type systems to realize vague spatial objects with only minimal effort by executable specifications.

Currently we are working on a comprehensive concept for *vague topological predicates* and also deal with *vague numerical operations*. Implementation tuning is another topic.

References

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

2. P. A. Burrough and A. U. Frank, editors. *Geographic Objects with Indeterminate Boundaries*. GISDATA Series, vol. 2. Taylor & Francis, 1996.
3. E. Clementini and P. Di Felice. *An Algebraic Model for Spatial Objects with Indeterminate Boundaries*, pp. 153–169. In Burrough and Frank [2], 1996.
4. A. G. Cohn and N. M. Gotts. *The 'Egg-Yolk' Representation of Regions with Indeterminate Boundaries*, pp. 171–187. In Burrough and Frank [2], 1996.
5. M. J. Egenhofer, A. Frank, and J. P. Jackson. A Topological Data Model for Spatial Databases. *1st Int. Symp. on the Design and Implementation of Large Spatial Databases*, LNCS 409, pp. 271–286. Springer-Verlag, 1989.
6. M. Erwig and M. Schneider. Vague Regions. *5th Int. Symp. on Advances in Spatial Databases*, LNCS 1262, pp. 298–320. Springer-Verlag, 1997.
7. R. H. Güting and M. Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal*, 4:100–143, 1995.
8. R.H. Güting, T. de Ridder, and M. Schneider. Implementation of the ROSE Algebra: Efficient Algorithms for Realm-Based Spatial Data Types. *Int. Symp. on Advances in Spatial Databases*, 1995.
9. OGC Abstract Specification. OpenGIS Consortium (OGC), 1999. URL: http://www.opengis.org/techno/specs.htm.
10. M. Schneider. *Modelling Spatial Objects with Undetermined Boundaries Using the Realm/ROSE Approach*, pp. 141–152. In Burrough and Frank [2], 1996.
11. M. Schneider. Uncertainty Management for Spatial Data in Databases: Fuzzy Spatial Data Types. *6th Int. Symp. on Advances in Spatial Databases*, LNCS 1651, pp. 330–351. Springer-Verlag, 1999.
12. M. Schneider. A Design of Topological Predicates for Complex Crisp and Fuzzy Regions. *Int. Conf. on Conceptual Modeling*, pp. 103–116, 2001.
13. M. Worboys. Computation with Imprecise Geospatial Data. *Computational, Environmental and Urban Systems*, 22(2):85–106, 1998.