Information Systems 35 (2010) 111–138

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

VASA: An algebra for vague spatial data in databases ☆

Alejandro Pauly, Markus Schneider *

University of Florida, Department of Computer and Information Science and Engineering, Gainesville, FL 32611, United States

ARTICLE INFO

Article history:
Received 6 February 2008
Received in revised form
22 April 2009
Accepted 26 May 2009
Recommended by: D. Shasha and Masatoshi

Keywords:
Vague spatial data types
Vague topological predicates
Characterization predicates
Executable specifications
Binary constraint networks

ABSTRACT

Many geographical applications deal with objects in space that cannot be adequately described by determinate, crisp spatial concepts because of their intrinsically indeterminate and vague nature. Geographical information systems and spatial database systems are currently unable to cope with this kind of data. To support the efficient representation, querying, and manipulation of vague spatial data in a database context, we present a formal data model called *vague spatial algebra* (*VASA*). This algebra comprises a set of *vague spatial data types* for *vague points*, *vague lines*, and *vague regions* together with a comprehensive collection of *vague spatial operations* and *vague topological predicates*. One of VASA's main benefits is that its formal framework is based on well known, general, and exact models of crisp spatial data types. This enables an exact definition of the vague spatial model since we can build upon an already existing theory of spatial data types. In particular, crisp spatial data types turn out to be a special case of their vague counterparts. In addition, our approach enables *executable specifications* for the operations, which can be immediately used as implementations. The article offers a precise and conceptually clean foundation for implementing a DBMS extension for vague spatial data and demonstrates the embedding of these new data types as attribute data types in a database schema as well as the incorporation of vague spatial operations and predicates into queries formulated in an SQL-like query language. All concepts have been verified in a prototype implementation.

1. Introduction

In the literature about spatial database systems and geographical information systems (GIS) the general opinion prevails that special data types are necessary to adequately model geometry and to efficiently represent geometric data in database systems. These data types are commonly denoted as *spatial data types* [38] such as *point*, *line*, and *region*. We speak of *spatial objects* as occurrences of spatial data types. So far, the mapping of spatial phenomena of the real world leads almost exclusively to precisely defined spatial objects. 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. This leads to *exact object models*. Examples are especially man-made spatial objects representing engineered artifacts (like monuments, highways, buildings, bridges) and predominantly immaterial spatial objects exerting social control (like countries, districts, and land parcels with their political, administrative, and cadastral boundaries). We denote this kind of entities as *crisp* or *determinate spatial objects*.

But for many geometric applications, the mapping into crisp spatial objects is an insufficient abstraction process since many geographic objects show the inherent feature of *spatial vagueness* or *spatial indeterminacy* [4]. Current GIS and spatial database systems are not capable of

☆ This work was partially supported by the National Science Foundation (NSF) under Grant number NSF-CAREER-IIS-0347574.

* Corresponding author. Tel.: +352 392 2697.
E-mail addresses: apauly@cise.ufl.edu (A. Pauly), mschneid@cise.ufl.edu (M. Schneider).

0306-4379/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2009.05.003

supporting applications based on vague geometric data. In these applications, the positions of points are not exactly known, the locations and routes of lines are unclear, and regions do not have sharp boundaries, or their boundaries cannot be precisely determined. Examples are natural phenomena (like soil quality, vegetation, oceans, valleys, mountains, oil fields, biotopes, deserts, clouds, sandbanks), cultural phenomena (like a Rhaeto-Romanic language speaking area) and social phenomena (like population density, unemployment rate, terrorists' refuges, and escape routes). We denote this kind of entities as *vague* or *indeterminate spatial objects*.

The goal of this article is the design, definition, and implementation of an object model for two-dimensional, vague spatial data, called *vague spatial algebra* (*VASA*), such that a DBMS data model and its query language are enabled to handle vague geometries. Our algebra (type system) framework provides a set of *vague spatial data types* for *vague points*, *vague lines*, and *vague regions* together with a comprehensive collection of *vague spatial operations* and *vague topological predicates*. All data types are closed under operations, and all data types and a number of operations are defined generically, i.e., without taking into account type-specific properties and constraints.

In contrast to other approaches that are based, e.g., on fuzzy set theory or probability theory, our approach is conservative in the sense that it makes use of a three-valued (and not a multi-valued) logic; this gives us the chance to learn about spatial vagueness. Further, it provides us with the advantage of conceptual simplicity, ease of understanding, ease of use, ease of implementation, and efficiency. One of VASA's main benefits is that its formal framework is based on well known, "traditional", general, and exact models of crisp spatial data types. That is, vague spatial data types and operations are based on their crisp counterparts and can be expressed by them so that we obtain *executable specifications*. These specifications can be directly used as implementations and minimize the needed implementation effort. Moving from an exact to a vague domain does not necessarily invalidate conventional (computational) geometry; in our case, it is merely an extension. VASA offers a precise and conceptually clean foundation for implementing a DBMS extension for vague spatial data. We propose to design vague spatial data types as *abstract data types* such that they can be leveraged as attribute data types in a database schema and to incorporate vague operations and predicates into the DBMS query language. This enables their use as column types in conventional relational DBMSs, or they may be integrated into object-oriented or object-relational DBMSs. In general, it is possible for a user or a third-party developer to implement abstract data types based on this article's definitions in an extensible DBMS. By posing example queries, we demonstrate the expressiveness of VASA and show the integration of its operations and predicates into an SQL-like query language. Seen from this standpoint, we extend, rather than replace, current spatial type systems in spatial database systems and GIS. In particular, crisp spatial data types turn out to be a special case of their vague counterparts. The restriction that has to be accepted for all these benefits is that our model cannot capture all potential applications dealing with vague spatial data. Fuzzy and probabilistic approaches allow the distinction and consideration of different degrees of spatial vagueness due to their multi-valued logic. However, this requires more and especially precise knowledge about uncertainty distributions that is frequently not available. Other problems of these approaches are lacking design concepts (e.g., for topological predicates) and lacking implementation concepts (e.g., with respect to data structures). This is the reason why we see VASA as a good starting point with many practical applications and as the basis for further research.

Section 2 discusses related work. Section 3 begins with an informal introduction of vague spatial objects and motivates them by application examples. Then a definition of vague spatial data types is given. In Section 4, we focus on vague spatial operations. This includes set operations, type-dependent operations, and numerical operations. In Section 5, we present our concept and definition of vague topological predicates. In Section 6, we address VASA's implementation concept, illustrate the embedding of vague spatial data types into an SQL-like query language, and give example queries. Finally, Section 7 draws some conclusions and addresses future work.

2. Related work

In this section, we discuss related work that is relevant for the design and construction of VASA. In Section 2.1 we deal with crisp, determinate spatial data types, operations, and topological relationships that serve as the basis of their vague counterparts. Section 2.2 delineates the phenomenon of spatial vagueness in general and gives a classification of current models for representing it. Finally, Section 2.3 gives an overview of the existing exact models of vague spatial objects.

2.1. Crisp spatial data types, operations, and topological relationships

In the spatial database and GIS community, *spatial data types* like *point*, *line*, or *region* have found wide acceptance as fundamental abstractions for modeling the structure of geometric entities, their relationships, properties, and operations. They form the basis of a large number of data models and query languages for processing spatial data and have gained access into commercial software products. The literature distinguishes *simple* spatial data types (e.g., [11,17,30]) and *complex* spatial data types (e.g., [5,18,21,28,38,40,47]), depending on the spatial complexity they are able to model. Simple spatial data types (Fig. 1(a)–(c)) only provide simple object structures like single points, continuous lines, and simple regions. However, from an application perspective, they are insufficient to cope with the variety and complexity of geographic reality. From a formal perspective, they are not closed under the geometric set operations *intersection*, *union*, and *difference*. Complex spatial data types (Fig. 1(d)–(f)) solve these problems. They provide

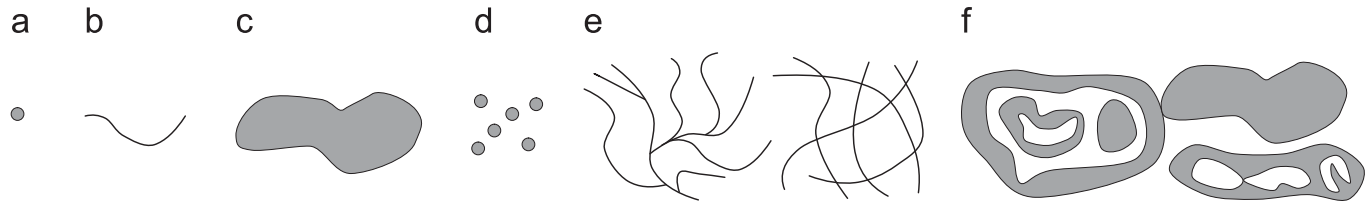


Fig. 1. Examples of a simple point object (a), a simple line object (b), a simple region object (c), a complex point object (d), a complex line object (e), and a complex region object (f).

universal and versatile spatial objects with multiple components, permit regions with holes, are closed under geometric set operations, and form the basis of our vague spatial data types.

A special emphasis of (even interdisciplinary) spatial research has been put on the exploration of *topological relationships* (e.g., *overlap*, *inside*, *disjoint*, *meet*) between crisp spatial objects in the two-dimensional space. They describe purely qualitative properties that characterize the relative positions of spatial objects towards each other and that are preserved under certain continuous transformations including all affine transformations. From a database and GIS perspective, their investigation has been motivated by the necessity of formally defined topological predicates as filter conditions for spatial selections and spatial joins in spatial query languages and as a support for spatial data retrieval and analysis tasks.

An important approach for characterizing them rests on the well known 9-*intersection model* (e.g., [12]), which employs point set theory and point set topology. The model is based on the nine possible intersections of the boundary (∂A), interior (A°), and exterior (A^-) of a spatial object[1] A with the corresponding components of another object B. The nine intersections are usually represented by a 3×3-matrix called the 9-*intersection matrix* (Fig. 14(a)). Each intersection is tested with regard to the topologically invariant criteria of emptiness and non-emptiness. $2^9 = 512$ different configurations are possible from which only a certain subset makes sense depending on the combination of spatial data types just considered. A complete collection of mutually exclusive topological relationships can be determined for each combination of simple spatial data types [13] and, as a generalization, for each combination of complex spatial data types [40]. A unique (9-intersection) *matrix number* has been assigned to each topological relationship [40]. Table 1 shows the numbers of topological predicates for each type combination.

Implementations of complex spatial data types are available in a number of commercial and public domain software systems. Examples are ESRI's Spatial Database Engine (ArcSDE) [15], the Informix Geodetic DataBlade [19], Oracle Spatial [29], DB2's Spatial Extender [20], PostGIS [35], and the JTS Topology Suite [43]. All these implementations (at least partially) support the specifications of the Open Geospatial Consortium [28,21] and can be used as a basis for implementing VASA (see Section 6.1).

Table 1
Numbers of topological predicates between two simple spatial objects (a) and between two complex spatial objects (b).

	Simple point	Simple line	Simple region
(a)			
Simple point	2	3	3
Simple line	3	33	19
Simple region	3	19	8
	Complex point	**Complex line**	**Complex region**
(b)			
Complex point	5	14	7
Complex line	14	82	43
Complex region	7	43	33

2.2. A classification of models for vague spatial objects

So far, spatial data modeling has represented spatial objects as entities with sharply determined boundaries emphasizing abrupt changes of spatial phenomena. The assumption of crisp boundaries harmonizes very well with the internal representation and processing of spatial objects in a computer which require precise and unique structures. Hence, in the past, there has been a tendency to force geographical reality into determinate spatial objects. In practice, however, there is no apparent reason for the whole contour of a line or the boundary of a region to be sharp. Numerous geographical application examples illustrate that the extent and the boundaries of spatial objects can be indeterminate (see, e.g., [1,3,4,22,23,44]). For instance, boundaries of geological, soil, and vegetation units are often crisp in some places and vague in others; concepts like the "Indian Ocean", "South England", or a biotope are intrinsically vague.

In the real, non-artifactual world, we can essentially find two categories of indeterminate boundaries: sharp boundaries whose position and shape are unknown or cannot be measured precisely, and boundaries which are not well defined or which are useless (e.g., between a mountain and a valley) and where essentially the

[1] In case of a *point* object, its boundary is empty, and its interior is the finite set of points defining it. In case of a *line* object, its boundary consists of all end points from which exactly one curve emanates, and its interior is the object without its end points. In case of a *region* object, its boundary comprises the contours of all components and holes, and its interior is the object without its boundary. In all cases, the exterior of a spatial object is the Euclidean plane without the object. See [40] for a formal definition.

topological relationship between spatial objects is of interest. Spatial objects with indeterminate boundaries are difficult to represent and are so far not supported in spatial database systems and GIS. *Spatial vagueness* describes the feature of a spatial object that we cannot be sure whether certain components belong completely or partially to the object or not. According to the two categories of boundaries, mainly two kinds of spatial vagueness can be identified: spatial uncertainty and spatial fuzziness. *Spatial uncertainty* is traditionally equated with randomness and chance occurrence and relates either to a lack of knowledge about the position and shape of a spatial object with an existing, real boundary (*positional* uncertainty) or to the inability of measuring such an object precisely (*measurement* uncertainty). *Spatial fuzziness* is an intrinsic feature of a spatial object itself and describes the vagueness of such an object which certainly has an extent but which inherently cannot or does not have a precisely definable boundary.

At least four alternatives have been proposed as general design methods of spatial vagueness: (1) *fuzzy models* (e.g., [1,3,9,23,39,45]) which are all based on fuzzy set theory and predominantly model fuzziness, (2) *probabilistic models* [2,3,16,41,48] which are based on probability theory and predominantly model positional and measurement uncertainty, (3) *rough models* [46] which are based on rough set theory and model both uncertainty and fuzziness but in a restricted way, and (4) *exact models* [6–8,14,31,37] which transfer type systems and concepts of spatial objects with sharp boundaries to objects with unclear boundaries and which model both uncertainty and fuzziness but in a restricted way.

Probability theory is able to represent uncertainty and defines the probability of an entity in a set by a statistically defined probability function. It deals with the *expectation* of a future event, based on something known now. Examples are the uncertainty about the spatial extent of regions defined by some property such as temperature, or the water level of a lake.

Fuzzy set theory deals only with fuzziness, i.e., it describes the *admission of the possibility* (given by a so-called *membership function*) that an individual is a member of a set or that a given statement is true. Hence, the vagueness represented by fuzziness is not the uncertainty of expectation. It is the vagueness resulting from the imprecision of the meaning of a concept. Examples of fuzzy spatial objects include mountains, valleys, biotopes, and oceans, which cannot be rigorously bounded by a sharp line.

Both theories require much more knowledge in terms of probability and membership functions than we assume or have in VASA. Hence, they allow a much more fine-grained modeling of vague spatial objects than VASA, which only assumes a three-valued logic. But drawbacks of both theories are the necessary successful and precise determination of appropriate probability and membership functions and the computationally much higher cost for the design and implementation of data structures and algorithms.

Rough set theory deals with lower and upper approximations of objects. This is similar to our approach but the terminology and formal framework used are rather different from ours. The lower approximation is always a subset of the upper approximation; we model known and vague parts as disjoint spatial objects.

Exact models, to which VASA belongs, extend data models, type systems, and concepts for crisp spatial objects to vague spatial objects. We will discuss the existing approaches in more detail in the next subsection.

2.3. Exact models for vague spatial objects

The attractiveness and benefit of exact models for modeling vague spatial objects rests on the wide range of existing definitions, techniques, data structures, and algorithms for crisp spatial objects that need not be redeveloped but only modified and extended, or simply used. The first models have been proposed for simplified vague regions and employed some kind of *zone* concept, either on the basis of simple regions without holes [6,8] or on the basis of simple regions with holes [37]. 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, the space that possibly belongs to the region, and the space that is certainly outside the region. They are modeled by a crisp simple region which represents the area that definitely belongs to the vague region and that is located in another larger crisp simple region. The geometric difference between the larger crisp region and the smaller one is considered to be the vague zone in which the actual boundary is situated. These simplified vague regions are called *regions with broad boundaries* in [6] and *egg-yolk regions* in [8]. The model in [37] proposes a model of simple vague regions with vague holes and focuses on their formal definition. Unfortunately, the three approaches do not satisfy closure properties. Two precursors of this article introduce our core concept of *vague regions* [14] and an extension to *vague points* and *vague lines* [31]. They are generalizations of the object specifications in [6,8,37] and satisfy closure properties. Their definitions leverage complex spatial objects and make the concept of vague spatial objects much more expressive and their handling much simpler. In this article, we further generalize the definitions of these data types and their operations and predicates, and, in particular, define precisely their semantics.

The primary goal in [6,8] is the exploration of topological predicates for simple regions with broad boundaries and egg-yolk regions, respectively. Their studies result in 44 and 46 different topological relationships, respectively. The result in [6] rests on considering the nine intersections of the interior, broad boundary, and exterior of a simple region with broad boundaries with the corresponding components of another simple region with broad boundaries. The work in [8] employs spatial logic. The approach in [6] is extended to *composite regions with broad boundaries* in [7] and leads to 56 topological relationships. Our concept of identifying topological predicates (see Section 5) is quite different since our

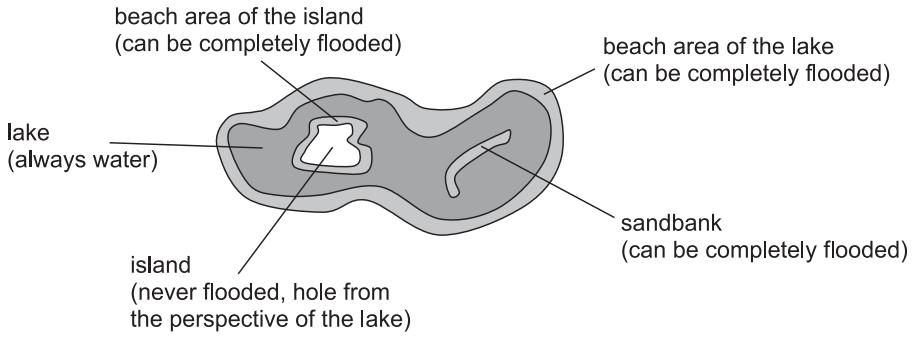


Fig. 2. The extent of a lake depending on the degree of evaporation and on the amount of precipitation.

object model is much more general. We allow complex regions and do not impose the constraint that the determinate region must be a subset of the indeterminate region. We even require that the determinate region and the indeterminate region are either disjoint or that they meet. We have described preliminary designs of vague topological predicates in [32–34]. In [32,33] we make use of so-called *cancelation rules* that filter out invalid constellations that definitely cannot lead to vague topological predicates. However, the problem is here to determine a complete set of cancelation rules. Our work in [34] sketches another idea that leverages *binary constraint networks* (BCN) for a definition of vague topological predicates. In this article, we largely extend, formalize, and generalize this approach to all combinations of vague spatial data types.

3. Vague spatial data types

Our type system *VASA* (short hand for *vague spatial algebra*) comprises a set of vague spatial data types together with a set of operations between these types.[2] A few operations are applicable to several types and are thus overloaded. We do not aim at developing a type system with a "complete" set of data types and operations (topological predicates are an exception (see Section 5)). It is common consensus in the spatial database community that this is impossible since always new data types, operations, and predicates can be designed. It is then more favorable to retroactively add them to the algebra; hence, we take an *extensible* approach. A feature of VASA is that it is both a *descriptive* algebra *and* an *executable* algebra. On the one hand, VASA offers a descriptive design of a type system with the specialty that vague spatial data types, operations, and predicates are all based on their crisp counterparts and can thus be expressed exclusively in terms of them. On the other hand, this fact means that we can leverage available implementations of crisp spatial algebras, realize VASA on top of them with only minimal effort, and directly *execute* vague geometric set operations and predicates without being forced to design and implement new algorithms for them. In other words, we obtain *executable specifications* that can be directly leveraged as an implementation.

In this section, we describe, illustrate, and formally define our concept of vague spatial objects as the first fundamental part of VASA. Regarding the classification of object models for vague spatial data in Section 2.2, our model belongs to the category of exact object models. Section 3.1 motivates our concept by several application scenarios. Section 3.2 provides the formal definition of vague spatial data types.

3.1. What are vague spatial objects?

The central idea of *vague spatial objects* is to base their definition on already well defined, geometric modeling techniques. Our algebra leverages general exact object models incorporating the crisp spatial data types *point2D*, *line2D*, and *region2D*, as they have been reviewed in Section 2.1. We assume that these data types are closed under (appropriately defined) geometric union, intersection, and difference operations.

Vague spatial objects can, e.g., represent the uncertainty about the precise paths or the spatial extent of phenomena in space; i.e., objects can shrink and extend and hence have a minimal and maximal extent. An example is a lake whose water level depends on the degree of evaporation and on the amount of precipitation. High evaporation implies dry periods and thus a minimal water level. High precipitation entails rainy periods and thus a maximal water level. Islands in the lake are less flooded by water in dry periods and more flooded in rainy periods. If an island can never be completely flooded by water, it forms a "hole" in the lake. But if an island like a sandbank can be flooded completely, it belongs to the vague part of the lake. Hence, we have confident information about the minimal and maximal extent of a lake. But the actual extent of a lake, which is somewhere between these two extreme limits, is vague. Fig. 2 illustrates this spatial constellation. Dark-gray shading shows areas that definitely belong to the lake. Light-gray shading indicates areas that perhaps belong to the lake. White color indicates areas that do definitely not belong to the lake.

Another example is a map of natural resources like iron ore. For some areas experts know the existence of iron ore with certainty because of soil samples and boreholes. For other areas experts are not sure and only assume the

[2] We assume that other needed, well known data types (e.g., alphanumerical data types and crisp spatial data types) with corresponding operations are provided by other algebras (type systems).

incidence of this mineral. These are the kinds of vague spatial objects (in this case: vague regions) we are especially interested in. On the other hand, our concept is also able to model the aspect of fuzziness that areal objects have an extent but cannot be bounded by a precise border like the transition between a mountain and a valley. Continuous changes of features (like air pollution continuously decreasing from city centers to rural areas) cannot be modeled by this concept. This is a predestined application of fuzzy object models (see Section 2.2).

A further example are oil spills. For environmental authorities it is very important to obtain information about the spread of oil slicks in order to be able to take measures for their removal, assess the consequences for the marina flora and fauna, and implement rescue measures. Due to radar and helicopter observations it is possible to determine the minimal distribution of oil slicks. Mathematical models fed by parameters like wind velocity and current enable the prediction of the movement and the possible extent of the oil pollution.

As a final illustrating example, which we also use to introduce our terminology deployed in VASA, we consider a homeland security scenario. Secret services (should) have knowledge of the whereabouts of terrorists. For each terrorist, some of their refuges are precisely known; some others are assumed and thus only conjectures. We can model all these locations as a single *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 a single *vague line* object. Some routes collected in a *kernel line* object have been identified with certainty. Other routes can only be assumed to be taken by a terrorist; they are gathered in a *conjecture line* object. 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 summarize them in a *kernel region* object. From other areas we can only assume that they are the target of terroristic activity; we denote them as a *conjecture region* object. Fig. 3 gives some examples. Dark-gray shaded areas, straight curves, and dark-gray points indicate kernel parts. Areas with light-gray interiors, dashed lines, and light-gray points refer to conjecture parts. White areas describe exterior parts. In this sense, many application scenarios can be found that could leverage our concept of vague spatial objects.

3.2. A generic definition of vague spatial data types

Based on the motivation in the previous subsection, we now give formal definitions of our *vague spatial data types*. An interesting observation is that these definitions can be given in a generic manner in the sense that type-specific considerations and distinctions are unnecessary. For the definition of vague points, vague lines, and vague regions we make use of the data types *point2D* for (complex) crisp points, *line2D* for (complex) crisp lines, and *region2D* for (complex) crisp regions (see Section 2.1), which are defined as special point sets (Section 2.1, [40]) and closed under the geometric set operations $\oplus$ (*union*), $\otimes$ (*intersection*), $\ominus$ (*difference*), and $\boxminus$ (*complement*). Given a type $\alpha \in \{point2D, line2D, region2D\}$, the signatures of the operations are $\otimes, \oplus, \ominus : \alpha \times \alpha \to \alpha$ and $\boxminus : \alpha \to \alpha$. Each type α together with the operations $\oplus$ and $\otimes$ forms a Boolean algebra. We denote the identity of $\otimes$ by $\mathbf{1}$, which corresponds to $\mathbb{R}^2$. We represent the identity of $\oplus$ by $\mathbf{0}$, which corresponds to the empty spatial object (empty point set $\emptyset$). Further, we leverage the three point set topological notions of boundary (∂A), interior (A°), and exterior (A^-) of a spatial object A (see Section 2.1). 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. This leads to the following generic definition of vague spatial data types:

Definition 1. Let $\alpha \in \{point2D, line2D, region2D\}$.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 $w = (w_k, w_c) \in \nu(\alpha)$ and an auxiliary function $points : \nu(\alpha) \to 2^{\mathbb{R}^2}$, which yields the (unknown) point set of a vague spatial object, holds:

(i) $w_k^\circ \cap w_c^\circ = \emptyset$
(ii) $w_k \subseteq points(w) \subseteq w_k \oplus w_c$
(iii) $points(w) \in \alpha$

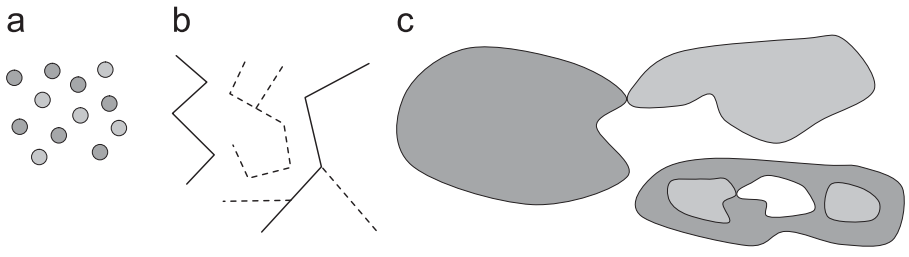


Fig. 3. Examples of a (complex) vague point object (a), a (complex) vague line object (b), and a (complex) vague region object (c). The term "complex" indicates that each collection of components forms a *single* vague spatial object.

We call $w \in v(\alpha)$ a (two-dimensional) *vague spatial object* with *kernel part* w_k and *conjecture part* w_c. Further, we call $w_o := \boxminus(w_k \oplus w_c)$ the *outside part* of w. If $\alpha = point2D$ holds, an element of $v(point2D) = point2D \times point2D =: vpoint2D$ is called a *vague point* object. Correspondingly, an element of $v(line2D) =: vline2D$ is called a *vague line* object, and an element of $v(region2D) =: vregion2D$ is called a *vague region* object. If $w_k = \mathbf{0}$ and $w_c = \mathbf{0}$, we call $w = (\mathbf{0}, \mathbf{0})$ the *empty vague spatial object*.

Syntactically, a vague spatial object is described as a pair of crisp complex spatial objects of the same type. Semantically, four constraints have to be satisfied. In condition (i) we require that the kernel part and the conjecture part have disjoint interiors, since a point of the same object cannot belong to both parts.

Condition (ii) makes two statements. First, it states that the kernel part w_k describes the determinate component of the vague spatial object, that is, the component that definitely and always belongs to the vague spatial object. Second, it states that the conjecture part w_c describes the vague component of the vague spatial object, that is, the component from which we cannot say with any certainty whether it or pieces of it belong to the vague spatial object or not. *Maybe* the conjecture part or pieces of it belong to the vague object; *maybe* this is not the case. We could also say that this is *unknown*. Another, related view is to regard w_k as a *lower* (minimal, guaranteed) *approximation* of w and $w_k \oplus w_c$ as an *upper* (maximally possible, speculative) *approximation* of w. This brings us near to a rough set theoretical view.

Condition (iii) requires that even if we do not know the exact point set of w, the actual point set $points(w)$ may not be arbitrary but must be compatible to type α. A direct conclusion from this requirement is that $points(w) \ominus w_k \in \alpha$ since $w_k \in \alpha$.

We conclude this section with two definitions. The first definition provides us with an alternative formulation of the disjointedness criterion in Definition 1(i). The disjointedness is here expressed on the basis of point set topological notions. We can also specify this constraint by means of *topological cluster predicates* on complex crisp spatial objects [40].

Definition 2. An alternative specification of the semantical constraint in Definition 1(i) in terms of topological cluster predicates is

$$\forall \alpha \in \{point2D, line2D, region2D\}\ \forall w = (w_k, w_c) \in v(\alpha): \\ disjoint(w_k, w_c) \vee meet(w_k, w_c)$$

The cluster predicates *disjoint* and *meet* are overloaded twofold. First, for each combination of crisp spatial data types, they summarize several similar basic topological predicates identified in [40]. Second, these predicate names are used independently of a particular type combination.

Finally, we give the definition of the *characteristic function* of a vague spatial object. This function decides about existence or non-existence of a point in a vague spatial object.

Definition 3. The *characteristic function* $\chi_w(p)$ for a point $p \in \mathbb{R}^2$ and a vague spatial object $w \in v(\alpha)$ is defined as

$$\chi_w(p) \in \begin{cases} \{1\} & \text{if } p \in w_k \\ \{0\} & \text{if } p \in \mathbb{R}^2 - (w_k \cup w_c) \\ \{0,1\} & \text{if } p \in w_c - w_k \end{cases}$$

Note the deliberate use of set-theoretic operations. Especially the common boundary points of w_k and w_c ($w_k \cap w_c \neq \emptyset$) are mapped to 1. Further, we especially obtain that $\chi_w(p) = 1$ for all $p \in points(w)$.

4. Vague spatial operations

In this section, we describe, illustrate, and formally define vague spatial operations as the second fundamental part of VASA. Section 4.1 motivates them by a few application scenarios. In Section 4.2 we give a formal definition of the vague spatial set operations *union*, *intersection*, and *difference*. An interesting aspect is that we define them generically, i.e., independently of the underlying data types. Section 4.3 introduces other generic vague spatial operations and predicates. Type-dependent vague spatial operations are discussed in Section 4.4. Finally, Section 4.5 deals with vague numeric operations.

4.1. What are vague spatial operations?

Unsurprisingly, *vague spatial operations* are spatial operations operating on vague spatial objects. That is, these operations take vague spatial objects as arguments and yield vague objects as a result. A resulting vague object can again be a vague spatial object, or it is a vague number or a vague Boolean. We will later specify what we mean by the latter two types. It is common consensus in the spatial database field that a spatial algebra is never complete since always new operations can be designed and added to it. Hence, we take an extensible approach and assume that VASA will be extended if necessary. Because of their particular importance, vague topological predicates will be dealt with in Section 5.

We now briefly present two real life applications and motivate the use of vague spatial operations. The first example is taken from the animal kingdom. We view the living spaces of different animal species and distinguish kernel parts where they mainly live and conjecture parts like peripheral areas or corridors where they in particular hunt for food or which they cross in order to migrate from one kernel part to another one. We consider the following example queries regarding their living spaces: (1) Find the animal species that (partially) share their living spaces. (2) Determine hunters that penetrate into the living space of other animals. (3) Ascertain the areas where two species can only meet by accident. For two animal species u and v, the interesting situations for the queries are illustrated in Fig. 4. The common task of all three queries is to compute the common living spaces of two animal species, i.e., to calculate the intersection of two vague regions. But the

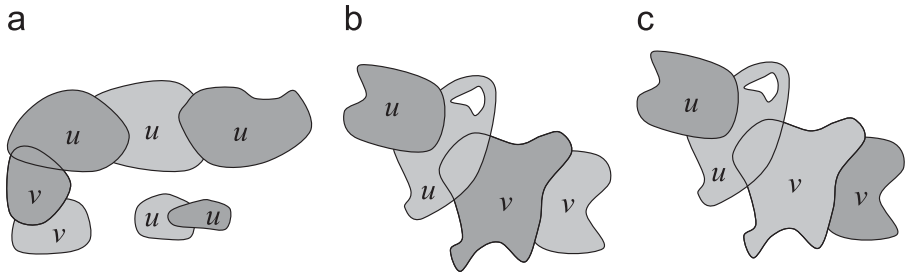


Fig. 4. Example scenarios with a focus on the intersection between two kernel parts (a), a kernel part and a conjecture part (b), and two conjecture parts (c) of two vague regions.

nature of the intersection is different in all three cases. The first query asks for an intersection between two kernel parts (Fig. 4(a)). The (non-empty) result is definitely a kernel part. The second query amounts to an intersection between a kernel part and a conjecture part but not between two kernel parts (Fig. 4(b)). Since a conjecture part is involved that describes a vague part, we cannot make a definite statement whether this intersection belongs to the kernel part. It only remains to regard this intersection as a conjecture part. The third query exclusively asks for an intersection between two conjecture parts (Fig. 4(c)); it is definitely a conjecture part. In total, we see that the "strength of intersection" decreases from left to right in Fig. 4. We show later that for any two vague spatial objects we are able to express both the general, overall intersection operation as well as the special intersection operations addressed in the above three queries. Last but not least, we mention that the intersection of an exterior part with anything else is an exterior part.

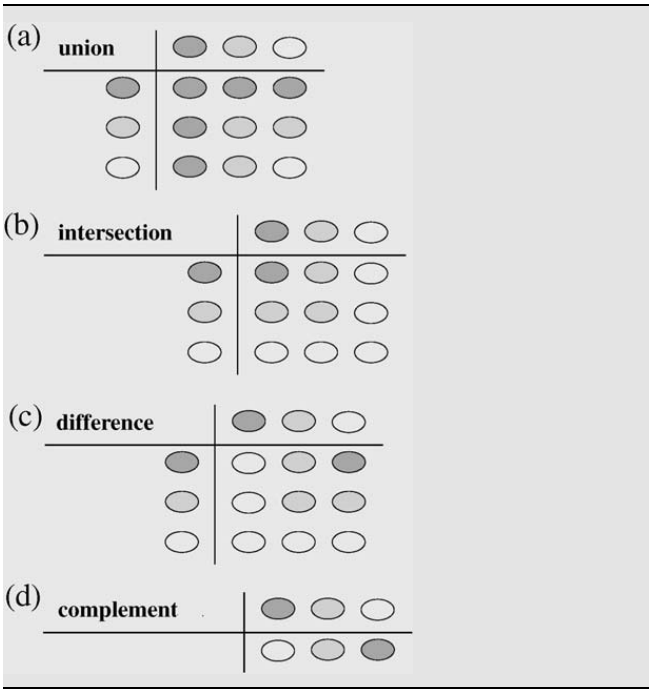
The second example refers to our homeland security scenario from Section 3.1. Taking into account spatial vagueness, we are able to pose interesting queries. We can ask for the locations where any two terrorists have definitely, perhaps, and/or never taken the same refuge. We can determine those terrorists that definitely, perhaps, and/or never operated in the same area. We can compute the locations where routes taken by different terrorists definitely, perhaps, and/or never crossed each other. We can find out the sphere of a number of terrorists on the basis of their locations as a *vague convex hull*.

Many further application scenarios and queries are conceivable. Vague concepts offer a greater flexibility and more nuances for modeling and computing properties of spatial phenomena in the real world than determinate "black or white" concepts do. Still, vague concepts comprise the modeling power of determinate concepts as a special case.

4.2. A generic definition of vague spatial set operations

The three vague geometric set operations **union**, **intersection**, and **difference** have all the same signature $v(\alpha) \times v(\alpha) \to v(\alpha)$ with $\alpha \in \{point, line, region\}$. In addition, we define the operation **complement** with the signature $v(\alpha) \to v(\alpha)$. It is our goal to define these operations 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 (Table 2). For each operation we give a table where a column/row labeled by ●, ◉, or ○ denotes the kernel part, conjecture part, or outside part of u and w, respectively. Each entry of the table denotes a possible combination, i.e., non-empty 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.

Table 2
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 **union** (a), **intersection** (b), **difference** (c), and **complement** (d).



Regarding the *union* operation (Table 2(a)), the intersection component of a kernel part with any other part belongs to the kernel part of the resulting vague spatial object r since the union operation asks for the definite membership in either part only. Likewise, the intersection component of the two conjecture parts or the intersection component of a conjecture part with an outside part belongs to the conjecture part of r, and only the intersection component of the two outside parts belongs to the outside part of r. Regarding the *intersection* operation (Table 2(b)), the intersection component of an outside part with any other part belongs to the outside part of r because intersection requires at least potential membership in both parts. The kernel part of r only contains components which definitely belong to the kernel parts of both operand objects. The intersection component of both conjecture parts or of a conjecture part and a kernel part contribute to the conjecture part of r. Obviously, the *complement* (Table 2(d)) of the kernel part is the outside part, 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, the result is the conjecture part itself. The *difference* (Table 2(c)) between any two parts is equal to the intersection of the first part with the complement of the second part.

Table 2 enables us now to formally define the four operations. 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, by employing an available crisp spatial algebra, we can directly *execute* the vague geometric set operations without being forced to design and implement new algorithms for them (see Section 6.1).

Definition 4. Let $u, w \in v(\alpha)$, and let u_k and w_k denote their kernel parts and u_c and w_c their conjecture parts. We define

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

According to Definition 4(ii) the intersection of two vague line objects yields again a vague line object. This is in contrast to most definitions of its crisp counterpart which is usually specified as $\otimes : \mathit{line2D} \times \mathit{line2D} \to \mathit{point2D}$ [38]. However, our specification stresses the aspects of generic definition, maintenance of closure properties, and consistency among the three instances of the **intersection** operation. An operation named **common_points** that computes the shared points of two vague lines is introduced in Section 4.3.

If u and v are vague spatial objects with empty conjecture parts, i.e., $u_c = \mathbf{0}$ and $v_c = \mathbf{0}$, the operations behave exactly like their crisp counterparts. This holds for all vague spatial operations defined in this article.

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 $(u_k(\boxminus(w_k \oplus w_c)), u_c w_c \oplus u_k w_c \oplus u_c(\boxminus(w_k \oplus w_c)))$.

The following lemma shows that the specifications of Definition 4 fit to the specifications in Table 2.

Lemma 1. *The specifications of Definition 4 realize the behavior of and are consistent to the specifications in Table 2.*

Proof. According to Table 2(a), for $z = u$ **union** w in Definition 4(i), 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) $z_c = u_c w_c \oplus u_c w_o \oplus u_o w_c$
(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:

$$z_k = (u_k(w_k \oplus w_c \oplus w_o)) \oplus ((u_k \oplus u_c \oplus u_o)w_k)$$

Since $w_k \oplus w_c \oplus w_o = \mathbf{1} = \mathbb{R}^2$ and $u_k \oplus u_c \oplus u_o = \mathbf{1}$, we get

$$z_k = (u_k \otimes \mathbf{1}) \oplus (w_k \otimes \mathbf{1}) = u_k \oplus w_k$$

which corresponds to the definition of the kernel part of **union**. For proving Eq. (2) we know that for $r, s \in \alpha$ holds:

$$r \oplus s = rs \oplus r(\boxminus s) \oplus (\boxminus r)s$$

We can use this identity to rewrite the conjecture part specification in Definition 4(i) as

$$(u_c w_c \oplus u_c(\boxminus w_c) \oplus (\boxminus u_c)w_c) \ominus (u_k w_k \oplus u_k(\boxminus w_k) \oplus (\boxminus u_k)w_k)$$

Next we evaluate all complements by using that $\boxminus w_c = w_k \oplus w_o$ and $\boxminus w_k = w_c \oplus w_o$. 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 $u_k w_c$ appear in both parts of the difference; all other intersections to be subtracted have no effect at all since all intersections are pairwise disjoint due to the definition of vague spatial objects. We obtain

$$u_c w_c \oplus u_c w_o \oplus u_o w_c$$

which corresponds exactly to the condition required for z_c. For proving Eq. (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 $\mathbf{1} = r \oplus (\boxminus r)$. Therefore, we know that $s = (r \oplus (\boxminus r))s = rs \oplus (\boxminus r)s$, and it

follows that $r \oplus s = r \oplus rs \oplus (\boxminus r)s$. We also know that $r \oplus rs = r(\mathbf{1} \oplus s) = r$ so that $r \oplus s = r \oplus (\boxminus r)s$. Since $(\boxminus r)s$ 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

$$z_o = \boxminus(z_k \oplus z_c) = \boxminus(u_k \oplus w_k \oplus ((u_c \oplus w_c) \ominus (u_k \oplus w_k))) = \boxminus(u_k \oplus w_k \oplus u_c \oplus w_c)$$

By commutativity and de Morgan's law this reduces to

$$(\boxminus(u_k \oplus u_c)) \otimes (\boxminus(w_k \oplus w_c))$$

which is by the definition of complement equal to $u_o \otimes w_o$, the condition required for z_o.

According to Table 2(b), for $z = u$ **intersection** w in Definition 4(ii), we have to show the three identities

(1) $z_k = u_k w_k$
(2) $z_c = u_c w_k \oplus u_c w_c \oplus u_k w_c$
(3) $z_o = u_o w_k \oplus u_o w_c \oplus u_o w_o \oplus u_k w_o \oplus u_c w_o$

The right-hand sides of the Eqs. (1) and (2) correspond directly to the specifications of the kernel part and the conjecture part of **intersection**. For proving Eq. (3) we duplicate the term $u_o \oplus w_o$, factorize both u_o and w_o, and simplify the equation to

$$z_o = u_o(w_k \oplus w_c \oplus w_o) \oplus (u_k \oplus u_c \oplus u_o)w_o = (u_o \otimes \mathbf{1}) \oplus (\mathbf{1} \otimes w_o) = u_o \oplus w_o$$

On the other hand, we know by definition that

$$z_o = \boxminus(z_k \oplus z_c) = \boxminus(u_k w_k \oplus u_c w_k \oplus u_c w_c \oplus u_k w_c)$$

Again we use the fact that $\otimes$ distributes over $\oplus$ and factorize first both w_k and w_c and then $w_k \oplus w_c$. Afterwards, by de Morgan's law, we obtain

$$z_o = \boxminus((u_k \oplus u_c)w_k \oplus (u_k \oplus u_c)w_c) = \boxminus((u_k \oplus u_c) \otimes (w_k \oplus w_c)) = (\boxminus(u_k \oplus u_c)) \oplus (\boxminus(w_k \oplus w_c))$$

which is by the definition of complement equal to $u_o \oplus w_o$.

According to Table 2(c), for $z = u$ **difference** w in Definition 4(iii), we have to show the three identities

(1) $z_k = u_k w_o$
(2) $z_c = u_k w_c \oplus u_c w_c \oplus u_c w_o$
(3) $z_o = u_k w_k \oplus u_c w_k \oplus u_o w_k \oplus u_o w_c \oplus u_o w_o$

With $w_o = \boxminus(w_k \oplus w_c)$, the right-hand sides of the Eqs. (1) and (2) correspond directly to the specifications of the kernel part and the conjecture part of **difference**. For proving Eq. (3) we duplicate the term $u_o \oplus w_k$, factorize both w_k and u_o, and simplify the equation to

$$z_o = (u_k \oplus u_c \oplus u_o)w_k \oplus u_o(w_k \oplus w_c \oplus w_o) = (\mathbf{1} \otimes w_k) \oplus (u_o \otimes \mathbf{1}) = u_o \oplus w_k$$

On the other hand, we know by definition that

$$z_o = \boxminus(z_k \oplus z_c) = \boxminus(u_k w_o \oplus u_k w_c \oplus u_c w_c \oplus u_c w_o)$$

Again we use the fact that $\otimes$ distributes over $\oplus$ and factorize first both u_k and u_c and then $w_c \oplus w_o$. Afterwards, by de Morgan's law, we obtain:

$$z_o = \boxminus(u_k(w_c \oplus w_o) \oplus u_c(w_c \oplus w_o)) = \boxminus((u_k \oplus u_c) \otimes (w_c \oplus w_o)) = (\boxminus(u_k \oplus u_c)) \oplus (\boxminus(w_c \oplus w_o))$$

which is by the definition of complement equal to $u_o \oplus w_k$.

According to Table 2(d), for $z =$ **complement** u in Definition 4(iv), we have to show the three identities

(1) $z_k = u_o$
(2) $z_c = u_c$
(3) $z_o = u_k$

Since $u_o = \boxminus(u_k \oplus u_c)$ holds, the right-hand side of Eq. (1) corresponds to the specification of the kernel part of **complement**. The right-hand side of Eq. (2) is equal to the conjecture part of **complement**. For proving Eq. (3) we have by definition that

$$z_o = \boxminus(z_k \oplus z_c) = \boxminus(u_o \oplus u_c) = u_k$$

This concludes the consideration of the correctness of all four operations. □

4.3. Other generic vague spatial operations and predicates

In Section 4.2, we have defined the operation **intersection** for two vague spatial objects of the same type. We extend this definition now to all mixed type combinations such that two vague spatial objects have a different dimension. This allows us, e.g., to determine the components of a vague line that intersect (i.e., lie in) a vague region, or the components of a vague point located on a vague line. The result is always an object of the lower dimension. Let "$<_T$" be an order relation on the three vague spatial data types that is defined as $\nu point2D <_T \nu line2D <_T \nu region2D$. The signatures of **intersection** are then either $\nu(\alpha) \times \nu(\beta) \to \nu(\alpha)$ or $\nu(\beta) \times \nu(\alpha) \to \nu(\alpha)$ with $\nu(\alpha) <_T \nu(\beta)$. To enable this extension, we generalize the crisp intersection operation $\otimes$ to all corresponding crisp variants of the just mentioned signatures. These crisp variants are well defined [38]. The already known Definition 4(ii) for **intersection** can then also be applied in case of the mixed type combinations.

Sometimes it is helpful to be able to only deal with the kernel part or the conjecture part of a vague spatial object and thus to suppress the other part, or to swap its kernel part and conjecture part.

Definition 5. Let $u = (u_k, u_c) \in \nu(\alpha)$. We define the following operations with the signature $\nu(\alpha) \to \nu(\alpha)$:

(i) **kernel**(u) $:=$ $(u_k, \mathbf{0})$
(ii) **conjecture**(u) $:=$ $(\mathbf{0}, u_c)$
(iii) **invert**(u) $:=$ (u_c, u_k)

The following operations give access to the components of vague spatial objects and enable structural comparisons.

Definition 6. Let $u = (u_k, u_c) \in v(\alpha)$ and $w = (w_k, w_c) \in v(\alpha)$. We define

(i) **k-proj**(u) $:=$ u_k
(ii) **c-proj**(u) $:=$ u_c
(iii) $u = w$ $:=$ $(u_k = w_k \wedge u_c = w_c)$
(iv) $u \neq w$ $:=$ $(u_k \neq w_k \vee u_c \neq w_c)$

The first two operations with the signature $v(\alpha) \to \alpha$ map the kernel part or the conjecture part of a vague spatial object to the corresponding crisp spatial object and hence leave VASA. The equality and inequality predicates with the signature $v(\alpha) \to bool$ compare the structural definition of both vague spatial objects. Note that the predicate "=" is different from the topological predicate *equal* defined in Section 5.

4.4. Type-dependent vague spatial operations

Type-dependent spatial operations only permit a limited subset of the available types as domain and codomain. We first introduce a few operations that either give access to object components of lower dimension (Definition 7(i)–(iv)) or produce an object of higher dimension (Definition 7(v)–(viii)). To keep the balance between kernel part and conjecture part, each operation is available in two versions in which either the kernel part or the conjecture part of the argument object dominates. The structure of the definitions is rather similar for both the "kernel versions" and the "conjecture versions" of all operations.

Definition 7. Let $p \in vpoint2D$, $l \in vline2D$, $m \in \{vline2D, vregion2D\}$, and $r \in vregion2D$. We define

(i) **k-vertices**(m) $:=$ $(vertices(m_k), vertices(m_c) \ominus vertices(m_k))$
(ii) **c-vertices**(m) $:=$ $(vertices(m_k) \ominus vertices(m_c), vertices(m_c))$
(iii) **k-boundary**(r) $:=$ $(boundary(r_k), boundary(r_c) \ominus boundary(r_k))$
(iv) **c-boundary**(r) $:=$ $(boundary(r_k) \ominus boundary(r_c), boundary(r_c))$
(v) **k-interior**(l) $:=$ $(interior(l_k), interior(l_c) \ominus interior(l_k))$
(vi) **c-interior**(l) $:=$ $(interior(l_k) \ominus interior(l_c), interior(l_c))$
(vii) **k-convex_hull**(p) $:=$ $(convex\_hull(p_k), convex\_hull(p_k \oplus p_c) \ominus convex\_hull(p_k))$
(viii) **c-convex_hull**(p) $:=$ $(convex\_hull(p_k \oplus p_c) \ominus convex\_hull(p_c), convex\_hull(p_c))$

All definitions make use of the representation of crisp and vague lines and regions as linear approximations and leverage well defined crisp spatial operations. The spatial operation *vertices* with the signatures $line2D \to point2D$ and $region2D \to point2D$ collects the segment end points of a crisp line or crisp region boundary, respectively. The operation *boundary* with the signature $region2D \to line2D$ determines the boundary of a crisp region object and represents it as a closed crisp line object. The operation *interior* with the signature $line2D \to region2D$ calculates the "outmost" cycles of a crisp line object and collects them into a crisp region object. That is, all segments of a cycle that lie within some other cycle are removed. The well known operation *convex_hull* with the signature $point2D \to region2D$ computes the smallest convex region that contains all points of a given *point2D* object.

If $op \in \{vertices, boundary, interior, convex\_hull\}$ with $op : \alpha \to \beta$, then, by *vague lifting*, we obtain the vague counterparts **k-op**, **c-op** : $v(\alpha) \to v(\beta)$ with their semantics given in Definition 7. We get a kernel version and a conjecture version for each crisp spatial operation since, according to Definition 1(i), the intersection of the interiors of the kernel part and the conjecture part of the resulting vague spatial object must be empty. Hence, the interior of the common components of the kernel part and the conjecture part of the resulting vague spatial object can and has to be either assigned to its kernel part or its conjecture part. A few examples of the vague spatial operations are given in Figs. 5 and 6.

Finally in Definition 8, we define two further intersection operations. The operation **common_points** with the signature $vline2D \times vline2D \to vpoint2D$ computes the point intersections between two vague line objects. It leverages the crisp operation *common_points* with the signature $line2D \times line2D \to point2D$ that determines the intersection points of two crisp line objects. The operation **common_border** computes the shared boundary of two extended vague spatial objects where at least one operand is a vague region object; the result is a vague line object. Its possible signatures are $vline2D \times vregion2D \to vline2D$, $vregion2D \times vline2D \to vline2D$, and $vregion2D \times vregion2D \to vline2D$.

Definition 8. Let $l, m \in vline2D$ and $r, s \in vregion2D$. We define

(i) **common_points**(l, m) $:=$ $(common\_points(l_k, m_k), common\_points(l_c, m_c) \oplus common\_points(l_k, m_c) \oplus common\_points(l_c, m_k))$
(ii) **common_border**(l, r) $:=$ **intersection**$(l,$ **k-boundary**$(r))$
(iii) **common_border**(r, l) $:=$ **common_border**(l, r)
(iv) **common_border**(r, s) $:=$ **intersection**(**k-boundary**(r), **k-boundary**$(s))$

4.5. Vague numeric operations

Spatial operators returning numerical values provide information about metric features of spatial objects. Examples are the *area* of a region or the *length* of a line. These operations are well understood and defined in the crisp spatial domain [38]. However, we will see in this subsection that *vague numeric operations* turn out to be more complicated than perhaps expected.

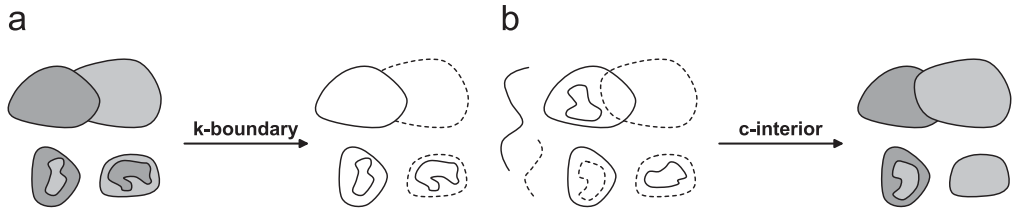


Fig. 5. Examples for the operations **k-boundary** (a) and **c-interior** (b).

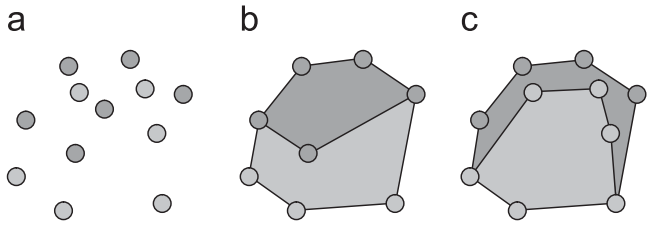


Fig. 6. A vague point object p (a), **k-convex_hull**(p) (b), and **c-convex_hull**(p) (c).

The reason is that the vague character of the spatial argument objects leads to vague numerical results. For example, the area of a vague region is at least equal to the area of the kernel part and at most equal to the area of the kernel part and the conjecture part, i.e., it is somewhere in between. We cannot express the vague area value by a single scalar number only. But an appropriate representation could be an *interval* given by the minimum and maximum area values. However, this leads to the problem that we have to introduce a data type for intervals together with an *interval arithmetic* [27] as a separate algebra into DBMSs. Further, interval arithmetic can be slow and often gives overly pessimistic results for real-world computations since chained operations on intervals often cause the size of the intervals to grow rapidly. In our case of VASA, the introduction of interval arithmetic would lead to a large overhead. We thus decide to keep things simple and instead define *two* operations for computing the lower bound and the upper bound of such intervals. This enables us to keep ordinary numeric operations.

Definition 9 specifies several unary vague spatial operators yielding numerical values. Minimal numerical result values are guaranteed by taking into account the kernel part of a vague spatial object only. Maximal numerical result values are bounded by considering the kernel part and the conjecture part together. The definitions of these operators are based on the well known crisp spatial operators $length : line2D \to real$, $area : region2D \to real$, $diameter : \alpha \to real$ with $\alpha \in \{point2D, line2D, region2D\}$, and $no\_of\_comp : point2D \to real$. The vague spatial operators are obtained by vague lifting of the crisp operators. They exist both in a "lower bound" version and an "upper bound" version and are self-explanatory.

Definition 9. Let $p \in vpoint2D$, $l \in vline2D$, $r \in vregion2D$, and $t \in \{vpoint2D, vline2D, vregion2D\}$. Then:

(i)	**min-length**(l)	$:=$	$length(l_k)$
(ii)	**max-length**(l)	$:=$	$length(l_k \oplus l_c) = length(l_k)$ $+length(l_c \ominus l_k)$
(iii)	**min-area**(r)	$:=$	$area(r_k)$
(iv)	**max-area**(r)	$:=$	$area(r_k \oplus r_c)$ $= area(r_k) + area(r_c)$
(v)	**min-diameter**(t)	$:=$	$diameter(t_k)$
(vi)	**max-diameter**(t)	$:=$	$diameter(t_k \oplus t_c)$
(vii)	**min-no_of_comp**(p)	$:=$	$no\_of\_comp(p_k)$
(viii)	**max-no_of_comp**(p)	$:=$	$no\_of\_comp(p_k)$ $+no\_of\_comp(p_c)$

There are other useful operations on crisp spatial objects for which a generalization to the vague case is surprisingly not quite so simple or even impossible. We illustrate this for two desired operations for which we can neither define a lower bound version nor an upper bound version. An operation returning the number of connected components of a vague line or region object depends heavily on the semantics of the conjecture part. Since vague line and region objects need not be connected, their conjecture parts, which express a "possibly line" and "possibly region"-semantics, respectively, might well allow several unconnected components. Fig. 7 shows possible instances of a vague line and a vague region and demonstrates the unpredictability of the number of components. Hence, we cannot give an upper bound on the number of components. We also cannot give a non-trivial lower bound, e.g., for the number of kernel part components since kernel part components might be connected by conjecture part components. In Fig. 8 the minimal number of components is 1 although there are three kernel part components.

If we consider the calculation of the perimeter of a vague region, in a first approach one could be attempted to define lower bound and upper bound versions similar to the definitions of area or length. However, this leads to wrong results. The conjecture part describes possible locations of the region's contour (and thus the region itself). This makes it impossible to determine any upper bound on the length of such a boundary contour. In particular, the actual contour might be much longer than the perimeter of the conjecture

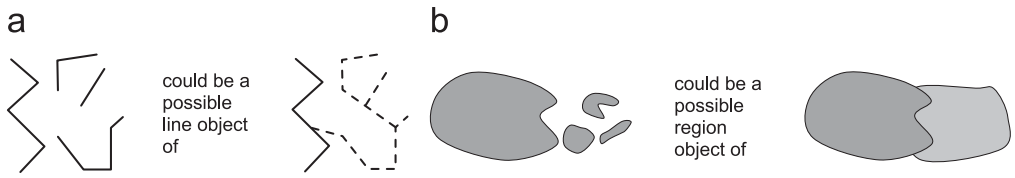


Fig. 7. A possible instance of a vague line (a) and a vague region (b) showing the impossibility to determine an upper bound for the number of components.

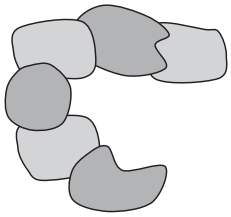


Fig. 8. Example vague region object where the kernel part consists of three components but the minimal number of components is 1.

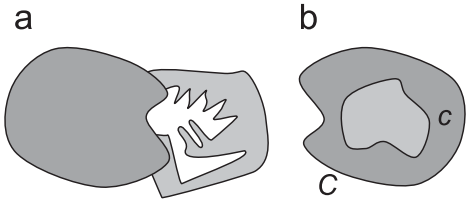


Fig. 9. Example showing that an upper bound (a) and a lower bound (b), respectively, of the perimeter of a vague region object cannot be determined.

region. Fig. 9(a) illustrates this situation. Moreover, we cannot simply take the perimeter of the kernel part as the minimal perimeter. Fig. 9(b) illustrates this. Usually, holes contribute to the perimeter of a region. If, e.g., a kernel part of a vague region v contains a hole which is equal to the conjecture part of v, the perimeter of the hole is not counted for the perimeter of the maximal possible region. In Fig. 9(b), the minimal possible region has the maximal possible perimeter $\mathit{length}(C) + \mathit{length}(c)$ whereas the maximal possible region has the minimal possible perimeter of $\mathit{length}(C)$.

Definition 10 presents two distance operators with the common signature $\nu(\alpha) \times \nu(\beta) \rightarrow \mathit{real}$ where $\alpha, \beta \in \{\mathit{point2D}, \mathit{line2D}, \mathit{region2D}\}$. Again, these operators exist both in a lower bound version and an upper bound version. The operations make use of the highly overloaded crisp distance functions $\mathit{mindist}, \mathit{maxdist} : \alpha \times \beta \rightarrow \mathit{real}$ that compute the nearest and farthest distance between any two crisp spatial objects.

Definition 10. Let $\alpha, \beta \in \{\mathit{vpoint2D}, \mathit{vline2D}, \mathit{vregion2D}\}$, $u \in \alpha$, and $v \in \beta$. We define

(i) **min-min-dist**$(u, v) \coloneqq \mathit{mindist}(u_k \oplus u_c, v_k \oplus v_c)$
(ii) **max-min-dist**$(u, v) \coloneqq \mathit{mindist}(u_k, v_k)$
(iii) **min-max-dist**$(u, v) \coloneqq \mathit{maxdist}(u_k, v_k)$
(iv) **max-max-dist**$(u, v) \coloneqq \mathit{maxdist}(u_k \oplus u_c, v_k \oplus v_c)$

If we consider the aspect of minimum distance first, an upper bound is obtained by the distance between the kernel parts of two vague spatial objects, say u and v. That is, we are sure that the distance is at most the distance between the kernel parts (operation **max-min-dist**). The distance might be smaller, but it is at least as large as the distance between the maximal extensions of u and v (operation **min-min-dist**). If **max-min-dist**$(u, v) =$ **min-min-dist**(u, v) holds, we can conclude that the minimum distance is exact since the conjecture parts have no effect on the distance computation.

If we consider the aspect of maximum distance, an assured lower bound is given by the maximal distance between the kernel parts of u and v (operation **min-max-dist**). The maximal distance might be larger if we in addition take into account the conjecture parts of u and v (operation **max-max-dist**). If **min-max-dist**$(u, v) =$ **max-min-dist**(u, v) holds, the maximum distance is exact since the conjecture parts have no influence on the distance computation.

5. Vague topological predicates

In this section, we describe and formally define our concept of vague topological predicates as the third fundamental part of VASA. Similarly to the definition of vague spatial objects and operations, we define vague topological predicates on the basis of well defined crisp topological predicates. Our goal is to obtain a complete set of mutually exclusive vague topological predicates. Section 5.1 gives a motivation for vague topological predicates and illustrates their properties. For the formal determination of these predicates, we employ a three-step method introduced in Section 5.2. The remaining subsections detail its three steps. The first step leverages a concept for representing vague topological predicates by means of crisp topological predicates and is described in Section 5.3. The second step delineated in Section 5.4 explicitly identifies all possible representations for vague topological predicates. The third and last step presented in Section 5.5 gives semantics to the possible representations and results in a set of vague topological predicates.

5.1. What are vague topological predicates?

A topological relationship characterizes the relative position of two spatial objects. If these objects are crisp, topological predicates operating on them can precisely answer questions like “Do states A and B share a common border?” or “Do two roads cross?” by means of a binary

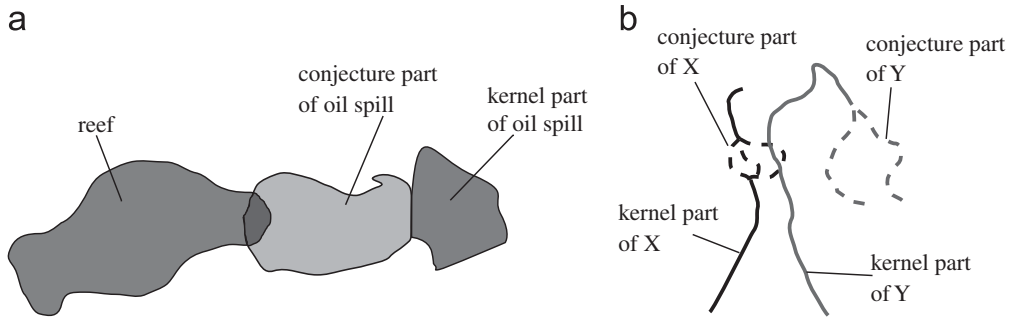


Fig. 10. Two example scenarios for illustrating vague topological relationships. In (a) the black area indicates the intersection of the conjecture part of the oil spill with the kernel part of the coral reef. In (b) the conjecture part of the route of X intersects with the kernel part of the route of Y.

decision. Answering such questions turns out to be more challenging when dealing with vague spatial objects. For illustration purposes, we present two examples based on situations described in Section 3. After an oil spill, authorities may need to make decisions on how to act depending on whether the predicted area of the oil spill overlaps with or is disjoint from a known coral reef. Let us again assume that the oil spill area and the reef are modeled as vague regions and further that the reef has an empty conjecture part. The authorities may need an answer to the question "Is the oil spill disjoint from the coral reef?". The answer to such a question might not be clear due to the existence of a conjecture part in the oil spill representation. If the kernel part of the spill region is disjoint from the reef but the conjecture part is not (as illustrated in Fig. 10(a)), we can say that *maybe* the two vague regions are disjoint, *maybe* they are not. In case of mapping terrorist routes using vague lines, authorities might need an answer to the question: "Have terrorist X and terrorist Y been at equal locations?" A simple look at their routes might give us a clear answer, or it might not. If the routes of X and Y meet or intersect at a point that belongs to the kernel of both routes, then X and Y surely met. If there are no intersection points between the routes, then they surely did not meet. But if their routes intersect at a point that is a conjecture point of either or both routes (as illustrated in Fig. 10(b)), then maybe they met; in this case there is no clear answer to the question.

The examples provided demonstrate that a two-valued, Boolean logic is insufficient for determining the topological relationships between vague spatial objects. We employ a three-valued logic that, in addition to the truth values true and false, includes a value maybe for taking into account the aspect of vagueness. A vague topological predicate returns then one of these three truth values. It returns true if the relationship definitely holds, false if the relationship does definitely not hold, and maybe if there is not enough information to make a clear decision.

As an example, we assume that two vague topological predicates between vague regions are named overlap and *equal*. Applied to the oil spill and reef example, we obtain $overlap(oil\_spill, reef) = maybe$ and $equal(oil\_spill, reef) = false$. It is clear that the two objects *oil_spill* and *reef* will never be topologically equal since their kernel parts are disjoint. It is not clear though whether or not the objects overlap, as this depends on the actual extension of the so far unknown oil spill.

5.2. A general method for the determination of vague topological predicates

Our approach to modeling vague topological predicates rests on our overall paradigm of leveraging crisp spatial concepts for the design of vague spatial concepts. Fig. 11 gives an overview of our general method of determining topological predicates on vague spatial objects. As first input parameters, the method requires any combination of vague spatial data types $\nu(\alpha)$ and $\nu(\beta)$, which are built from the crisp spatial data types α and β by the type constructor ν (Section 3.2). The second input parameter is the complete and mutually exclusive collection $T^{\alpha,\beta}$ of topological relationships between the types α and β from [40].

Our method incorporates three main steps. In a first step, Section 5.3 specifies a *characterization* of vague topological relationships. The idea is to characterize the vague topological relationship that holds between two vague spatial objects by means of the crisp topological relationships that hold between their kernel and con-

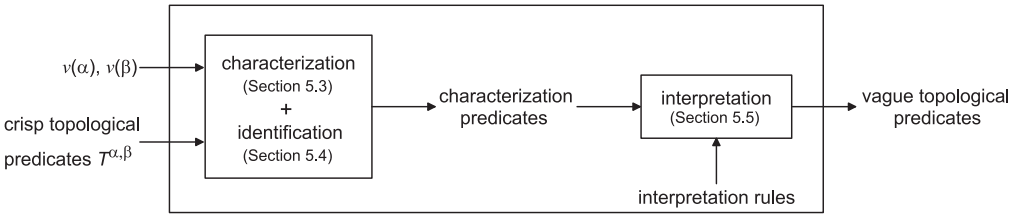


Fig. 11. General method for determining vague topological predicates.

jecture parts. Unfortunately, not all characterizations that can be syntactically formed are semantically valid. Therefore, in a second step, Section 5.4 proceeds with the *identification* of the complete collection of unique characterizations that are valid, crisp representations of the topological relationships between two vague spatial objects. We denote the members of this collection as *characterization predicates*. The third step of our method is the *interpretation* step, which we detail in Section 5.5. During this step, the semantics of the kernel and conjecture parts are taken into account in order to generate *interpretation rules* that analyze each characterization predicate to reach a definition of topological predicates between vague spatial objects. The application of the interpretation rules results in three-valued predicates that besides the typical Boolean values *true* and *false* can return the value *maybe* in cases for which an answer is unclear.

5.3. Characterization predicates for representing vague topological predicates

The first step of our method provides the characterization of vague topological predicates on the basis of crisp topological predicates. It gives an answer to the question which topological relationships between the individual kernel and conjecture parts of two vague spatial objects we must take into account to adequately represent the topological relationship between the vague spatial objects themselves. We have seen in Fig. 10 that we cannot appropriately represent the topological relationships between vague spatial objects by only taking into account their kernel parts as their lower approximations. Instead, we also have to consider the upper approximation of each object, i.e., the kernel part and the conjecture part together. As a result, we characterize the topological relationship of two vague spatial objects by determining the topological relationships of the lower and upper approximations of one vague spatial object with the lower and upper approximations of the other vague spatial object, respectively. Further, we have to consider the *implicitly* given topological relationship of the lower and upper approximations of the *same* vague spatial object. For the scenario in Fig. 10(a), the characterization of the relative position of the oil spill and the reef results in the following conjunction of crisp topological predicates:

$$disjoint(oil\_spill_k, reef_k) \wedge overlap(oil\_spill_k \oplus oil\_spill_c, reef_k) \wedge coveredBy(oil\_spill_k, oil\_spill_k \oplus oil\_spill_c)$$

The first two elements of this conjunction are topological predicates from the set $T^{\alpha,\beta}$ that operate on objects of the crisp spatial data types α and β. In our example, $\alpha = \beta = region$ holds. The third element of the conjunction is a topological predicate from the set $T^{\alpha,\alpha}$ that operates on two objects of the crisp spatial data type α (here: *region*) and characterizes the relationship between the lower and upper approximations of the oil spill. In general, we denote such a conjunction of crisp topological predicates as a *characterization predicate*. As a shortcut, we represent it as the n-tuple of n crisp topological predicates contained in its conjunction. In the example above, it is the 3-tuple $(disjoint, overlap, coveredBy)$.

In our general considerations, we do not know the exact topological relationship between the kernel part and the conjecture part of a vague spatial object. Definition 1(ii) only requires for an object $A = (A_k, A_c) \in \nu(\alpha)$ that $A_k \subseteq A_k \oplus A_c$ must hold. Hence, we define a subset $T^{\alpha,\alpha}_{in}$ as the set of topological relationships that respect this condition. A special subset of relationships representing equality is given by $T^{\alpha,\alpha}_{eq}$.

Definition 11. Let $\alpha \in \{point2D, line2D, region2D\}$. The set $T^{\alpha,\alpha}_{in}$ of *containment* relationships is given by

$$T^{\alpha,\alpha}_{in} := \{p \in T^{\alpha,\alpha} | \exists A = (A_k, A_c) \in \nu(\alpha) : p(A_k, A_k \oplus A_c)\}$$

The set $T^{\alpha,\alpha}_{eq}$ of *equality* relationships is given by

$$T^{\alpha,\alpha}_{eq} := \{p \in T^{\alpha,\alpha} | \exists A = (A_k, A_c) \in \nu(\alpha): p(A_k, A_k \oplus A_c) \Rightarrow (A_k = A_k \oplus A_c \Leftrightarrow A_c = \emptyset)\}$$

Indeed, $T^{\alpha,\alpha}_{in}$ is a set of predicates. For example, for simple regions, this set is equal to $\{inside, coveredBy, equal\}$ [12]. An analysis of the topological relationships identified in [40] shows that $|T^{point2D,point2D}_{in}| = 2$, $|T^{line2D,line2D}_{in}| = 10$, and $|T^{region2D,region2D}_{in}| = 5$. The set $T^{\alpha,\alpha}_{eq} (\subset T^{\alpha,\alpha}_{in})$ is also a set of predicates. From [40] we know that $|T^{point2D,point2D}_{eq}| = 1$, $|T^{line2D,line2D}_{eq}| = 2$, and $|T^{region2D,region2D}_{eq}| = 1$.

The specification of a characterization predicate also depends on the non-emptiness and emptiness of the kernel and conjecture parts serving as operands of the crisp topological predicates in its conjunction. The reason is that the emptiness of a crisp spatial object (i.e., an empty crisp spatial object) makes a topological predicate false.

Definition 12. Let $\alpha, \beta \in \{point2D, line2D, region2D\}$. Then we obtain

$$\forall A \in \alpha\ \forall B \in \beta\ \forall p \in T^{\alpha,\beta} : (A = \mathbf{0} \vee B = \mathbf{0}) \Rightarrow p(A, B) = false$$

In a conjunction of predicates, a predicate yielding false makes the conjunction yield false. Table 3 lists all 16 possible combinations of non-empty and empty kernel and conjecture parts from two vague spatial objects and determines for each combination a valid conjunction of crisp topological predicates that operate on the non-empty parts of their operand objects. This conjunction serves as the definition of the *parameterized characterization predicate* for such a pair of vague spatial objects. Such a characterization predicate is called parameterized since it still depends on an appropriate selection of the parameters $p, q, r, s \in T^{\alpha,\beta}$, $v \in T^{\alpha,\alpha}_{in} - T^{\alpha,\alpha}_{eq}$, and $w \in T^{\beta,\beta}_{in} - T^{\beta,\beta}_{eq}$ (see Section 5.4).

Case 1 of Table 3 forms the basis for the construction of all (parameterized) characterization predicates. It represents the standard case in which all kernel and conjecture parts are non-empty. Hence, the characterization

Table 3
Definition of characterization predicates on the basis of the non-emptiness and emptiness of the kernel and conjecture parts of two vague spatial objects A and B and the crisp topological predicates $p, q, r, s \in T^{\alpha,\beta}$, $v \in T^{\alpha,\alpha}_{in} - T^{\alpha,\alpha}_{eq}$, and $w \in T^{\beta,\beta}_{in} - T^{\beta,\beta}_{eq}$.

Case	A_k	A_c	B_k	B_c	Conjunction of crisp topological predicates
1	$\neq \mathbf{0}$	$\neq \mathbf{0}$	$\neq \mathbf{0}$	$\neq \mathbf{0}$	$p(A_k, B_k) \wedge q(A_k \oplus A_c, B_k) \wedge r(A_k, B_k \oplus B_c) \wedge s(A_k \oplus A_c, B_k \oplus B_c) \wedge v(A_k, A_k \oplus A_c) \wedge w(B_k, B_k \oplus B_c)$
2	$\neq \mathbf{0}$	$\neq \mathbf{0}$	$\neq \mathbf{0}$	$= \mathbf{0}$	$p(A_k, B_k) \wedge q(A_k \oplus A_c, B_k) \wedge v(A_k, A_k \oplus A_c)$
3	$\neq \mathbf{0}$	$\neq \mathbf{0}$	$= \mathbf{0}$	$\neq \mathbf{0}$	$r(A_k, B_c) \wedge s(A_k \oplus A_c, B_c) \wedge v(A_k, A_k \oplus A_c)$
4	$\neq \mathbf{0}$	$\neq \mathbf{0}$	$= \mathbf{0}$	$= \mathbf{0}$	*false*
5	$\neq \mathbf{0}$	$= \mathbf{0}$	$\neq \mathbf{0}$	$\neq \mathbf{0}$	$p(A_k, B_k) \wedge r(A_k, B_k \oplus B_c) \wedge w(B_k, B_k \oplus B_c)$
6	$\neq \mathbf{0}$	$= \mathbf{0}$	$\neq \mathbf{0}$	$= \mathbf{0}$	$p(A_k, B_k)$
7	$\neq \mathbf{0}$	$= \mathbf{0}$	$= \mathbf{0}$	$\neq \mathbf{0}$	$r(A_k, B_c)$
8	$\neq \mathbf{0}$	$= \mathbf{0}$	$= \mathbf{0}$	$= \mathbf{0}$	*false*
9	$= \mathbf{0}$	$\neq \mathbf{0}$	$\neq \mathbf{0}$	$\neq \mathbf{0}$	$q(A_c, B_k) \wedge s(A_c, B_k \oplus B_c) \wedge w(B_k, B_k \oplus B_c)$
10	$= \mathbf{0}$	$\neq \mathbf{0}$	$\neq \mathbf{0}$	$= \mathbf{0}$	$q(A_c, B_k)$
11	$= \mathbf{0}$	$\neq \mathbf{0}$	$= \mathbf{0}$	$\neq \mathbf{0}$	$s(A_c, B_c)$
12	$= \mathbf{0}$	$\neq \mathbf{0}$	$= \mathbf{0}$	$= \mathbf{0}$	*false*
13	$= \mathbf{0}$	$= \mathbf{0}$	$\neq \mathbf{0}$	$\neq \mathbf{0}$	*false*
14	$= \mathbf{0}$	$= \mathbf{0}$	$\neq \mathbf{0}$	$= \mathbf{0}$	*false*
15	$= \mathbf{0}$	$= \mathbf{0}$	$= \mathbf{0}$	$\neq \mathbf{0}$	*false*
16	$= \mathbf{0}$	$= \mathbf{0}$	$= \mathbf{0}$	$= \mathbf{0}$	*false*

predicate includes the crisp topological predicates between the lower approximations (kernel parts), between the upper approximations (kernel parts and conjecture parts together), and between the combinations of lower and upper approximations of both objects. In addition, we have to take into account the implicitly given topological relationships between the kernel and conjecture parts of the same object. The resulting characterization predicate is represented by the 6-tuple (p, q, r, s, v, w) with $p, q, r, s \in T^{\alpha,\beta}$, $v \in T^{\alpha,\alpha}_{in} - T^{\alpha,\alpha}_{eq}$, and $w \in T^{\beta,\beta}_{in} - T^{\beta,\beta}_{eq}$. For v and w, we have to subtract the equality relationships since $A_k \neq A_k \oplus A_c$ (i.e., $A_c \neq \emptyset$) and $B_k \neq B_k \oplus B_c$ (i.e., $B_c \neq \emptyset$) must hold due to the assumptions in case 1.

All other characterization predicates are reductions of this conjunction of six topological predicates. A reduction is performed in the sense that topological predicates that would yield false according to Definition 12 are removed from this conjunction. For example, the difference between case 9 and case 1 in Table 3 is that $A_k = \emptyset$ holds in case 9. For the 6-tuple of case 1 this means that $p(A_k, B_k) = p(\emptyset, B_k) = \mathit{false}$, $r(A_k, B_k \oplus B_c) = r(\emptyset, B_k \oplus B_c) = \mathit{false}$, and $v(A_k, A_k \oplus A_c) = v(\emptyset, A_k \oplus A_c) = \mathit{false}$ hold. Hence, the predicates p, r, and v are removed in case 9, and we obtain the characterization predicate $q(A_c, B_k) \wedge s(A_c, B_k \oplus B_c) \wedge w(B_k, B_k \oplus B_c)$ and hence the 3-tuple (q, s, w).

Cases 2 and 5 deal with those situations in which one of the objects has an empty conjecture part (see Fig. 10 where the reef has an empty conjecture part). Their characterization predicates are denoted by the 3-tuples (p, q, v) and (p, r, w), respectively. Cases 3 and 9 describe scenarios in which exactly one of the objects has an empty kernel part. In such a case, the characterization predicates include the crisp topological predicates between the upper approximation of one object, which is here its conjecture part only, and the lower and upper approximations of the other object. The characterization predicates are represented by the 3-tuples (r, s, v) and (q, s, w).

Case 6 represents a scenario in which both objects have non-empty kernel parts and empty conjecture parts. This effectively renders the objects crisp. Thus, we denote the characterization predicate as the crisp topological predicate p that holds between their kernel parts. Case 11 represents a scenario in which both objects have empty kernel parts and non-empty conjecture parts. The characterization predicate of such a situation is defined as the topological relationship s between the upper approximations (here conjecture parts only) of both objects involved. Cases 7 and 10 handle the situations where one object has only a non-empty kernel part and the other object has only a non-empty conjecture part. The resulting conjunction contains a single topological predicate between the lower approximation of one object and the upper approximation of the other object, which is here its conjecture part only. The characterization predicates are represented by the topological predicates r and q, respectively.

Finally, cases 4, 8, and 12–16 deal with those situations for which there is no valid conjunction of crisp topological predicates that can be used to define the characterization predicates. This is due to the fact that in all these cases either A or B is the empty vague spatial object ($\mathbf{0}$, $\mathbf{0}$). The lower and upper approximations of an empty vague spatial object are therefore the empty crisp spatial object $\mathbf{0}$, and, according to Definition 12, a topological predicate will yield false when one of its operands is the empty object. As a result, no satisfiable conjunction can be specified in these cases. Therefore, we only have to further consider the non-*false* cases in Table 3 which represent the syntactically possible characterizations.

5.4. Identification of valid characterization predicates

It turns out that the syntactically possible characterizations identified in Section 5.3 do not all represent semantically valid characterization predicates. Section 5.4.1 gives reasons for this fact and formalizes the identification problem by leveraging the well known

concept of *binary constraint networks*. Hence, as the second step of our general method for determining vague topological predicates (see Fig. 11), in Section 5.4.2, we perform the *identification* of all *valid* characterization predicates and use concepts from *relation algebra* (not to be mixed up with *relational algebra*) for this purpose.

5.4.1. The identification problem

The cases $i \in \{1,2,3,5,6,7,9,10,11\}$ in Table 3 represent those characterizations, i.e., conjunctions of crisp topological predicates, that are syntactically possible and not false from the beginning. However, these characterizations are parameterized and so far independent of the underlying combination of crisp spatial data types α and β, the set $T^{\alpha,\beta}$ of topological predicates specified for such a type combination, and the selected topological predicates p, q, r, and s from $T^{\alpha,\beta}$, v from $T^{\alpha,\alpha}_{in} - T^{\alpha,\alpha}_{eq}$, and w from $T^{\beta,\beta}_{in} - T^{\beta,\beta}_{eq}$. Therefore, the question arises whether all these characterizations are also *valid* (i.e., semantically possible) if we make actual choices for the parameters α, β, p, q, r, s, v, and w.

Let $V^{\alpha,\beta}_i$ denote the sets of valid characterization predicates that correspond to the characterization of case i. Rephrasing the question, we are interested in the cardinality and composition of the sets $V^{\alpha,\beta}_i$, i.e., the issue is whether all potential n-tuples ($n \in \{1,3,6\}$) represent valid characterization predicates. We can immediately conclude that $V^{\alpha,\beta}_6 = V^{\alpha,\beta}_7 = V^{\alpha,\beta}_{10} = V^{\alpha,\beta}_{11} = T^{\alpha,\beta}$ must hold since the elements of these sets are non-composite values, i.e., 1-tuples, and all $|T^{\alpha,\beta}|$ elements are known to be valid topological predicates [40] and thus valid characterization predicates. This is different for the 6-tuple set $V^{\alpha,\beta}_1$, for which $V^{\alpha,\beta}_1 \subset T^{\alpha,\beta} \times T^{\alpha,\beta} \times T^{\alpha,\beta} \times T^{\alpha,\beta} \times (T^{\alpha,\alpha}_{in} - T^{\alpha,\alpha}_{eq}) \times (T^{\beta,\beta}_{in} - T^{\beta,\beta}_{eq})$ holds, for the 3-tuple sets $V^{\alpha,\beta}_2$ and $V^{\alpha,\beta}_3$, for which $V^{\alpha,\beta}_2 = V^{\alpha,\beta}_3 \subset T^{\alpha,\beta} \times T^{\alpha,\beta} \times (T^{\alpha,\alpha}_{in} - T^{\alpha,\alpha}_{eq})$ holds, and for the 3-tuple sets $V^{\alpha,\beta}_5$ and $V^{\alpha,\beta}_9$, for which $V^{\alpha,\beta}_5 = V^{\alpha,\beta}_9 \subset T^{\alpha,\beta} \times T^{\alpha,\beta} \times (T^{\beta,\beta}_{in} - T^{\beta,\beta}_{eq})$ holds. That is, the 6-tuple set and the 3-tuple sets are all *proper* subsets of Cartesian products over $T^{\alpha,\beta}$, $T^{\alpha,\alpha}_{in}$, and $T^{\beta,\beta}_{in}$, respectively. The reason is that, due to possible semantical contradictions between different topological predicates in the same conjunction, some n-tuples do not hold, and we have thus to identify the valid combinations. For example, for case 5 in Table 3, let us set $p = overlap$, $r = disjoint$, and $w = inside$ for $\alpha = \beta = region$. We obtain the term $overlap(A_k, B_k) \wedge disjoint(A_k, B_k \oplus B_c) \wedge inside(B_k, B_k \oplus B_c)$. This is obviously a contradiction since $disjoint(A_k, B_k \oplus B_c) \Rightarrow disjoint(A_k, B_k) \neq overlap(A_k, B_k)$ holds. Hence, the 3-tuple (*overlap, disjoint, inside*) is invalid and not an element of $V^{\alpha,\beta}_5$. Note that we can already say that the 1-tuple sets are equal, the 3-tuple sets $V^{\alpha,\beta}_2$ and $V^{\alpha,\beta}_3$ are equal, and the 3-tuple sets $V^{\alpha,\beta}_5$ and $V^{\alpha,\beta}_9$ are equal since the structure and the argument types of their respective conjunctions are equal. However, all 3-tuples will be later assigned a different semantics (Section 5.5) since they describe the topological relationship between different combinations of kernel parts, conjecture parts, and whole vague spatial objects.

The *identification problem* is therefore to determine all valid characterization predicates for the 6-tuple set $V^{\alpha,\beta}_1$ and the 3-tuple sets $V^{\alpha,\beta}_2$, $V^{\alpha,\beta}_3$, $V^{\alpha,\beta}_5$, and $V^{\alpha,\beta}_9$ for each pair α, β of crisp spatial data types and each set $T^{\alpha,\beta}$ of topological predicates on these types. This means that for the determination of $V^{\alpha,\beta}_1$ the topological consistency and thus the validity of $|T^{\alpha,\beta}|^4(|T^{\alpha,\alpha}_{in}| - |T^{\alpha,\alpha}_{eq}|)(|T^{\beta,\beta}_{in}| - |T^{\beta,\beta}_{eq}|)$ 6-tuples has to be checked. For $V^{\alpha,\beta}_2$ and $V^{\alpha,\beta}_3$ we have to check $|T^{\alpha,\beta}|^2(|T^{\alpha,\alpha}_{in}| - |T^{\alpha,\alpha}_{eq}|)$ 3-tuples, and for $V^{\alpha,\beta}_5$ and $V^{\alpha,\beta}_9$ the amount of $|T^{\alpha,\beta}|^2(|T^{\beta,\beta}_{in}| - |T^{\beta,\beta}_{eq}|)$ 3-tuples. Table 5 shows the actual numbers. Topological consistency denotes the absence of logical contradictions among topological predicates of the same spatial scenario.

To come closer to a solution, we regard the identification problem as a binary *constraint satisfaction problem* [25]. According to Table 3, each 6-tuple and each 3-tuple represent a logical conjunction of so far unknown binary constraints where each constraint represents a binary topological predicate operating on two crisp spatial objects. In our case, the task is to assign actual topological predicates to the parameters p, q, r, s, v, and w so that the resulting conjunction is consistent. Such a conjunction can be represented as a directed graph called *binary constraint network* [25]. Since our case involves spatial objects and topological relationships, we call such a network a *binary spatial constraint network* (*BSCN*). Each spatial object that is an argument of a binary relation in a conjunction is mapped to a distinct node in the network, and the binary relations among them are represented by the *direction* and the *label* of the edges. An edge is directed to distinguish between a binary relation and its converse.

Fig. 12 shows *parameterized BSCNs* (*p-BSCNs*) for the 6-tuple and the 3-tuples. The BSCNs are parameterized since they contain the parameters p, q, r, s, v, and w as edge labels. For each assignment of topological predicates to these parameters, we obtain a BSCN whose topological consistency has to be checked. Each node of a BSCN represents a spatial object involved in an n-tuple (conjunction). That is, the object assigned to a node is either A_k, B_k, A_c, B_c, $A = A_k \oplus A_c$, or $B = B_k \oplus B_c$, respectively. All nodes are disjoint. The directed edges between the nodes represent their topological relationships and are labeled with p, q, r, s, v, or w. For example, for a predicate $p(A_k, B_k)$ we get the edge $A_k \xrightarrow{p} B_k$. In order to obtain the complete description of the topology of a scene with n spatial objects and their $\frac{1}{2}(n^2 + n)$ binary topological relationships, we supplement each p-BSCN to a *complete directed graph with self-loops*, i.e., there is a directed edge between any ordered pair of nodes. First, we add all edges that represent the converse relation $\bar{z}$ for any relation z. In case of topological relationships, we have $z \in T^{\alpha,\beta} \Leftrightarrow \bar{z} \in T^{\beta,\alpha}$ [40]. For example, $inside(A, B) = contains(B, A)$ and $disjoint(A, B) = disjoint(B, A)$ hold for simple regions. Second, to any node whose assigned spatial object is of type α, we add a self-loop representing the *identity relation* $id^{\alpha,\alpha}$ of $T^{\alpha,\alpha}$. Each set $T^{\alpha,\alpha}$ has such an identity relation [40]. For example, in the case of simple regions, the identity relation is the *equal* predicate. Any two nodes within the

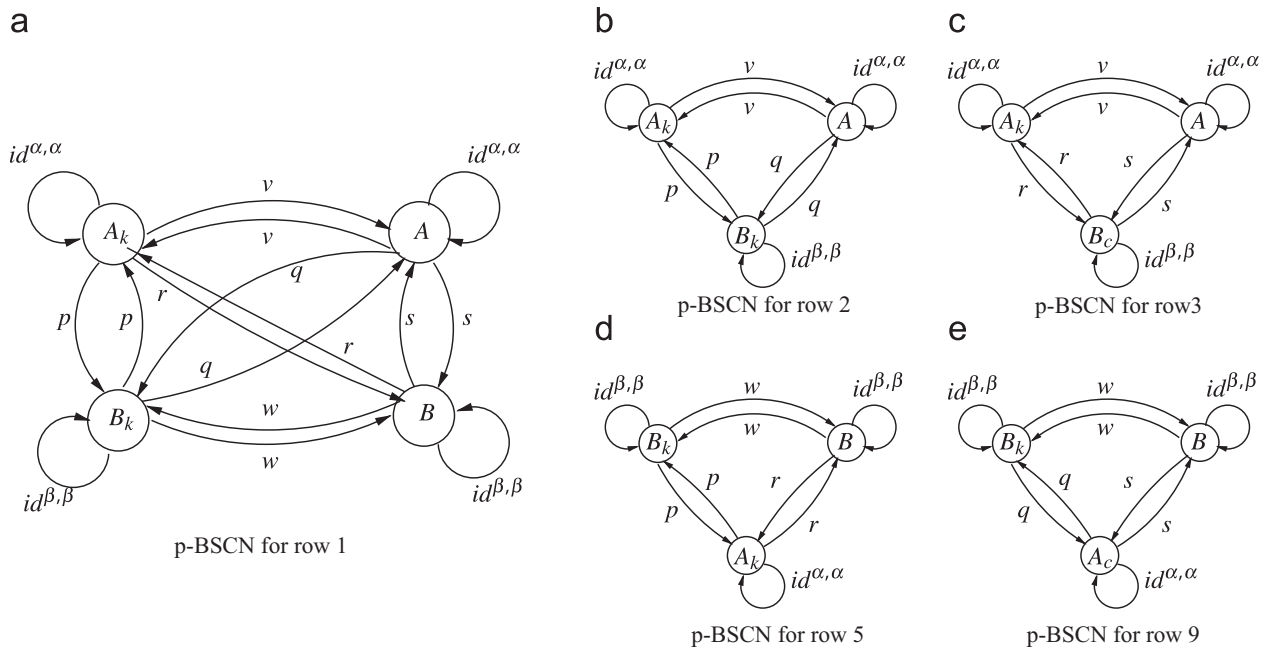


Fig. 12. Parameterized BSCN representations of the 6-tuples and 3-tuples.

BSCN are linked by a *directed path*, i.e., a sequence of edges that point in the same direction. The number of edges along a directed path is called the *path length*.

5.4.2. The identification process

The view of our identification problem as a constraint satisfaction problem and its representation as a p-BSCN in the form of a complete directed graph with self-loops enable us to provide an algorithm that checks for each given instantiation of the p-BSCN with crisp topological predicates whether it leads to a topologically consistent scenario. If a consistent instantiation is found, its topological predicates entail a conjunction that represents a valid characterization predicate. In case of an inconsistent instantiation, its combination of topological predicates has to be ignored. The development of an algorithm finding all topologically consistent n-tuples ($n \in \{3, 6\}$) of topological predicates for the p-BSCNs is the goal of this subsection. We already know that all possible 1-tuples are consistent.

To provide a consistent BSCN, it has been proven in [24] that its nodes, edges, and paths must fulfill the three conditions of *node consistency*, *arc consistency*, and *path consistency*. If all three levels of consistency are fulfilled for the topological predicates of a BSCN, the respective scenario is topologically consistent. The first two conditions are immediately implied by the properties of topological predicates and our construction of the p-BSCNs. *Node consistency* requires for each node the existence of a self-loop denoting the identity relation $id^{\alpha,\alpha}$. It adds the property of *reflexivity* to a BSCN. *Arc consistency* presupposes a node consistent constraint network and additionally requires for each directed edge $A \xrightarrow{q} B$, which represents the topological predicate $q(A, B)$, the existence of an edge $B \xrightarrow{\overline{q}} A$ in the reverse direction from B to A, which represents the converse topological predicate $\overline{q}(B, A)$. In this way, it adds the property of *symmetry* to a BSCN. While node consistency and arc consistency are defined relative to a single binary relation, *path consistency* considers the consistency of two (and more) binary relations. More precisely, it guarantees the consistency of the *compositions* of binary relations and adds the property of *transitivity* to a BSCN. The *composition* operation "$\circ$" is an operation on relations and an element of the *relation algebra* (see [42] where the symbol ";" is used for composition). Its general definition is as follows:

Definition 13. Let α, β, and γ be (not necessarily different) sets, and let $p \subseteq \alpha \times \beta$ and $q \subseteq \beta \times \gamma$ be two binary relations. The (*existential*) *composition* of p and q is defined as

$$p \circ q = \{(A, C) | A \in \alpha, C \in \gamma, \exists B \in \beta : (A, B) \in p \wedge (B, C) \in q\} \subseteq \alpha \times \gamma$$

In our p-BSCNs, $\alpha, \beta, \gamma \in \{point2D, line2D, region2D\}$, $p \in T^{\alpha,\beta}$ and $q \in T^{\beta,\gamma}$ hold. Note that in our context we cannot conclude in general that $p \circ q \in T^{\alpha,\gamma}$ holds since $p \circ q$ is frequently not a basic topological relation $r \in T^{\alpha,\gamma}$ but corresponds to a union of disjoint topological relations $r_1, \ldots, r_m \in T^{\alpha,\gamma}$ with $1 \leq m \leq |T^{\alpha,\gamma}|$. That is, $p \circ q = r_1 \dot{\cup} \cdots \dot{\cup} r_m$ where $\dot{\cup}$ denotes disjoint set union. Therefore, we call the composition on topological relations *weak composition*. To illustrate weak composition, consider three simple regions A, B, and C and the topological relationships A *meet* B and B *covers* C. Inferring the relation between A and C does not lead to a unique topological relationship since A *disjoint* C or A *meet* C may hold. That is, $meet \circ covers = disjoint \dot{\cup} meet$.

Usually, any two nodes A and C in a BSCN can be connected by several paths of possibly different lengths. However, we can leverage the fact proven in [26] that it is sufficient to *only* consider *all* compositions of path length 2 that connect A to C in order to infer the consistency of their relation. This means that in a BSCN we have to investigate

Table 4
The composition table $C^{point2D,point2D,point2D}$.

$\circ$		1 $di = disjoint$	2 $eq = equal$	3 $in = inside$	4 $co = contains$	5 $ov = overlap$
1	di	$\{di, eq, in, co, ov\}$	$\{di\}$	$\{di, in, ov\}$	$\{di\}$	$\{di, in, ov\}$
2	eq	$\{di\}$	$\{eq\}$	$\{in\}$	$\{co\}$	$\{ov\}$
3	in	$\{di\}$	$\{in\}$	$\{in\}$	$\{di, eq, in, co, ov\}$	$\{di, in, ov\}$
4	co	$\{di, co, ov\}$	$\{co\}$	$\{eq, in, co, ov\}$	$\{co\}$	$\{co, ov\}$
5	ov	$\{di, co, ov\}$	$\{ov\}$	$\{in, ov\}$	$\{di, co, ov\}$	$\{di, eq, in, co, ov\}$

Note that the two-letter codes for the topological relations stand for their matrix numbers.

the consistency of all "triangles" (i.e., triples of relations) that have edges of the kind $A \xrightarrow{p} B$, $B \xrightarrow{q} C$, and $A \xrightarrow{r} C$. As an example, we assume three simple regions A, B, and C, and the three topological relationships A *disjoint* B, B *contains* C, and A *overlap* C. We have to test whether $overlap \subseteq disjoint \circ contains$ holds, i.e., whether overlap can be inferred from the composition of *disjoint* and contains. Intuitively, this is obviously not the case. However, $disjoint \subseteq disjoint \circ contains$ holds, and we can even conclude equality. These considerations enable us to specify testing criteria for path consistency in the following way:

Definition 14. Let $\mathcal{N}$ be a BSCN with nodes $A_1, \ldots, A_n$, and let p_{ij} be the topological relation between the two nodes A_i and A_j. We define

(i) $\mathcal{N}$ is *path consistent* $\Leftrightarrow$ $\forall 1 \le i,j \le n :$
$$p_{ij} \subseteq \bigcap_{1 \le k \le n} p_{ik} \circ p_{kj}$$

(ii) $\mathcal{N}$ is *path consistent* $\Leftrightarrow$ $\forall 1 \le i,j,k \le n : p_{ij} \subseteq p_{ik} \circ p_{kj}$

where composition binds closer than intersection.

If the equality in (i) holds, then p_{ij} is the topological relation that completely and uniquely describes the result of the intersection of all compositions of path length 2 between the objects A_i and A_j. As said above, the composition on topological relations is weak. Hence, the fulfillment of the subset relationship for p_{ij} in (i) and (ii) is sufficient to lead to a consistent triangle. Statement (ii) is a simple conclusion of statement (i) and will be used in our algorithm.

For the algorithmic solution of our problem, we first have to determine the composition of topological relations between complex crisp spatial objects. For this purpose, we apply the reasoning method from [10] to our case. This method is based on a set of inference rules which operate on the 9-intersection matrices (Section 2.1, [12,40]) representing the topological relationships and which leverage standard laws for the transitivity of set inclusion. A detailed algorithmic description of this method is beyond the scope of this paper. The result of this method is a collection of *composition tables* $C^{\alpha,\beta,\gamma}$ with $\alpha, \beta, \gamma \in \{point2D, line2D, region2D\}$. The first column of such a table lists the *matrix numbers* i of the topological relations (predicates) $p_i \in T^{\alpha,\beta}$ for $1 \le i \le |T^{\alpha,\beta}|$ as they have been specified in [40] for each combination of spatial data types α and β. In the same way, the first row lists the matrix numbers j of the topological relations $q_j \in T^{\beta,\gamma}$ for the spatial data types β and γ and $1 \le j \le |T^{\beta,\gamma}|$. $C^{\alpha,\beta,\gamma}$ contains $|T^{\alpha,\beta}| \cdot |T^{\beta,\gamma}|$ elements (see Table 1 for the values of $|T^{\alpha,\beta}|$and $|T^{\beta,\gamma}|$). The table entry $C^{\alpha,\beta,\gamma}(i,j)$ represents the subset $L \subseteq \{1, \ldots, |T^{\alpha,\gamma}|\}$ of those matrix numbers such that for all $k \in L$ the topological predicate $r_k \in T^{\alpha,\gamma}$ belongs to the weak composition $p_i \circ q_j$ of the relations $p_i \in T^{\alpha,\beta}$ and $q_j \in T^{\beta,\gamma}$, i.e., $r_k \subseteq p_i \circ q_j$ holds.

As an example, Table 4 shows the composition table $C^{point2D,point2D,point2D}$. Due to the small number of five topological relations between two *point2D* objects, we can give them names that are listed too. We observe that the composition of some pairs of topological relationships between complex *point2D* objects results in more than one relation (matrix number). For instance, $disjoint \circ overlap = disjoint \cup inside \cup overlap$. For $A, B, C \in point2D$ this means that $(A,B) \in disjoint$ and $(B,C) \in overlap$ implies that either $(A,C) \in disjoint$ or $(A,C) \in inside$ or $(A,C) \in overlap$. In predicate notation, we can write: $(disjoint(A,B) \wedge overlap(B,C)) \Rightarrow (disjoint(A,C) \vee inside(A,C) \vee overlap(A,C))$.

In total, 21 different composition tables $C^{\alpha,\beta,\gamma}$ are needed. This number can be explained as follows: We aim at checking the correctness of statements of the form $(X,Y) \in p \wedge (Y,Z) \in q \Rightarrow (X,Z) \in r$ where X, Y, and Z are spatial objects of possibly different types *point2D*, *line2D*, or *region2D* and where p, q, and r are topological predicates. However, in our case, objects of at most two different types can be involved in a BSCN since $A_k, A_c, A \in \alpha$ and $B_k, B_c, B \in \beta$. That is, at least two spatial objects must have the same type in a triangle. If X and Y have the same type, all three types are possible for Z. Since there are three possibilities so that X and Y have the same type, we get $3 \cdot 3 = 9$ combinations. If X and Y have different types, the type of Z must be either equal to the type of X or to the type of Y. Since there are six possibilities such that X and Y have different types, we get $6 \cdot 2 = 12$ combinations. Since, in general, $p \circ q \neq q \circ p$ holds, composition is not commutative so that this cannot reduce the total number of 21 composition tables.

Further, our algorithmic solution requires a representation of our BSCNs. We represent a BSCN describing the topological relationships of $n \in \{3,4\}$ different nodes (spatial objects) $O_1, \ldots, O_n$ by an $n \times n$-matrix M where each entry M_{ij} contains the matrix number of the topological relationship that holds between the spatial objects O_i and O_j. We call M *constraint matrix* or *relation*

```
01 algorithm ValidCharacterizationPredicateIdentification
02 input:  α1 = α2 = α ∈ {point2D, line2D, region2D}
03         α3 = α4 = β ∈ {point2D, line2D, region2D}
04         // α1 is the type of O1 = Ak, α2 is the type of O2 = A,
05         // α3 is the type of O3 = Bk, α4 is the type of O4 = B
06         T^{α,α} if α = β; T^{α,α}, T^{α,β}, T^{β,β} otherwise
07 output: V1^{α,β}
08 begin
09   // Generate composition tables
10   if α = β then generate C^{α,α,α}
11   else generate C^{α,α,α}, C^{α,α,β}, C^{α,β,α}, C^{β,α,α}, C^{β,β,α},
12                 C^{β,α,β}, C^{α,β,β}
13   endif;
14   V1^{α,β} := ∅;
15   for each mp in 1...|T^{α,β}|, mq in 1...|T^{α,β}|,
16            mr in 1...|T^{α,β}|, ms in 1...|T^{α,β}|,
17            mv in 1...|T_in^{α,α}|−| T_eq^{α,α}|,
18            mw in 1...|T_in^{β,β}|−| T_eq^{β,β}| do
19     // Reinitialize constraint matrix M
20     M11 := m_id^{α,α}; M12 := mv; M13 := mp; M14 := mr;
21     M21 := mv; M22 := m_id^{α,α}; M23 := mq; M24 := ms;
22     M31 := mp; M32 := mq; M33 := m_id^{β,β}; M34 := mw;
23     M41 := mr; M42 := ms; M43 := mw; M44 := m_id^{β,β};
24     // Check the path consistency of all triangles of M
25     valid := true;
26     for each i in 1...4, j in 1...4, k in 1...4 do
27       valid := valid and (Mij ∈ C^{αi,αk,αj}(Mik, Mkj))
28     endfor;
29     if valid then
30       V1^{α,β} := V1^{α,β} ∪ {(p, q, r, s, v, w)}
31     endif
32   endfor
33 end ValidCharacterizationPredicateIdentification.
```

Fig. 13. Path consistency algorithm for the identification of all valid characterization predicates for case 1.

matrix. M_{ii} represents the matrix number of the identity relation. For case 1 we set $O_1 = A_k$, $O_2 = A$, $O_3 = B_k$, $O_4 = B$, for case 2 $O_1 = A_k$, $O_2 = A$, $O_3 = B_k$, for case 3 $O_1 = A_k$, $O_2 = A$, $O_3 = B_c$, for case 5 $O_1 = A_k$, $O_2 = B_k$, $O_3 = B$, and for case 9 $O_1 = A_c$, $O_2 = B_k$, and $O_3 = B$ such that the indices can now be used for M. Any other assignment would work too. We are now able to derive and formulate the concept of path consistency in terms of our data structure:

Lemma 2. *Let M be the constraint matrix of a p-BSCN in Fig. 12 with the spatial objects $O_1, \ldots, O_n$ ($n \in \{3, 4\}$). Let the function τ yield the type of O_i ($1 \le i \le n$). Then the following statement holds*:

M is path consistent

$$\Leftrightarrow \forall 1 \le i,j,k \le n : M_{ij} \in C^{\tau(O_i),\tau(O_k),\tau(O_j)}(M_{ik}, M_{kj})$$

Proof. According to Definition 14(ii), we have to show that the topological relation p_{ij} between any two objects O_i and O_j can be inferred by all compositions of any two topological relations p_{ik} between O_i and a third object O_k and p_{kj} between O_k and O_j, i.e., $p_{ij} \subseteq p_{ik} \circ p_{kj}$ must hold for all $1 \le k \le n$. We represent each relation p_{ij} by its matrix number M_{ij}. Each composition $p_{ik} \circ p_{kj}$ can be determined by a lookup in one of the precalculated composition tables by taking the matrix numbers M_{ik} and M_{kj} of p_{ik} and p_{kj}, respectively as indices. The lookup yields the set $L^{i,k,j}$ of those matrix numbers whose pertaining topological relations can be inferred. The choice of the composition table depends on the types $\tau(O_i)$, $\tau(O_k)$, and $\tau(O_j)$ of the involved objects. Finally, the question whether p_{ij} can be inferred from the composition of p_{ik} and p_{kj} amounts to the membership test whether $M_{ij} \in L^{i,k,j}$ holds. □

For the identification of the valid characterization predicates from the p-BSCNs in Fig. 12, we need three path consistency algorithms. One algorithm covers case 1, the second algorithm deals with the cases 2 and 3, and the third algorithm handles the cases 5 and 9. The algorithms are very similar but differ in the number of involved spatial objects and/or in the number of object components per type. Therefore, we present only the algorithm for case 1. This algorithm is shown in Fig. 13. A first input of this algorithm (lines 2–5) are the data types α_1, α_2, α_3, and α_4 of the four components A_k, A, B_k, and B. Either the components are of two different data types α and β, or all components have the same data type α. In case of a single type α, a second input (line 6) is the set of topological predicates defined on this type. Otherwise, we need the topological predicates on both data types α and β alone and between both data types. The output is a set of 6-tuples that describe the valid characterization predicates for case 1.

The algorithm first generates the needed composition tables (lines 9–13). In case of two involved types, we need seven composition tables; otherwise, one table is sufficient. The assignment of actual topological predicates to the p-BSCN in Fig. 12(a) is performed by a multiple nested for-loop (lines 15–32). This loop consecutively traverses the matrix number ranges of the parameters p, q, r, s, v, and w (lines 15–18) for the topological relations and thus also enables access to the matrix numbers of the parameters $\overline{p}$, $\overline{q}$, $\overline{r}$, $\overline{s}$, $\overline{v}$, and $\overline{w}$ for the converse topological relations. These matrix numbers are denoted by the variables m_p, m_q, m_r, m_s, m_v, m_w, $m_{\overline{p}}$, $m_{\overline{q}}$, $m_{\overline{r}}$, $m_{\overline{s}}$, $m_{\overline{v}}$, and $m_{\overline{w}}$ and are assigned to the corresponding elements of the constraint matrix M (lines 20–23).Each new instantiation of the matrix number variables results in a new constraint matrix M and thus in a new BSCN that has to be checked for path consistency.

Path consistency is tested for each constraint matrix, i.e., BSCN (lines 25–28). The strategy is to consider all node pairs i and j of the BSCN and to check whether all triangles that can be formed with an additional node k are path consistent. For this purpose, we perform the membership test of Lemma 2 for each triangle (line 27). If any membership test of any triangle fails, the BSCN is not path consistent, and the 6-tuple under consideration is skipped. Only if all membership tests regarding all

Table 5
Syntactically possible characterizations and semantically possible (i.e., valid) characterization predicates for the cases identified in Table 3.

Case	Type combination α/β					
	point2D/point2D	*point2D/line2D*	*point2D/region2D*	*line2D/line2D*	*line2D/region2D*	*region2D/region2D*
1	625/46	307 328/2157	9604/166	2 893 579 264/1 491 955	109 401 632/369 831	18 974 736/186 985
2, 3	25/11	196/46	49/19	53 792/4788	14 792/2071	4356/1092
5, 9	25/11	1568/269	196/89	53 792/4788	7396/1558	4356/1092
6, 7, 10, 11	5/5	14/14	7/7	82/82	43/43	33/33

Table 6
Vague logical operators.

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

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

not	*t*	*m*	*f*
	f	*m*	*t*

triangles succeed, the current 6-tuple specifies a valid characterization predicate and is added to $V_1^{\alpha,\beta}$ (lines 29–31).

Two similar algorithms enable us to compute $V_2^{\alpha,\beta}$ and $V_3^{\alpha,\beta}$ as well as $V_5^{\alpha,\beta}$ and $V_9^{\alpha,\beta}$. Table 5 presents the numbers of valid characterization predicates for all type combinations. Due to space limitations and the large number of valid characterization predicates, their listing is not given here.

A comparison (not presented here also due to space limitations) to the available approaches for vague topological predicates [6–8,32–34] based on exact models (Section 2.3) reveals that any such predicate has a matching in our collection of characterization predicates.

5.5. Interpretation of characterization predicates for determining vague topological predicates

The third and last step of our general method consists in providing an *interpretation* (see Fig. 11) of the systematically derived characterization predicates into vague topological predicates. We remember that characterization predicates are conjunctions of crisp topological predicates (Table 3) and thus Boolean predicates. Their interpretation is needed since they do not distinguish the different semantics of kernel parts and conjecture parts of the two vague spatial operand objects but treat all parts as crisp objects. As we have indicated in Section 5.1, vague topological predicates require a *three-valued logic* that, in addition to the truth values true and false, includes a value maybe for taking into account the aspect of vagueness. The result of a vague topological predicate is then a value of a new vague data type $v(bool) = vbool = \{true, false, maybe\}$. Each vague topological predicate has the signature $v(\alpha) \times v(\beta) \rightarrow vbool$. The definition of the *vague logical operators* **and**, **or**, and **not** (Table 6) reflects the influence of the value maybe and parallels the definition of the vague spatial operations **union**, **intersection**, and **complement** in Table 2 (*t*, *f*, and *m* are used as abbreviations for true, false, and maybe).

The assignment of an interpretation to the characterization predicates is a subjective procedure. Hence, different interpretations of the same collection of characterization predicates are conceivable. This can be of interest for various applications preferring a different interpretation of the same characterization predicates. All interpretations have in common that they have to take into account the semantics of the kernel and conjecture parts of the two vague spatial objects being the operands of a vague topological predicate. For example, an intersection of two kernel parts has to be assessed differently than the intersection of two conjecture parts. In this subsection, we will give one possible example of such an interpretation.

Finding an appropriate interpretation is impeded by the large numbers of characterization predicates for almost all type combinations (Table 5) that make their manageability difficult. The user is unable to handle a large, overwhelming set of detailed predicates for each type combination and prefers a reduced and manageable set instead. Thus, an interpretation must incorporate a *clustering* or *grouping* of the characterization predicates, which we can hence also regard as (uninterpreted) *unclustered vague topological predicates*. As an example, we present an interpretation of the characterization predicates that leads to the eight *clustered vague topological predicates* named **disjoint**, **meet**, **inside**, **coveredBy**, **equal**, **overlap**, **contains**, and **covered**, by analogy to the well known eight topological predicates between simple regions. The clustered predicates are defined generically in the sense that first they are applicable to all six type combinations and that second they cover all cases listed in Table 5.

In order to achieve these goals, we pursue a strategy consisting of two elements. The first element defines each of the eight clustered vague topological predicate by three *interpretation rules* that specify when the predicate is supposed to yield true, false, and maybe respectively. The second, subsequent element matches all characterization predicates of all type combinations for all cases against the eight clustered vague topological predicates. In total, we achieve a clustering in two ways. On the one hand, each single clustered vague topological predicate subdivides the characterization predicates into three groups depending on which of the three interpretation rules a characterization predicate satisfies. On the other hand, several characterization predicates yield the same truth

values for all eight clustered vague topological predicates. Both elements of our strategy are not independent of each other. The three interpretation rules of each clustered vague topological predicate must be complete in the sense that they can provide a response to all characterization predicates of a given type combination. Further, the vague topological predicates must fulfill the requirement of mutual exclusiveness. This ensures that a single characterization predicate is not interpreted as *true* for more than one clustered vague topological predicate, or as *maybe* for at least one such predicate and *true* for another. For example, the interpretation of a characterization predicate applied to two vague spatial objects A and B cannot yield "certainly disjoint" (i.e., **disjoint**$(A, B) =$ true) and at the same time "maybe overlap" (i.e., **overlap**$(A, B) =$ maybe) since this leads to a possible contradiction. Besides, we exclude that a characterization predicate is interpreted as false by *all* vague topological predicates.

We describe this strategy in more detail now and present its first element comprising the interpretation rules for the eight clustered vague topological predicates. The rules are designed in a manner that they hold for all type combinations. Since the crisp topological predicates used in the characterization predicates are specific for each type combination and also not detailed enough, the definition of the interpretation rules rests on the non-emptiness and emptiness of the intersections between the boundary, interior, and exterior components of the kernel parts, conjecture parts as well as the union of the kernel and conjecture parts of both vague spatial objects. Due to space limitations, we only present the interpretation rules for (the most general) case 1 of Table 3.

The clustered vague topological predicate **disjoint** considers two vague spatial objects to be certainly disjoint if the interiors and boundaries of both objects do not intersect each other. They are certainly not disjoint if their interiors or their boundaries intersect, or if their boundaries intersect their interiors. Otherwise it is unclear whether they are disjoint or not. For $A \in \alpha$, $B \in \beta$ with $\alpha, \beta \in \{vpoint2D, vline2D, vregion2D\}$ we obtain

$$\textbf{disjoint}(A,B) = \begin{cases} true & \text{if } (A_k \oplus A_c)^\circ \cap (B_k \oplus B_c)^\circ = \emptyset \wedge \\ & (A_k \oplus A_c)^\circ \cap \partial(B_k \oplus B_c) = \emptyset \wedge \\ & \partial(A_k \oplus A_c) \cap (B_k \oplus B_c)^\circ = \emptyset \wedge \\ & \partial(A_k \oplus A_c) \cap \partial(B_k \oplus B_c) = \emptyset \wedge \\ & \partial A_k \cap \partial B_k = \emptyset \\ false & \text{if } A_k^\circ \cap B_k^\circ \neq \emptyset \vee A_k^\circ \cap \partial B_k \neq \emptyset \vee \\ & \partial A_k \cap B_k^\circ \neq \emptyset \vee \partial A_k \cap \partial B_k \neq \emptyset \\ maybe & \text{otherwise} \end{cases}$$

The next clustered vague topological predicate we discuss is **meet**. Two vague spatial objects certainly meet if their boundaries definitely intersect but their interiors definitely do not. Since the interpretation rules involve the evaluation of the intersection of boundaries, the predicate always results in *false* when at least one of the operands is of type *vpoint2D* since the boundary of a *point2D* object is defined as being empty. For other operand types, it will certainly result in *false* if the interiors of the kernel parts of the objects intersect or the objects are certainly disjoint. For all other configurations it is not possible to certainly say whether the objects meet or not, i.e.,

$$\textbf{meet}(A,B) = \begin{cases} true & \text{if } \partial A_k \cap \partial B_k \neq \emptyset \wedge (A_k \oplus A_c)^\circ \\ & \cap (B_k \oplus B_c)^\circ = \emptyset \\ false & \text{if } A_k^\circ \cap B_k^\circ \neq \emptyset \vee \textbf{disjoint}(A,B) = true \\ & \vee (\partial(A_k \oplus A_c) \cap \partial(B_k \oplus B_c) = \emptyset \wedge \\ & \partial(A_k \oplus A_c) \cap (B_k \oplus B_c)^\circ = \emptyset \wedge \\ & \partial(A_k \oplus A_c) \cap (B_k \oplus B_c)^- = \emptyset) \vee \\ & (\partial(A_k \oplus A_c) \cap \partial(B_k \oplus B_c) = \emptyset \wedge \\ & (A_k \oplus A_c)^\circ \cap \partial(B_k \oplus B_c) = \emptyset \wedge \\ & (A_k \oplus A_c)^- \cap \partial(B_k \oplus B_c) = \emptyset) \\ maybe & \text{otherwise} \end{cases}$$

For the clustered vague topological predicate **inside**, a vague spatial object is certainly inside another vague spatial object if the interior of the former object is contained in the interior of the kernel part of the latter object and if there are no boundary intersections. The former object is certainly not inside the latter object if the interior of its kernel part intersects the exterior of the other object, or the interiors of both objects do not intersect, or there is a boundary intersection. In all other cases, we cannot make a definite decision due to the existence of the conjecture parts. We obtain:

$$\textbf{inside}(A,B) = \begin{cases} true & \text{if } (A_k \oplus A_c)^\circ \subset B_k^\circ \wedge \\ & \partial(A_k \oplus A_c) \cap \partial(B_k \oplus B_c) = \emptyset \wedge \\ & \partial A_k \cap \partial(B_k \oplus B_c) = \emptyset \wedge \\ & \partial(A_k \oplus A_c) \cap \partial B_k = \emptyset \wedge \\ & \partial A_k \cap \partial B_k = \emptyset \wedge \\ & (A_k \oplus A_c)^- \cap B_k^\circ \neq \emptyset \\ false & \text{if } A_k^\circ \cap (B_k \oplus B_c)^- \neq \emptyset \vee \\ & (A_k \oplus A_c)^\circ \cap (B_k \oplus B_c)^\circ = \emptyset \vee \\ & (\partial A_k \cap \partial B_k \neq \emptyset \wedge \\ & \partial A_k \cap \partial(B_k \oplus B_c) \neq \emptyset) \\ maybe & \text{otherwise} \end{cases}$$

Note that a subset relationship of the kind $X^\circ \subset Y^\circ$ is equivalent to the expression $X^\circ \cap Y^\circ \neq \emptyset \wedge \partial X \cap Y^\circ \neq \emptyset \wedge X^- \cap Y^\circ \neq \emptyset$. We define the clustered vague topological predicate **contains** as the converse of **inside**, i.e., **contains**$(A,B) =$ **inside**(B,A).

Now, we define the clustered vague topological predicate **coveredBy**. A vague spatial object is certainly covered by another vague spatial object if the interior of the former object is contained in the interior of the kernel part of the latter object and the boundary of the kernel part of the former object partially coincides with the boundary of the kernel part of the latter object and is otherwise located in the interior of the kernel part of the latter object. The former object is certainly not covered by the latter object if the interior of its kernel part intersects the exterior of the latter object or does not intersect the interior of the latter object, or if the former object is certainly inside the latter object. In all other cases, the predicate results in *maybe* due to the existence of the

conjecture parts. That is,

$$\textbf{coveredBy}(A,B) = \begin{cases} \mathit{true} & \text{if } (A_k \oplus A_c)^\circ \subset B_k^\circ \wedge \partial A_k \cap \partial B_k \neq \emptyset \wedge (A_k \oplus A_c)^- \cap B_k^\circ \neq \emptyset \\ & \quad \wedge A_k^\circ \cap B_k^- = \emptyset \wedge \partial A_k \\ & \quad \cap \partial(B_k \oplus B_c) \neq \emptyset \\ \mathit{false} & \text{if } A_k^\circ \cap (B_k \oplus B_c)^- \neq \emptyset \vee A_k^\circ \\ & \quad \cap (B_k \oplus B_c)^\circ = \emptyset \\ & \quad \vee \textbf{inside}(A,B) = \mathit{true} \\ \mathit{maybe} & \text{otherwise} \end{cases}$$

The predicate **covers** is defined as the converse of **coveredBy**, i.e., $\textbf{covers}(A,B) = \textbf{coveredBy}(B,A)$.

For the clustered vague topological predicate **equal**, we consider two vague spatial objects certainly equal only if their kernel parts are equal and their conjecture parts are empty. Two objects are certainly not equal if the interior of the kernel part of one object intersects the exterior of the other object. Otherwise, it is not possible to certainly determine whether the objects are equal, That is,

$$\textbf{equal}(A,B) = \begin{cases} \mathit{true} & \text{if } A_k^\circ \cap B_k^\circ \neq \emptyset \wedge \partial A_k \cap \partial B_k \neq \emptyset \wedge \\ & \quad A_k^- \cap B_k^- \neq \emptyset \wedge A_k^\circ \cap \partial B_k = \emptyset \wedge \\ & \quad A_k^\circ \cap B_k^- = \emptyset \wedge \partial A_k \cap B_k^\circ = \emptyset \wedge \\ & \quad \partial A_k \cap B_k^- = \emptyset \wedge A_k^- \cap \partial B_k = \emptyset \wedge \\ & \quad A_k^- \cap B_k^\circ = \emptyset \wedge A_k^\circ \cap \partial(B_k \oplus B_c) = \emptyset \wedge \\ & \quad A_k^\circ \cap (B_k \oplus B_c)^- = \emptyset \wedge \partial A_k \cap (B_k \oplus B_c)^\circ = \emptyset \wedge \\ & \quad \partial A_k \cap (B_k \oplus B_c)^- = \emptyset \wedge A_k^- \cap \partial(B_k \oplus B_c) = \emptyset \wedge \\ & \quad A_k^- \cap (B_k \oplus B_c)^\circ = \emptyset \wedge (A_k \oplus A_c)^\circ \cap \partial B_k = \emptyset \wedge \\ & \quad (A_k \oplus A_c)^\circ \cap B_k^- = \emptyset \wedge \partial(A_k \oplus A_c) \cap B_k^\circ = \emptyset \wedge \\ & \quad \partial(A_k \oplus A_c) \cap B_k^- = \emptyset \wedge (A_k \oplus A_c)^- \cap \partial B_k = \emptyset \wedge \\ & \quad (A_k \oplus A_c)^- \cap B_k^\circ = \emptyset \wedge \\ & \quad (A_k \oplus A_c)^\circ \cap \partial(B_k \oplus B_c) = \emptyset \wedge \\ & \quad (A_k \oplus A_c)^\circ \cap (B_k \oplus B_c)^- = \emptyset \wedge \\ & \quad \partial(A_k \oplus A_c) \cap (B_k \oplus B_c)^\circ = \emptyset \wedge \\ & \quad \partial(A_k \oplus A_c) \cap (B_k \oplus B_c)^- = \emptyset \wedge \\ & \quad (A_k \oplus A_c)^- \cap \partial(B_k \oplus B_c) = \emptyset \wedge \\ & \quad (A_k \oplus A_c)^- \cap (B_k \oplus B_c)^\circ = \emptyset \\ \mathit{false} & \text{if } A_k^\circ \cap (B_k \oplus B_c)^- \neq \emptyset \vee \\ & \quad (A_k \oplus A_c)^- \cap B_k^\circ \neq \emptyset \\ \mathit{maybe} & \text{otherwise} \end{cases}$$

Finally, we determine the clustered vague topological predicate **overlap**. Two vague spatial objects certainly overlap if the interiors of the kernel parts of both objects intersect each other and if the interior of each kernel part intersects the exterior of the other object. The objects certainly do not **overlap** if their interiors do not intersect, or if their interiors do not intersect the exterior of the kernel part of the other object, i.e.,

$$\textbf{overlap}(A,B) = \begin{cases} \mathit{true} & \text{if } A_k^\circ \cap B_k^\circ \neq \emptyset \wedge A_k^\circ \cap (B_k \oplus B_c)^- \neq \emptyset \wedge \\ & \quad (A_k \oplus A_c)^- \cap B_k^\circ \neq \emptyset \\ \mathit{false} & \text{if } (A_k \oplus A_c)^\circ \cap B_k^- = \emptyset \vee \\ & \quad (A_k \oplus A_c)^\circ \cap (B_k \oplus B_c)^\circ = \emptyset \vee \\ & \quad A_k^- \cap (B_k \oplus B_c)^\circ = \emptyset \\ \mathit{maybe} & \text{otherwise} \end{cases}$$

For the other cases 2, 3, 5, 6, 7, 9, 10, and 11 of Table 3, we have specified similar collections of interpretation rules for the same eight vague topological predicates as above. But due to space limitations, we omit the description of these collections here. Note that case 6 describes the special case where only the kernel parts of both vague spatial objects exist. Here, the interpretation rules coincide with the clustering rules of the corresponding eight crisp predicates in [40].

The second element of our strategy matches *all* characterization predicates of *all* type combinations for *all* cases against the eight clustered vague topological predicates. This leads to 54 *interpretation tables* since we have six type combinations multiplied by nine cases in Table 5. As an example, we take the *point2D*/*point2D* type combination and deal with case 1. Table 7 shows an excerpt of the corresponding interpretation table of 46 characterization predicates. Column 1 lists the numbers of the characterization predicates. Columns 2–7 show some of the possible combinations of crisp topological predicates between two complex *point2D* objects such that their conjunction is a valid characterization predicate (compare to Table 3, case 1). The next eight columns show which truth values each characterization predicate returns with respect to the interpretation rules of all eight clustered vague topological predicates. The characterization predicates 1 and 46 (besides the non-shown predicates 23 and 35) describe the unambiguous situations that exactly one vague topological predicate (**disjoint**, **overlap**) yields true while all other vague topological predicates consequently yield false. Note that definite equality is impossible since the conjecture parts of both vague objects are assumed to be non-empty. The characterization predicates 8–11 reveal that if a vague topological predicate returns maybe, then the other predicates can only return maybe or false. The case that all vague topological predicates return false does not appear. Further, these four characterization predicates demonstrate the feature of an *interpretation cluster*. That is, each vague topological predicate yields the same truth value for them. All the other interpretation tables are constructed in a similar way.

The remaining question is how the truth values in the interpretation tables are determined. For this purpose, we analyze the entries of the 9-intersection matrices (Section 2.1) that uniquely represent the characterization predicate's underlying crisp topological predicates p, q, r, s, v, and w. For example, the 9-intersection matrix entries for the crisp topological predicate *disjoint* between two complex *point2D* objects are given in Fig. 14(b) by evaluating the matrix predicates of the 9-intersection matrix in Fig. 14(a). Each term used in the interpretation rules of the eight clustered vague topological predicates corresponds to one of these matrix predicates (inequalities) or their negation. We first give a few examples for the characterization predicate 1 in Table 7. In this case, $p = q = r = s = \mathit{disjoint}$ and $v = w = \mathit{inside}$ holds. The predicates v and w do not play a role for the interpretation rules since we only consider intersections between components of different vague spatial objects. A check of the true condition of **disjoint** shows that each of its terms is true according to Fig. 14(b). Hence, their conjunction is true. Next, we check the true condition of **overlap**. The

Table 7
Excerpt of the interpretation table for the *point2D*/*point2D* type combination regarding case 1. The two-letter codes are taken from Table 4.

	$p(A_k, B_k)$	$q(A_k \oplus A_c, B_k)$	$r(A_k, B_k \oplus B_c)$	$s(A_k \oplus A_c, B_k \oplus B_c)$	$v(A_k, A_k \oplus A_c)$	$w(B_k, B_k \oplus B_c)$	**disjoint**	**meet**	**coveredBy**	**covers**	**inside**	**contains**	**equal**	**overlap**
1	*di*	*di*	*di*	*di*	*in*	*in*	*t*	*f*	*f*	*f*	*f*	*f*	*f*	*f*
2	*di*	*di*	*di*	*ov*	*in*	*in*	*m*	*f*	*f*	*f*	*f*	*f*	*f*	*m*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
8	*di*	*co*	*in*	*eq*	*in*	*in*	*m*	*f*	*m*	*m*	*m*	*m*	*m*	*m*
9	*di*	*co*	*in*	*in*	*in*	*in*	*m*	*f*	*m*	*m*	*m*	*m*	*m*	*m*
10	*di*	*co*	*in*	*co*	*in*	*in*	*m*	*f*	*m*	*m*	*m*	*m*	*m*	*m*
11	*di*	*co*	*in*	*ov*	*in*	*in*	*m*	*f*	*m*	*m*	*m*	*m*	*m*	*m*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
46	*ov*	*ov*	*ov*	*ov*	*in*	*in*	*f*	*f*	*f*	*f*	*f*	*f*	*f*	*t*

The letters *t*, *f*, and *m* stand for the truth values *true*, *false*, and *maybe* respectively.

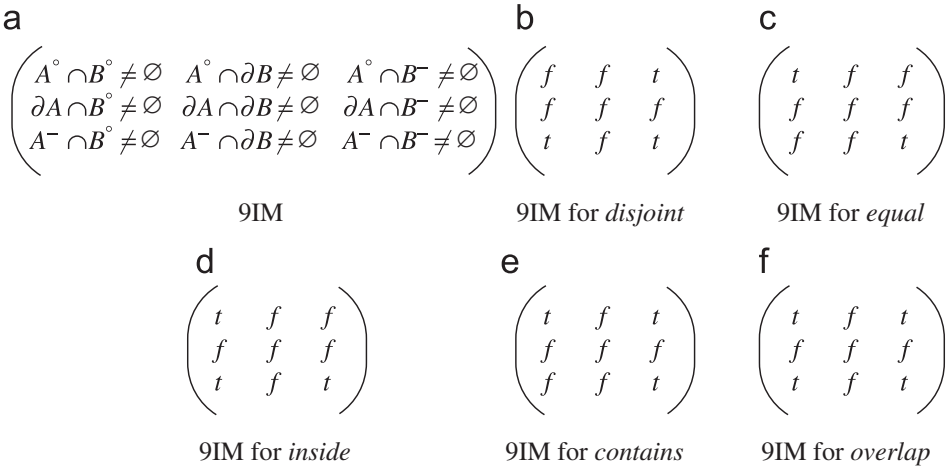


Fig. 14. The 9-intersection matrix (9IM) with its matrix predicates (a) and the corresponding matrices for the five crisp topological predicates between two complex *point2D* objects ((b) to (f)).

first term already leads to a mismatch. However, the false condition of this predicate holds since the second term of the disjunction returns true. Similarly, we find that the false conditions of the other six clustered vague topological predicates hold.

As another example, we consider characterization predicate 8 in Table 7 for which we have $p(A_k, B_k) = disjoint(A_k, B_k)$, $q(A_k \oplus A_c, B_k) = contains(A_k \oplus A_c, B_k)$, $r(A_k, B_k \oplus B_c) = inside(A_k, B_k \oplus B_c)$, and $s(A_k \oplus A_c, B_k \oplus B_c) = equal(A_k \oplus A_c, B_k \oplus B_c)$ (note again that v and w do not have to be considered) together with the 9-intersection matrices for these predicates shown in Fig. 14(b)–(e) and taken from [40]. As an example for evaluating the eight clustered vague topological predicates, we take the predicate **disjoint**. The true condition first requires that $(A_k \oplus A_c)^\circ \cap (B_k \oplus B_c)^\circ = \emptyset$. The crisp topological predicate with the arguments $A_k \oplus A_c$ and $B_k \oplus B_c$ is $s = equal$. Hence, we lookup the value of the matrix predicate for the interior/interior case in the 9-intersection matrix of *equal*. In Fig. 14(c) we find the value *t* which contradicts the first term of the true condition so that this condition (conjunction) does not hold. The false condition refers to A_k and B_k for all of its terms so that we have to compare to the crisp topological predicate $p = disjoint$ having A_k and B_k as arguments. It turns out that none of the four terms is satisfied so that thefalse condition (disjunction) does also not hold. This means that the value maybe is the result of **disjoint**. Similarly, all the other predicates are evaluated; they yield maybe except for **meet** that yields false. In this way, for all type combinations, we can interpret all characterization predicates regarding all eight clustered vague topological predicates and regarding all cases.

6. Implementing and querying vague spatial objects in databases

Section 6.1 sketches our implementation of VASA. Section 6.2 demonstrates the practical use and integration of our VASA concepts into the relational data model and into a proposed extension of the SQL-like query language.

6.1. Implementation of VASA

In this article, we have so far presented VASA from the perspective of a descriptive algebra (see Section 3). That is, we propose what the relevant vague spatial data types, operations, and predicates are and what their semantics is but we do not specify how these concepts are implemented. However, we have here the rare case that VASA can be

viewed as both a descriptive algebra *and* an executable algebra. The reason is that our vague spatial data types, operations, and predicates are all based on their crisp counterparts and can be expressed exclusively in terms of them. Hence, we can leverage preexisting implementations of crisp spatial type systems (algebras) like the commercial implementations mentioned in Section 2.1 and implement VASA on top of them with only minimal effort. The idea is to represent vague spatial data types on the basis of crisp spatial data types and *execute* vague geometric set operations and predicates according to their definitions on their crisp counterparts without being forced to design and implement new algorithms. In other words, we obtain *executable specifications* that can be directly deployed as an implementation.

In more detail, each of VASA's three data types *vpoint2D*, *vline2D*, and *vregion2D* is defined as a pair of its crisp counterparts *point2D*, *line2D*, and *region2D*, respectively. All operations and predicates implemented for a vague data type work by calling methods implemented for their underlying crisp data type. As an example, we remember the definition of **union** from Definition 4(i):

$$u \textbf{ union } w := (u_k \oplus w_k, (u_c \oplus w_c) \ominus (u_k \oplus w_k))$$

The code sample in Fig. 15 illustrates the implementation of **union** between two *vregion2D* objects.Lines 2 to 5 are in charge of exactly following the executable specification of this operation by applying the crisp operations **cunion** and **cdifference** to crisp kernel and conjecture *region2D* objects.

The evaluation of vague topological predicates is a little different. Given two vague spatial objects, we first determine the crisp topological predicates of the characterization predicate depending on the type combination and the case (Table 3). This requires *predicate determination techniques* that we have developed in [36] and that answer the question which (unique) topological relationship exists between two given crisp spatial objects. Afterwards, we take the characterization predicate found as input and look up the truth value for the vague topological predicate of interest in the corresponding interpretation table (Table 7).

The connection between VASA and a DBMS is enabled through data type plug-in extension mechanisms available

```
    vregion2D vregion2D::union(const vregion2D &other) const
    {
        /* Declare a variable to store the result */
1       vregion2D result;

        /* Build the kernel part of the result from the conceptual definition */
2       result.setKernel(getKernel().cunion(other.getKernel()));

        /* Build the conjecture part of the result */
3       region2D temp1 = getConjecture().cunion(other.getConjecture());
4       region2D temp2 = getKernel().cunion(other.getKernel());
5       result.setConjecture(temp1.cdifference(temp2));

6       return result;
    }
```

Fig. 15. Code sample that illustrates the implementation of VASA and how it interacts with a crisp spatial type system to perform the **union** operation between two vague region objects.

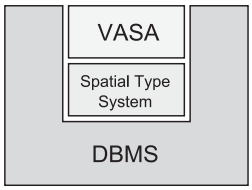


Fig. 16. Plug-in integration of VASA on top of a crisp spatial type system into an extensible DBMS.

in most DBMS like Informix's Data Blades, Oracle's Data Cartridges, and DB2's Extenders (see Fig. 16). They allow user defined data types to be specified or registered with a DBMS. Along with the data type specifications, all their operations must be defined and registered. This enables a DBMS to make the correct operation call when a query attempts to execute an operation on an object of a user defined data type. The operations themselves are implemented as external methods in a library. In our prototype implementation in Oracle, we have, e.g., registered *vregion2D* as a new vague spatial data type and specified it as a pair of *region2D* compatible data types of Oracle Spatial. We have also registered the signature of all operations on vague regions and determined where the code of those operations is located. In our case, the execution code lies in an external, shared library written in C ++.

6.2. Querying vague spatial objects

A first issue refers to the embedding of vague spatial data types into relational database schemas. Our strategy is to use these types in the same way as attribute data types as standard data types like *integer*, *float*, or *string*. The main difference is the complex internal structure of the vague types that we deliberately hide from the user. That is, we embed the vague spatial data types as *abstract data types* into relational schemas. Information about vague spatial objects can only be obtained by operations and predicates and not by direct access to object components. For example, we may specify the relation schema weather(climate: *string*; area: *vregion2D*) where the column named *area* contains vague region values for climatic conditions indicated by the column *climate*. The types *string* and *vregion2D* are used in the same way.

A second issue relates to the fact that SQL predicates are Boolean expressions only. Hence, the three truth values true, false, and maybe of our three-valued logic have to be translated to the Boolean logic that SQL understands in order to leverage vague topological predicates in SQL. As a solution, we assign the three Boolean predicates **true_p**, **maybe_p**, and **false_p** to each clustered vague topological predicate **p**:

$$\begin{aligned}
&\textbf{true_p}(A,B) && = \mathit{true} \Rightarrow \mathbf{p}(A,B) = \mathit{true}\\
&\textbf{true_p}(A,B) && = \mathit{false} \Rightarrow \mathbf{p}(A,B) = \mathit{maybe} \vee \mathbf{p}(A,B) = \mathit{false}\\
&\textbf{maybe_p}(A,B) && = \mathit{true} \Rightarrow \mathbf{p}(A,B) = \mathit{maybe}\\
&\textbf{maybe_p}(A,B) && = \mathit{false} \Rightarrow \mathbf{p}(A,B) = \mathit{true} \vee \mathbf{p}(A,B) = \mathit{false}\\
&\textbf{false_p}(A,B) && = \mathit{true} \Rightarrow \mathbf{p}(A,B) = \mathit{false}\\
&\textbf{false_p}(A,B) && = \mathit{false} \Rightarrow \mathbf{p}(A,B) = \mathit{true} \vee \mathbf{p}(A,B) = \mathit{maybe}
\end{aligned}$$

Note that $\neg$**true_p**$(A,B) \neq$ **false_p**(A,B). In the following, we present several simplified scenarios that illustrate VASA concepts and their possible embedding into SQL-like queries.

6.2.1. Scenario 1: Homeland security

Secret services are interested in locations of terroristic activity (see also Sections 3.1 and 4.1). We store information about terrorists in the following relation:

terrorist(id: *integer*, name: *string*, refuge: *vpoint2D*, route: *vline2D*, active_area: *vregion2D*)

The attribute *refuge* models all locations that a terrorist has definitely or possibly used as a refuge. The routes a terrorist has definitely and possibly taken to move between refuges are modeled in the attribute *route*. Areas of potential terroristic activity are stored in the attribute *active_area*.

The first query asks for the locations where any two terrorists have definitely or possibly taken the same refuge. It demonstrates the use of vague topological predicates and vague spatial operations (**intersection**).

select A.id, B.id, **intersection**(A.refuge, B.refuge)
from terrorist **as** A, terrorist **as** B
where A.id $\neq$ B.id **and** A.refuge (**true_overlap** | **maybe_overlap**) B.refuge

We have introduced the notation $A(\mathbf{p_1}|\mathbf{p_2}|...|\mathbf{p_n})B$ for $\mathbf{p_1}(A,B)$ **or** $\mathbf{p_2}(A,B)$ **or** ... **or** $\mathbf{p_n}(A,B)$. We obtain the refuges that have definitely (perhaps) been taken by two terrorists by omitting the predicate **maybe_overlap** (**true_overlap**).

The following query determines the names of terrorists and the locations where their routes have definitely or possibly crossed each other. The expression (**0**, **0**) is a generic type constructor for the empty vague spatial object (here empty vague point).

select A.name, B.name, **common_points**(A.route, B.route) **as** crossing
from terrorist **as** A, terrorist **as** B
where A.id $\neq$ B.id **and common_points**(A.route, B.route) $\neq$ (**0**, **0**)

The next query finds out the known sphere of all terrorists on the basis of their refuges.

select **k-convex_hull**(**sum**(refuge)) **from** terrorist

The function **sum** is an overloaded spatial aggregation function that here computes the geometric union of a collection of vague point objects of the column *refuge*.

The final query checks whether the area of the convex hull computed in the previous query really fully belongs to the area of activity of all terrorists.

select **difference**(**k-convex_hull**(**sum**(refuge)), **sum**(active_area)) = (**0**, **0**) **as** empty
from terrorist

The query subtracts all areas of terroristic activity from the convex hull. If the convex hull is completely covered, the difference yields the empty vague region, and the attribute *empty* obtains the value *true*.

6.2.2. Scenario 2: Ecological application

This scenario assumes an ecological database with the following relations:

weather(climate: *string*; area: *vregion2D*)
soil(quality: *string*, area: *vregion2D*)

The relation *weather* has a column named *area* containing vague region values for various climatic conditions given by the column *climate*. The relation *soil* describes the soil quality for certain regions.

The first query is supposed to find out all regions of bad ecological conditions, i.e., all locations where a lack of water or a bad soil quality is a hindrance for cultivation.

select **union**(dry_area, bad_soil) **as** bad_region
from **select sum**(area) **as** dry_area **from** weather **where** climate = "dry",
select sum(area) **as** bad_soil **from** soil **where** quality = "bad"

In the *from* clause, we create two temporary relations that contain the aggregated areas of dry climate and bad soil quality, respectively. Each relation contains a single tuple with a single attribute value of type *vregion2D*. In the *select* clause, we compute the union of the two attribute values of both tuples.

The next query determines the numerical area measure of those weather zones that we can definitely classify and that we can only vaguely classify.

select climate, **min_area**(**sum**(area)) **as** definite_area, **area**(**c-proj**(**sum**(area))) **as** vague_area
from weather

Note that the **area** operation is a crisp operation.

6.2.3. Scenario 3: Environmental application

Pollution is nowadays a central environmental problem and causes 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 assume an environmental database with the two self-explaining relations

pollution(type: *string*; zone: *vregion2D*)
land_use(use: *string*, area: *vregion2D*)

The first query asks for inhabitable areas which are air polluted (where the kernel part of an air polluted zone denotes heavily polluted areas and the conjecture part

only gives slightly polluted regions).

select **intersection**(**sum**(zone), **sum**(area))
from pollution, land_use
where use = "inhabited" **and** type = "air"

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

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

select **intersection**(**kernel**(**sum**(zone)), **sum**(area))
from ...

A slightly different query is to find out all areas where people are definitely or possibly endangered by pollution. It is an example of a *vague spatial join*.

select **sum**(**intersection**(zone, area)) **as** endangered_area
from pollution, land_use
where use = "inhabited" **and** zone (**true_overlap**| **maybe_overlap**) area

7. Conclusions and future work

In this article, we have dealt with the problem of spatial vagueness that is inherent to many database applications in the geosciences and in geographical information systems. Our solution is the design, formal definition, and implementation of the *vague spatial algebra* that can be embedded into any extensible and commercially or publicly available database system and its query language SQL. VASA is a type system that provides an object model for the two-dimensional *vague spatial data types* named *vpoint2D*, *vline2D*, and *vregion2D* together with a comprehensive collection of *vague spatial operations* and *vague topological predicates*. An essential feature of VASA is that it is an extension of crisp or determinate spatial data models. This enables a smooth migration from crisp to vague spatial concepts and facilitates their treatment in the same framework. Since our approach is based on exact spatial modeling concepts, it allows us to build upon existing work and simplifies many definitions. In particular, we can leverage already existing implementations of crisp spatial type systems to implement vague spatial objects, operations, and predicates with only minimal effort by *executable specifications*.

For future work, we consider the extension of the three-valued logic of VASA to a *many-valued logic* that distinguishes different *degrees of spatial vagueness*. However, including different degrees of spatial vagueness requires more and especially more precise knowledge about the distribution of uncertainty values. In this article, our assumption has been that this knowledge does not exist; this is often the case. We see at least two possible approaches to an extension. A first approach could lead to an extension of VASA in which a vague spatial object is described by a finite number of crisp spatial objects and each such crisp spatial object is annotated with its degree of spatial vagueness. We call these crisp spatial objects *spatial plateau objects* since a certain degree does not only hold for some point but also for infinitely many points in a connected neighborhood in case of lines and regions. A second approach could be to employ fuzzy logic and fuzzy set theory to describe *fuzzy spatial objects*. This would allow a continuous change of the degree of spatial vagueness also in the interior of such an object. A number of approaches has already been proposed in the literature (see Section 2.2). A further research issue is whether and how spatial vagueness exerts influence on spatial indexing. Current spatial index structures assume crisply defined spatial objects. The question is whether *vague spatial index structures* are needed.

References

[1] D. Altman, Fuzzy set theoretic approaches for handling imprecision in spatial analysis, International Journal of Geographical Information Systems 8 (3) (1994) 271–289.
[2] M. Blakemore, Generalization and error in spatial databases, Cartographica 21 (1984).
[3] P.A. Burrough, Natural objects with indeterminate boundaries, in: P.A. Burrough, A.U. Frank (Eds.), Geographic Objects with Indeterminate Boundaries, GISDATA Series, vol. 2, Taylor & Francis, London, 1996, pp. 3–28.
[4] P.A. Burrough, A.U. Frank (Eds.), Geographic Objects with Indeterminate Boundaries, GISDATA Series, vol. 2, Taylor & Francis, London, 1996.
[5] E. Clementini, P. Di Felice, A model for representing topological relationships between complex geometric features in spatial databases, Information Sciences 90 (1–4) (1996) 121–136.
[6] E. Clementini, P. Di Felice, An algebraic model for spatial objects with indeterminate boundaries, in: P.A. Burrough, A.U. Frank (Eds.), Geographic Objects with Indeterminate Boundaries, GISDATA Series, vol. 2, Taylor & Francis, London, 1996, pp. 153–169.
[7] E. Clementini, P. Di Felice, A spatial model for complex objects with a broad boundary supporting queries on uncertain data, Data & Knowledge Engineering 37 (2001) 285–305.
[8] A.G. Cohn, N.M. Gotts, The 'egg-yolk' representation of regions with indeterminate boundaries, in: P.A. Burrough, A.U. Frank (Eds.), Geographic Objects with Indeterminate Boundaries, GISDATA Series, vol. 2, Taylor & Francis, London, 1996, pp. 171–187.
[9] A. Dilo, R.D. de By, A. Stein, A system of types and operators for handling vague spatial objects, International Journal of Geographical Information Science 21 (4) (2007) 397–426.
[10] M.J. Egenhofer, Reasoning about binary topological relations, in: Second International Symposium on Advances in Spatial Databases, Lecture Notes in Computer Science, vol. 525, Springer, Berlin, 1991, pp. 143–160.
[11] M.J. Egenhofer, Spatial SQL: a query and presentation language, IEEE Transactions on Knowledge and Data Engineering 6 (1) (1994) 86–94.
[12] M.J. Egenhofer, R.D. Franzosa, Point-set topological spatial relations, International Journal of Geographical Information Systems 5 (2) (1991) 161–174.
[13] M.J. Egenhofer, J. Herring, Categorizing binary topological relations between regions, lines, and points in geographic databases, Technical Report 90-12, National Center for Geographic Information and Analysis, University of California, Santa Barbara, 1990.
[14] M. Erwig, M. Schneider, Vague Regions, in: Fifth International Symposium on Advances in Spatial Databases, Lecture Notes in Computer Science, vol. 1262, Springer, Berlin, 1997, pp. 298–320.
[15] ESRI Spatial Database Engine (SDE), Environmental Systems Research Institute, Inc., 1995.

[16] J.T. Finn, Use of the average mutual information index in evaluating classification error and consistency, International Journal of Geographical Information Systems 7 (4) (1993) 349–366.
[17] R.H. Güting, Geo-relational algebra: a model and query language for geometric database systems, in: International Conference on Extending Database Technology, 1988, pp. 506–527.
[18] R.H. Güting, M. Schneider, Realm-based spatial data types: the ROSE algebra, VLDB Journal 4 (1995) 100–143.
[19] IBM, Informix geodetic datablade module: user's guide, 2002.
[20] IBM, DB2 spatial extender and geodetic data management feature—user's guide and reference, 2006.
[21] International Standard Organization, ISO 19107: geographic information—spatial schema, 2003.
[22] V.J. Kollias, A. Voliotis, Fuzzy reasoning in the development of geographical information systems, International Journal of Geographical Information Systems 5 (2) (1991) 209–223.
[23] P. Lagacherie, P. Andrieux, R. Bouzigues, Fuzziness and uncertainty of soil boundaries: from reality to coding in GIS, in: P.A. Burrough A.U. Frank (Eds.), Geographic Objects with Indeterminate Boundaries, GISDATA Series, vol. 2, Taylor & Francis, London, 1996, pp. 275–286.
[24] A. Mackworth, Consistency in networks of relations, Artificial Intelligence 8 (1977) 99–118.
[25] R. Maddux, Some algebras and algorithms for reasoning about time and space, Technical Report, Department of Mathematics, Iowa State University, 1990.
[26] U. Montanari, Network of constraints: fundamental properties and applications to picture processing, Information Sciences 7 (1974) 95–132.
[27] R.E. Moore, Interval Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1966.
[28] Open Geospatial Consortium Incorporation, OpenGIS implementation specification for geographic information—simple feature access—part 1: common architecture, 2006.
[29] Oracle Corporation, Oracle spatial user's guide and reference 10g release 2, 2005.
[30] J.A. Orenstein, F.A. Manola, PROBE spatial data modeling and query processing in an image database application, IEEE Transactions on Software Engineering 14 (1988) 611–629.
[31] A. Pauly, M. Schneider, Vague spatial data types, set operations, and predicates, in: Eighth East-European Conference on Advances in Databases and Information Systems, 2004, pp. 379–392.
[32] A. Pauly, M. Schneider, Identifying topological predicates for vague spatial objects, in: 20th ACM Symposium on Applied Computing, 2005, pp. 587–591.
[33] A. Pauly, M. Schneider, Topological predicates between vague spatial objects, in: Ninth International Symposium on Spatial and Temporal Databases, 2005, pp. 418–432.
[34] A. Pauly, M. Schneider, Topological reasoning for identifying a complete set of topological predicates between vague spatial objects, in: 19th Florida Artificial Intelligence Research Society Conference, 2006, pp. 731–736.
[35] PostGIS, PostGIS 1.3.5 manual, 2009.
[36] R. Praing, M. Schneider, Efficient implementation techniques for topological predicates on complex spatial objects, GeoInformatica 12 (3) (2008) 313–356.
[37] M. Schneider, Modelling spatial objects with undetermined boundaries using the realm/ROSE approach, in: P.A. Burrough, U.A. Frank (Eds.), Geographic Objects with Indeterminate Boundaries, GISDATA Series, vol. 2, Taylor & Francis, London, 1996, pp. 141–152.
[38] M. Schneider, Spatial Data Types for Database Systems—Finite Resolution Geometry for Geographic Information Systems, Lecture Notes in Computer Science, vol. 1288, Springer, Berlin, Heidelberg, 1997.
[39] M. Schneider, Uncertainty management for spatial data in databases: fuzzy spatial data types, in: Sixth International Symposium on Advances in Spatial Databases, Lecture Notes in Computer Science, vol. 1651, Springer, Berlin, 1999, pp. 330–351.
[40] M. Schneider, T. Behr, Topological relationships between complex spatial objects, ACM Transactions on Database Systems 31 (1) (2006) 39–81.
[41] R. Shibasaki, A Framework for Handling Geometric Data with Positional Uncertainty in a GIS Environment, GIS: Technology and Applications, World Scientific, Singapore, 1993, pp. 21–35.
[42] A. Tarski, On the calculus of relations, Journal of Symbolic Logic 6 (3) (1941) 73–89.
[43] Vivid Solutions, JTS topology suite: technical specifications, 2003.
[44] F. Wang, G.B. Hall, Fuzzy representation of geographical boundaries in GIS, International Journal of Geographical Information Systems 10 (5) (1996) 573–590.
[45] F. Wang, G.B. Hall, Subaryono, Fuzzy information representation and processing in conventional GIS software: database design and application, International Journal of Geographical Information Systems 4 (3) (1990) 261–283.
[46] M. Worboys, Computation with imprecise geospatial data, computational, Environmental and Urban Systems 22 (2) (1998) 85–106.
[47] M.F. Worboys, P. Bofakos, A canonical model for a class of areal spatial objects, in: Third International Symposium on Advances in Spatial Databases, Lecture Notes in Computer Science, vol. 692, Springer, Berlin, 1993, pp. 36–52.
[48] D. Zinn, J. Bosch, M. Gertz, Modeling and querying vague spatial objects using shapelets, in: International Conference on Very Large Data Bases, 2007, pp. 567–578.