Spatial Plateau Algebra for Implementing Fuzzy Spatial Objects in Databases and GIS: Spatial Plateau Data Types and Operations

Markus Schneider*
University of Florida
Department of Computer & Information Science & Engineering
Gainesville, FL 32611, USA
Email: mschneid@cise.ufl.edu, Phone: +1-352-505-1584

Abstract

Many geographical applications have to deal with spatial objects that reveal an intrinsically vague or fuzzy nature. A spatial object is fuzzy if locations exist that cannot be assigned completely to the object or to its complement. Spatial database systems and Geographical Information Systems (GIS) are currently unable to cope with this kind of data. Based on an available abstract data model of *fuzzy spatial data types* for *fuzzy points*, *fuzzy lines*, and *fuzzy regions* that leverages fuzzy set theory and fuzzy point set topology, this article proposes a *Spatial Plateau Algebra* that provides *spatial plateau data types* as an implementation of fuzzy spatial data types. Each *spatial plateau object* consists of a finite number of crisp counterparts that are all adjacent or disjoint to each other, are associated with different membership values, and hence form different *plateaus*. The formal framework and the implementation are based on well known, exact models and implementations of crisp spatial data types. *Spatial plateau operations* as geometric operations on spatial plateau objects are expressed as a combination of geometric operations on the underlying crisp spatial objects. This article offers a conceptually clean foundation for implementing a database extension for fuzzy spatial objects and their operations, and demonstrates the embedding of these new data types as attribute data types in a database schema as well as the incorporation of fuzzy spatial operations into a database query language.

Keywords: Spatial fuzziness; spatial vagueness; fuzzy point object; fuzzy line object; fuzzy region object; plateau point object, plateau line object; plateau region object; spatial plateau operation; executable specification

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

1 Introduction

Spatial databases as the data management foundation of Geographical Information Systems (GIS) represent geometries such as point, line, and region objects by special data types called *spatial data types* [41]. These data types can be used in the same way as attribute data types as integers, floats, or characters. Their objects have the fundamental and tacitly assumed property that they are crisp, that is, they have a definite extent, boundary, and shape. However, many, or even most, spatial objects in reality cannot be described by crisp concepts since they are inherently fuzzy, vague, or indeterminate. A spatial object is fuzzy if it contains locations that cannot be assigned completely to the object or to its complement. Hence, *spatial fuzziness* captures the property of objects that do not have sharp boundaries and crisp interiors but rather vague or indeterminate boundaries and interiors. Examples are natural, social, or cultural phenomena like oceans, pollution areas, and English speaking regions. It is, for example, impossible to say with precision where the Indian Ocean ends and the Arabian Sea begins since the transition between them is smooth and seamless. So far, available spatial database systems and GIS are unable to represent indeterminate spatial objects.

From a modeling standpoint, especially the GIS community has propagated the use of *fuzzy set theory* [56] to characterize and describe indeterminate spatial data. The spatial database community has produced a few approaches to the conceptual modeling of such data through *fuzzy spatial data types*. In previous work [42], at an abstract, fuzzy point set theoretic and fuzzy point set topological level, we have defined *fuzzy points*, *fuzzy lines*, and *fuzzy regions* by assigning a *membership value* ranging from 0 to 1 to each point of such an object by an appropriately defined and application-specific *membership function*. A membership value indicates how strongly or weakly a point belongs to an object. From an implementation standpoint, adequate implementation approaches to the representation of fuzzy spatial data types in spatial database systems and GIS are lacking. A first main reason is that the sole approximation of the fuzzy boundary of a fuzzy spatial object is insufficient since the interior of such an object is usually fuzzy too. A second main reason is that each of the infinitely many points of the interior of a fuzzy spatial object can have a different membership value and that a *finite* representation of such a constellation is challenging and difficult.

The goal of this article is to propose an appropriate implementation concept for these types that enables their efficient representation, querying, and manipulation in a database context. Our approach introduces an *executable algebra* (or *type system*) called *Spatial Plateau Algebra* that bases the implementation of fuzzy spatial data types, operations, and predicates on a *plateau concept*. The idea is to provide *spatial plateau data types* that approximate a fuzzy point, fuzzy line, or fuzzy region object by a *plateau point*, *plateau line*, or *plateau region* object respectively. Each *spatial plateau object* consists of a finite number of crisp and finitely representable counterparts (that is, crisp point, line, or region objects) that are all adjacent or disjoint to each other and that are associated with different membership values which form *plateaus* and which determine the degree of belonging to the fuzzy spatial object. A main benefit of the Spatial Plateau Algebra is that its formal framework and its implementation are based on well known, general, and exact models and implementations of crisp spatial data types. This enables its own exact and robust definition and implementation. Geometric operations (like geometric union, intersection, and difference) on spatial plateau objects are called *spatial plateau operations* and are expressed as a combination of corresponding operations on the underlying crisp spatial objects. Hence, our approach enables *executable specifications* of the spatial plateau operations such that they can be immediately used as implementations. The spatial plateau data types are, in particular, closed under spatial plateau operations. Further, spatial plateau objects are represented as *compact storage structures* (arrays) that are directly processed by geometric algorithms, avoid any serialization and deserialization cost, and thus enable an efficient object transfer between main memory and database. This article offers a precise and conceptually clean foundation for implementing a database extension for fuzzy spatial objects and operations, and demonstrates the embedding of these new data types as attribute data types in a database schema as well as the incorporation of fuzzy spatial operations into queries formulated in a database query language. Spatial plateau predicates implementing fuzzy topological

predicates and fuzzy directional predicates as well as metric plateau operations implementing fuzzy metric operations are not considered in this article.

Section 2 discusses related work on the fuzzy approach to spatial data handling, both from a conceptual and an implementation perspective. It also briefly reviews crisp spatial data types as the underlying basis of our Spatial Plateau Algebra. Section 3 introduces spatial plateau objects as an implementation concept for fuzzy spatial objects and provides a formal definition of the spatial plateau data types. Section 4 focuses on the specification of spatial plateau operations for their fuzzy counterparts in terms of their crisp counterparts. An emphasis is on the plateau versions of the fuzzy geometric set operations *fuzzy union*, *fuzzy intersection*, and *fuzzy difference*. These versions are named *plateau union*, *plateau intersection*, and *plateau difference*. Fuzziness can be expressed in different ways. Hence, beside an established and predefined understanding of fuzziness for our spatial plateau operations, we also present a variation of these operations that is context-dependent and lets the application or user select an appropriate interpretation of fuzziness. Section 5 shows a few application scenarios and queries that leverage the operations of the Spatial Plateau Algebra. Section 6 discusses implementation aspects of the Spatial Plateau Algebra. Section 7 draws some conclusions and considers future work.

2 Related Work

In this section, we discuss related work that has been proposed for modeling fuzzy spatial objects and that is relevant for describing our implementation concept based on crisp spatial data types. As indicated in the Introduction, the goal of this article is to propose an implementation concept for fuzzy spatial data types in terms of spatial plateau data types that themselves are based on well known crisp and complex spatial data types for point, line, and region objects. Correspondingly, fuzzy spatial operations are implemented in terms of spatial plateau operations that themselves are based on well known crisp spatial operations. Hence, in Section 2.1, we first delineate the available models of fuzzy spatial objects. Section 2.2 deals with implementation approaches to fuzzy spatial objects. Implementations are missing so far and only available for indeterminate spatial objects that are based on a 3-valued logic. Finally, Section 2.3 gives a concise overview of crisp spatial data types and operations as the foundation of our implementation concept.

2.1 Approaches to Modeling Fuzzy Spatial Objects

Fuzzy set-based models rest on *fuzzy set theory* [56], which describes the *admission of the possibility* (given by a *membership function*) that an individual is a full or partial member of a set. It refers to the vagueness resulting from the imprecision of the meaning of a concept. *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 and/or interior. Examples of *fuzzy spatial objects* include mountains, valleys, biotopes, polluted areas, and oceans, which cannot be rigorously bounded by a sharp line and which can have a blurred interior.

The deployment of fuzzy set theory and fuzzy logic for spatial applications and spatial analysis methods has gained increasing popularity in the geoscience and GIS communities and also led to a large amount of literature on fuzzy approaches in the GIS domain. A spatial concept is vague or fuzzy if locations exist that cannot be completely associated with the concept or with its complement. For example, when mapping vegetation, it is difficult to determine whether a certain location belongs to one vegetation class or to another. Examples from the geosciences that exhibit transition zones instead of sharp boundaries and that have been modeled with fuzzy concepts are geomorphological units [9], soil types and boundaries [16, 34], landscape objects [11], forest types [4], and soil pollution classes in environmental applications [28]. The work in [7] introduces fuzzy geographical objects for modeling natural objects with indeterminate boundaries. Fuzzy spatial objects also play a role in fuzzy architectural spatial analysis [2] which applies fuzzy concepts to

architectural spatial planning and analysis and explores the spatial formation and architectural space intensity within architectural structures. The authors of [21, 22, 33] deal with qualitative spatial and temporal reasoning using fuzzy logic. The approaches in [23, 51] deal with fuzzy representations of geographical boundaries in GIS. The author in [47] defines concepts like core and boundary of a fuzzy region and gives examples of membership functions for modeling fuzzy regions. The approach in [19] classifies geographic objects according to the features of crispness and vagueness of their interior and boundary. Geographic objects can have a crisp interior and crisp boundary, a crisp interior and fuzzy boundary, a fuzzy interior and crisp boundary, or a fuzzy interior and fuzzy boundary. The work in [50] presents a fuzzy query approach in order to introduce more natural language expressions into GIS user interfaces. The authors in [52] propose a fuzzy relational data model for geographic information; each tuple is annotated with a membership value.

The references mentioned so far demonstrate the usefulness of fuzzy concepts in geoscience applications from a modeling standpoint. However, they do not deal with the issues of representing, storing, retrieving, and querying fuzzy spatial objects in *fuzzy spatial databases*. Increasingly, there is interest in pursuing these goals. The work in [1] presents fuzzy set theoretic approaches for handling imprecision in spatial analysis and introduces *fuzzy regions* as a binary relation on the domain of $\mathbb{N}^2$ ($\mathbb{N}$ denotes the set of natural numbers). Distance and directional metrics on fuzzy regions demonstrate their possible use in qualitative spatial analysis. The studies in [5, 6] introduce *fuzzy plane geometry* and provide alternative definitions for each of the three categories of fuzzy points, fuzzy lines, and fuzzy regions. One definition in each category is selected with respect to its capability to be visualized. Fuzzy points and fuzzy lines have an areal structure so that one could describe them as special kinds of fuzzy regions. The work in [57] proposes a concentric core-boundary model for simple fuzzy regions and assumes a monotonically decreasing membership function from the core towards the boundary. For the determination of topological relationships, α-cut level regions of a fuzzy region are used. The authors in [35] propagate the incorporation of fuzzy spatial data and operations into a fuzzy object-oriented data model and database. Fuzziness of spatial objects is modeled in a class hierarchy such that the membership of a class in its superclass need not be crisp. Once fuzzy membership values have been assigned to (spatial) objects, fuzzy (spatial) operations can be defined on them. The data structures and algorithms provided by the Spatial Plateau Algebra can help the authors implement their approach. The approach in [20] defines *fuzzy partitions* over the Euclidean plane to model fuzzy spatial objects and distinguishes their possibly overlapping fuzzy interior, fuzzy boundary, and fuzzy exterior. The study in [45] represents a fuzzy region by a core and a boundary that are approximated by their *fuzzy minimum bounding rectangles*. The work in [17] leverages *generalized constraints* introduced by Lofti Zadeh to cope with imperfect spatial information and relies on a many-valued logic based on extended possibilistic truth values.

The approach in [42] provides the foundation of *fuzzy spatial data types* for *fuzzy points*, *fuzzy lines*, and *fuzzy regions* and also specifies *fuzzy spatial operations* like *fuzzy spatial union*, *fuzzy spatial intersection*, and *fuzzy spatial difference*. The author presents an abstract specification in which fuzzy spatial objects are defined as special fuzzy sets over $\mathbb{R}^2$. A similar type system of *vague spatial data types* with a comprehensive set of operations is introduced in [18]. Instead of the term "fuzzy", the authors use the term "vague". The work in [46] offers a similar (informal) concept of fuzzy regions but different concepts of fuzzy points and fuzzy lines which have an areal structure too. The Spatial Plateau Algebra uses especially the models and concepts in [18, 42] as a specification for its implementation. With some modifications, our approach could also be applied to the kind of fuzzy spatial objects as they are defined in [46].

The work in [43] proposes a conceptual model and an implementation model of fuzzy spatial objects that are *not* defined on the Euclidean plane but on a discrete geometric domain called *grid partition*. It takes into account finite-precision number systems available in computers. Membership values are assigned to the points, edges, and cells as elements of the grid partition, and fuzzy objects are built from these grid elements.

An important requirement for a model of fuzzy spatial objects is that the fuzzy spatial data types are

closed under geometric operations. This means, for example, if a model defines a concept of fuzzy region objects, then the geometric intersection, union, and difference of two fuzzy region objects should yield a fuzzy region object again according to the corresponding type definition. Invalid result objects will let operations on them fail. Unfortunately, from all the models cited in this subsection, only the models in [18, 42] satisfy this requirement.

2.2 Approaches to Implementing Fuzzy Spatial Objects

To the author's knowledge, an implementation of fuzzy spatial data types, or, at least, an implementation of an adequate approximation of them, does not exist so far. In the crisp case, all points of a spatial object belong completely to that object so that a linear boundary representation can adequately serve as a finite representation. Line objects can be represented by a finite number of segments, and each segment can be finitely described by its two boundary endpoints. Region objects can be represented by a finite number of cycles. Each cycle can be represented by a finite number of segments, and each segment can be finitely described by its two boundary endpoints. In the fuzzy case, we are confronted with the problem that each point of a spatial object is associated with a membership value that is an element of the infinite range of real numbers between 0 and 1 in order to indicate partial membership. But only a finite number n of membership values can be explicitly represented in a computer system. This amounts to an n-valued approximation approach and to an n-valued logic.

For the case that $n = 3$ holds, several models (for example, [13, 14, 38]) have been proposed that lead to feasible and efficient implementations. Due to the strict restriction of n to 3, we cannot regard these approaches as fuzzy spatial object models but they help us better understand the concept and implementation idea of the Spatial Plateau Algebra. Without going into detail, despite some conceptual differences, they have all in common that they are 3-valued approximation approaches which, for an indeterminate spatial object, model parts that *definitely* belong to this object, parts that do *definitely not* belong to this object, and parts that *perhaps* belong to this object. In our own approach, called *Vague Spatial Algebra* (*VASA*) [38], these parts are called *kernel parts*, *exterior parts*, and *conjecture parts* respectively. All approaches rest on a 3-valued logic with the truth values *true*, *false*, and *maybe*. Their attractiveness and benefit rests on the fact that their conceptual and implementation framework is based on well known, general, and exact models of crisp spatial data types (see Section 2.3) and thus on a 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. In case of VASA, both the kernel part and the conjecture part of an indeterminate spatial object are of the same crisp spatial data type *point*, *line*, or *region*. This enables an exact definition of these models. In particular, this enables *executable specifications* for the operations on indeterminate spatial objects in terms of operations on the corresponding crisp counterparts. Hence, the executable specifications can be immediately used as implementations and thus minimize the needed implementation effort.

As we will see, the Spatial Plateau Algebra picks up and extends these ideas by representing fuzzy spatial objects on the basis of their crisp counterparts and by permitting n object parts with n different and arbitrary membership values such that, in general, $n \geq 1$ holds and, in particular, $n > 3$ is possible. Therefore, the Spatial Plateau Algebra is an extension of VASA from a 3-valued approach and logic to an n-valued approach and logic. Its benefit is a much more fine-grained modeling of spatial vagueness than in VASA, just in the sense of a fuzzy approach. But it requires much more, and much more precise, knowledge about the vagueness of spatial objects from the application side than we need in VASA. This knowledge has to be provided in terms of appropriate, application-specific membership functions.

An interesting alternative is the approach in [48] which provides two models for fuzzy spatial data. The first one is a raster (bitmap) approach, and the second one is a vector-based approach based on triangulated irregular networks (TINs). The approach in [2] uses a fuzzy inference system for spatial analysis and creates raster-based fuzzy region objects representing different transparency and stress intensities. The raster

(a) (b) (c) (d) (e) (f)

Figure 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).

approach (see also [49]) and the plateau approach proposed in this article show conceptual resemblance. But the goal of the Spatial Plateau Algebra is to leverage available crisp spatial (vector) data types and the operations defined on them as they can be found in spatial databases. The TIN model is quite interesting since it allows the user to model fuzzy spatial regions with continuous membership functions. However, it is only applicable to areal spatial features but not point and line features. Further, the cognitive view of fuzzy region objects gets lost since the user only sees a large collection of artificial triangles whose end points are annotated with membership values. The plateau approach is unable to implement the TIN model since all points of a component of a spatial plateau object have the same membership value.

Closure properties are also and in particular important at the implementation level. At this level, algorithms should only get input objects and only produce result objects that reflect the properties of the formal definitions of the data types at the modeling level and whose data structures correspond to the data structure definitions of the corresponding input and result data types at the implementation level. To be able to produce correct result values, algorithms need correct input values; otherwise, algorithms will fail and produce invalid results. Beside the Vague Spatial Algebra with its restricted logic, only the TIN-based model in [48] fulfills closure properties. Although the intersection, union, and difference of two TINs is usually not a TIN, a Delaunay triangulation can help us transform the intermediate intersection result into a TIN.

2.3 Crisp Spatial Data Types and Operations

In the spatial database and GIS communities, crisp *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 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 (for example, [24, 27]) and *complex* spatial data types (for example, [12, 41, 44, 53, 54]), depending on the spatial complexity they are able to model. Simple spatial data types (Figure 1(a)-(c)) only provide simple object structures like single points, continuous lines, and simple regions. However, from an application perspective, there is a common consensus that 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 (Figure 1(d)-(f)) solve these problems. They provide universal and versatile spatial objects with finitely many components, permit regions with holes, and are closed under geometric set operations [41]. We employ them as the definition basis of our spatial plateau data types. Implementations of complex spatial data types are available in ESRI's Spatial Database Engine (ArcSDE) [25], Oracle Spatial [37], the Informix Geodetic DataBlade [29], DB2's Spatial Extender [30], PostGIS [39], MySQL Spatial Support [36], Java Topology Suite (JTS) [31], and Geometry Engine Open Source (GEOS) [26]. They can be used as an implementation basis of our Spatial Plateau Algebra.

3 Spatial Plateau Objects for Representing Fuzzy Spatial Objects

Fuzzy spatial data types have been defined in [18, 42] on the basis of fuzzy set theory [56] and fuzzy topology [55]. Hence, they are an abstract concept only, and one of their main goals is to serve as a clear specification for a possible implementation. The objective of our *Spatial Plateau Algebra* is to provide such an implementation. Our algebra, or type system, comprises a set of *spatial plateau data types* and a set of operations between these types[1] to implement their abstract, fuzzy counterparts. A few operations are applicable to several types or type combinations and are thus overloaded. We do not aim at developing a type system with a "complete" set of data types and operations. 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 the Spatial Plateau Algebra is that it is both a *descriptive* algebra *and* an *executable* algebra. On the one hand, it offers a descriptive design of a type system with the specialty that spatial plateau 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 the Spatial Plateau Algebra on top of them with minimal effort, and directly *execute* spatial plateau 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 and formally define our concept of spatial plateau data types (and spatial plateau objects as their values) as the first fundamental part of the Spatial Plateau Algebra. Section 3.1 reviews the abstract definition of fuzzy spatial data types for fuzzy points, fuzzy lines and fuzzy regions from an informal viewpoint and illustrates these types by some examples. Section 3.2 informally introduces and motivates plateau points, plateau lines and plateau regions as implementation concepts of their fuzzy counterparts. Finally, Section 3.3 provides a formal definition of the three spatial plateau data types.

3.1 Review: Fuzzy Spatial Data Types

So far, spatial data handling in spatial database systems and GIS rests exclusively on the assumption that a spatial object like a point, line, or region object is precisely determined, homogeneous, and universally recognized, that each interior point fully belongs to that object, and that the object is delimited by a precisely specified boundary. However, many spatial objects, especially those describing natural, social, and cultural phenomena, do not follow this pattern. They are characterized by the inherent feature of *spatial vagueness* including vague interiors and blurred boundaries. Examples are land features with continuously changing properties (such as population density, soil quality, vegetation, pollution, temperature, air pressure), oceans, deserts, English speaking areas, or mountains and valleys. The transition between a valley and a mountain usually cannot be exactly ascertained so that the two spatial objects "valley" and "mountain" cannot be precisely separated and defined in a crisp way. For some areas of a natural resource like iron ore, experts know its existence with high certainty because of soil samples and boreholes and represent it by high membership values. For other areas, experts are only sure to some extent and assume the incidence of this mineral; this can be modeled by lower membership values. In case of oil spills, it is very important for environmental authorities 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 determination of the possible extent of the oil pollution.

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

(a) (b) (c)

Figure 2: Examples of air polluted areas caused by a chemical factory (a), (b), and polluted rivers (note that thick lines are used to visualize the feature of fuzziness) (c)

Figures 2a and 2b show an example and possible visualization of a *fuzzy region* that models the expansion of air pollution caused by a chemical factory. The exhaust fumes emitted by the factory spread around in the region surrounding the factory and create a pollution cloud. The shaded region shows the area which has been affected by the pollution particles. The density of pollution particles around the factory is not uniform but varies. Figure 2a shows a radial expansion where the degree of pollution concentrates in the center (darker locations) and decreases with increasing distance from the chemical factory (brighter locations). Figure 2b has the same theme but this time we imagine that the chemical factory is surrounded by high mountains to the north, the south, and the west. Hence, the pollution cannot escape in these directions and finds its way out of the valley in eastern direction. In both cases we can recognize the smooth, fuzzy transitions to the exterior, that is, there is no clear boundary of this region.

Fuzzy spatial objects can also represent the vagueness of the extent of phenomena in space; that is, objects can shrink and extend and hence have a minimum definite and maximum vague extent. Such an example is given by 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 minimum water level, that is, we obtain the parts of the lake with higher membership values. High precipitation entails rainy periods and thus a maximum water level, that is, we additionally obtain the parts of the lake with lower membership values. Islands in the lake form "holes" in the lake. The parts of an island that are never or rarely flooded by water have higher membership values. The parts that are flooded more often have lower membership values. Hence, a lake and an island can share common parts to some degree (multiple membership).

Fuzzy set theory [56] deals with fuzziness. It describes the *admission of the possibility* (given by a *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 like in probability theory. It is the vagueness resulting from the imprecision of the meaning of a concept. Fuzzy set theory has been a very popular approach to modeling vague spatial objects and resulted in a formal concept of *fuzzy spatial data types* for *fuzzy point objects*, *fuzzy line objects*, and *fuzzy region objects* [18, 42]. The definition of these spatial data types is based on two equivalent views, their flat view and their structured view [42]. The *flat view* regards fuzzy spatial objects as pure point sets in $\mathbb{R}^2$ which are characterized by particular properties and whose points are annotated with membership values. The *structured view* considers the *connected components* formed by these point sets.

We first review the flat view of fuzzy spatial objects at an abstract level and compare to crisp spatial objects. A crisp spatial object of the spatial data type *point*, *line*, or *region* is conceptually modeled as a point set of the Euclidean plane with particular features [41, 44]. Each of its points belongs definitely and completely to it. If $A \in \alpha \in \{point, line, region\}$, this means that there is a *characteristic function* $\chi_A : \mathbb{R}^2 \to \{0,1\}$ such that for all $p \in \mathbb{R}^2$ holds that $\chi_A(p) = 1$ if, and only if, $p \in A$. Otherwise, $\chi_A(p) = 0$ holds. Thus, χ discriminates sharply between points that belong to a spatial object and those that do not. In contrast, a fuzzy spatial object is conceptually modeled as a point set of the Euclidean plane with the particular feature that each of its points may completely, partially, or not at all belong to it. This especially

implies that a point can belong to multiple fuzzy spatial objects. If $\tilde{A} \in \alpha \in \{fpoint, fline, fregion\}$, this means that each point of $\mathbb{R}^2$ is mapped to a value of the real interval $[0,1]$ that represents the degree of its membership in $\tilde{A}$. We call such a single annotated point a *fuzzy point*[2]. Hence, for a fuzzy spatial object $\tilde{A}$, $\mu_{\tilde{A}} : \mathbb{R}^2 \to [0,1]$ is its membership function, and $\tilde{A} = \{(p, \mu_{\tilde{A}}(p)) \mid p \in \mathbb{R}^2\}$ describes all its fuzzy points, that is, all its points in $\mathbb{R}^2$ with their membership values. A *fuzzy point object* represents a *finite* set of fuzzy points, that is, points annotated with a membership value out of $]0,1]$ indicating partial membership. A *fuzzy line object* $\tilde{A}$ represents an infinite one-dimensional subset of $\mathbb{R}^2$ (with special properties discussed in [42]) whose elements are annotated with a membership value out of $]0,1]$ indicating partial membership. A *fuzzy region object* is an infinite two-dimensional subset of $\mathbb{R}^2$ (with special properties discussed in [42]) whose elements are annotated with a membership value out of $]0,1]$ indicating partial membership. The distribution of membership values within a fuzzy spatial object may be smooth, continuous, or piecewise continuous.

While the flat view regards fuzzy spatial objects as pure point sets with particular properties, the equivalent structured view looks at the connected components of these point sets and identifies their geometric and topological features. All fuzzy spatial objects are complex, that is, they may consist of several connected components, and fuzzy regions may have holes. This is in line with the definition of crisp complex spatial data types [41, 44]. The flat view and the structured view coincide for a *fuzzy point object* since each fuzzy point is a connected component. The structured view of a *fuzzy line object* distinguishes a finite number of connected components called *fuzzy blocks*. Each fuzzy block consists of a finite number of meeting or disjoint *fuzzy curves*. Each fuzzy curve represents a single continuous curve in which each element is a fuzzy point and which may have smooth transitions of membership grades between neighboring points. The structured view of a *fuzzy region object* distinguishes a finite number of areal and connected components possibly with holes. A component is a called a *fuzzy face*. Both the boundary and the interior of a fuzzy region object may be fuzzy.

If we consider an air polluted area modeled as a fuzzy region object (Figures 2a and 2b), then each point in this area is a fuzzy point indicating the concentration of air pollution at that point. That is, the degree of membership of such a fuzzy point in the air polluted area is larger than 0 and less than or equal to 1. In Figures 2a and 2b, each fuzzy region object consists of a single fuzzy face without holes. Fuzzy points can also arise from the geometric intersection of two fuzzy line objects; together, they form a fuzzy point object. The pollution of a river in a low scale map can be represented by a fuzzy line object, as shown in Figure 2c, where each point on the fuzzy line object represents the concentration of the pollutants at that location. The concentration may be different at different points. In Figure 2c, the darker line parts represent river sections of greater pollution, and the lighter line parts represent river sections of lower pollution. The fuzzy line object here consists of a single fuzzy block with three fuzzy curves meeting in a common point.

3.2 Spatial Plateau Objects

To the author's best knowledge, implementations of fuzzy spatial objects are not available, especially not in a spatial database and GIS context. Hence, this article is a first approach in this direction. Since conceptually, that is, at an abstract level, crisp and fuzzy spatial objects are modeled as infinite point sets that are not directly representable in finite computer systems, finite representations are needed and usually obtained by linear approximations. For example, a crisp, curvilinear line is usually approximated by a polyline consisting of a finite sequence of straight segments. A crisp, curvilinear region is usually approximated by well known polygonal structures for the outer cycles and holes cycles of its components with the assumption that the enclosed interior belongs completely to the region. The employment of approximations essentially means that we are only able to represent a proper subset of the actual set of conceptually possible lines or regions.

[2]The literature knows different definitions of a *fuzzy point*. In some cases, they correspond to our understanding of *simple fuzzy point objects* (see below). In our context here, the term of a fuzzy point is used as a counterpart to a crisp point as an element of the Euclidean plane.

However, even such approximations are not easy to obtain for fuzzy spatial objects since first, these objects usually have an indeterminate boundary and/or a blurred interior, second, they have infinitely many interior points but only finite representations can be kept in a computer, and third, each point can have a different membership value.

In this article, the fundamental idea for representing, implementing, and approximating fuzzy spatial objects is to leverage available crisp spatial data types and corresponding software packages (Section 2.3) implementing them. Several reasons have led to this design decision. First, this strategy enables us to take advantage of existing definitions, techniques, data structures, algorithms, etc., which need not be redeveloped but only modified and extended, or simply used. For the spatial plateau data types this means that they are based on their crisp counterparts. For the spatial plateau operations (like geometric union, intersection, and difference) and spatial plateau predicates (like topological predicates) this means that they are translated to crisp spatial operations and predicates (see Section 4). By using this approach, the formal specification of spatial plateau data types, operations, and predicates is at the same time their implementation; we call this *executable specification*. Second, this strategy improves the correctness of both the conceptual specification and the implementation of the Spatial Plateau Algebra. At the conceptual level, the correctness of the definitions of the spatial plateau concepts largely rests on the correctness of the already defined crisp spatial concepts; thus, we reduce the chance of errors in our definitions. At the implementation level, having an available, tested and robust implementation of crisp spatial data types, their operations, and predicates, we can robustly implement fuzzy spatial data types in terms of spatial plateau data types on top of them (see Section 6). Third, several available implementations of crisp spatial data types are tailor-made for an embedding and usage in a database and querying context. This means that the data structures of crisp point, line, and region objects are *compact storage structures* (arrays) that can be directly stored in or retrieved from a database and that are directly processed by the algorithms implementing the operations. In particular, these data structures do not use pointer data structures in main memory which would imply high serialization and deserialization cost for a transfer between main memory and database. We apply this concept of compact storage representation to our spatial plateau objects.

Our *Spatial Plateau Algebra* is a *type system*, or *algebra*, which includes spatial data types, operations, and predicates for *plateau points*, *plateau lines*, and *plateau regions* that implement their fuzzy counterparts. The approximation step from fuzzy spatial objects to spatial plateau objects contains two aspects. First, since we are unable to explicitly represent an infinite number of membership values, we confine ourselves to an arbitrary but representable and thus finite number of membership values for each spatial plateau object. However, different spatial plateau objects may have different finite numbers of membership values. Second, in a spatial plateau object, the unit of representation is not a point as is the case in an abstract fuzzy spatial object (Section 3.1) but a (possibly very small) corresponding crisp spatial object which itself is already a (linear) approximation and which is additionally labeled with a membership value. A spatial plateau object is then a finite collection of their crisp counterparts where each crisp spatial object (called a *component object*) is associated with a unique membership value and thus forms a flat "plateau" consisting of a conceptually infinite number of points of equal membership. The precision with which we represent a fuzzy spatial object as a spatial plateau object depends on the size and the number of the crisp component objects of the spatial plateau object. The smaller the size and the larger the number of the crisp component objects is, the better is the approximation and the higher is the precision of the representation of a fuzzy spatial object.

Figure 3 illustrates the concept. The conceptual, abstract fuzzy region object shown in Figure 3a is represented as a plateau region object shown in Figure 3b with the nine crisp *component regions* $r_1, \ldots, r_9$ and their associated membership values. Any pair of component regions is either *disjoint* (for example, r_1 and r_9) or *adjacent* (for example, r_2 and r_5). A component region is, in general, a complex region (see Section 2.3) and can hence consist of several faces (for example, r_3 has two faces) that all have the same membership value. Different component regions have different membership values assigned to them. Similarly, Figure 3c shows a fuzzy line object, and Figure 3d its possible plateau representation. We will see

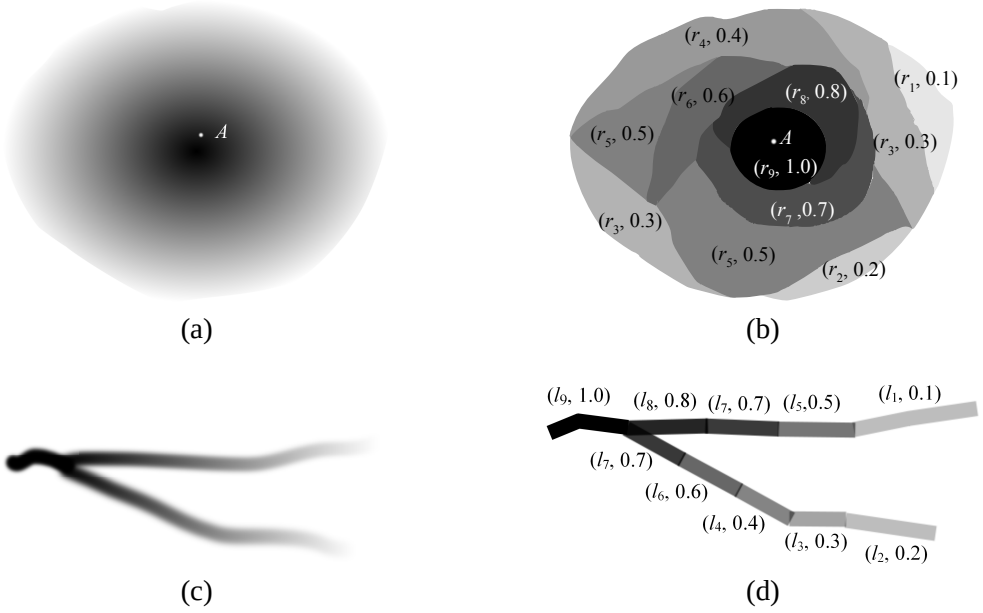
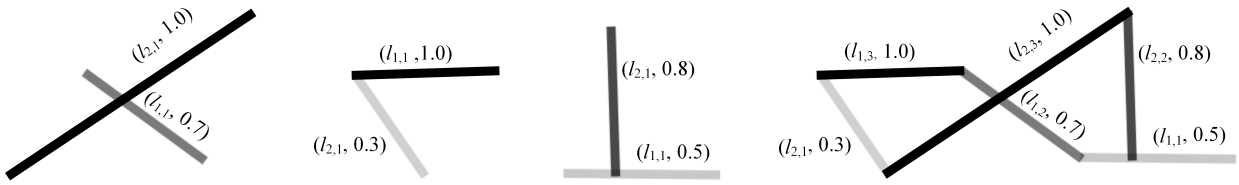


Figure 3: A fuzzy region object (a), its (possible) representation as a plateau region object (b), a fuzzy line object (c), and its (possible) representation as a plateau line object (d)

below that the topological relationships allowed between the component lines of a plateau line object are slightly different in comparison to the plateau region case. A fuzzy point object and its plateau representation are structurally the same; both consist of a finite number of point objects where the elements of each point object are associated with the same membership value and where different point objects carry different membership values. The only difference is the underlying number system used for the representation of the coordinates (for example, infinitely many *real* numbers $\mathbb{R}$ from mathematics for modeling fuzzy spatial objects versus finitely many *floating point* numbers (as approximations of real numbers) in computer systems for implementing spatial plateau objects).

Another, equivalent view of a spatial plateau object is that all its points that have the same membership value are aggregated into groups and that each group of points is associated as a whole with the same membership value. In addition, to obtain a finite representation, we require that the number of groups is finite (finitely many membership values) and that each group forms a component object (atomic representation unit) of the corresponding crisp spatial data type. Hence, spatial plateau objects can only represent a clearly defined, proper subset of fuzzy spatial objects (see Section 3.3).

As a consequence of this aggregation process, any two crisp component regions of a plateau region object must be either disjoint or adjacent. Otherwise, two crisp component regions would share interior points with different membership values. While this can be avoided for interior points, this is not the case for the boundaries of two or more adjacent crisp component regions since they have common points with different membership values. In Figure 3b, for example, the points on the boundary between r_3 and r_5 have the membership values 0.3 and 0.5, respectively, and the single boundary point shared by r_4, r_6, and r_8 has the membership values 0.4, 0.6, and 0.8 respectively. In case of a plateau line object, the situation is slightly different. The component lines of a plateau line object can, of course, be disjoint (for example, l_2 and l_8 in Figure 3d) or adjacent (for example, l_4 and l_6). Adjacency means that two or more component lines meet in a common boundary end point. Again, for each meeting component line, the membership value might be different in such an end point. In addition and different from the plateau region case, we must also allow that two component lines intersect in a common interior point and that a boundary point of one component line coincides with an interior point of another component line. In summary, we must permit arbitrary point



(a) (b) (c) (d)

Figure 4: Two plateau line objects with intersecting (a), meeting (b), touching (c), intersecting, meeting, and touching (d) crisp component lines

intersections of component lines. The reason is that the geometric union of two plateau line objects can, in general, lead to arbitrary boundary/boundary, boundary/interior and interior/interior point intersections that have to be maintained in the resulting single plateau line object in order to ensure the closure property of the fuzzy/plateau union operation (see Section 4). Figure 4 shows examples of all three kinds of point intersections.

Two questions arise from these observations. The first question refers to the semantics of common boundary parts of adjacent component regions in plateau region objects and common single points of meeting, intersecting, or touching component lines in plateau line objects. A point that is shared by more than one component object obtains the highest membership value of all sharing component objects since it is guaranteed to belong to the spatial plateau object (and thus the fuzzy spatial object) by this maximum membership value. The second question is how to handle these observations from a representation standpoint. We pursue the strategy *not* to explicitly model and represent the possible multiple memberships of a point in the *same* spatial plateau object but to maintain consistent component objects in which all points have the same membership value. This implies that for the determination of the membership value of a particular point of a spatial plateau object we have to consult all its component objects that contain this point and yield the maximum membership value.

The limitations of the Spatial Plateau Algebra can be summarized as follows. First, the number of membership values for each spatial plateau object is variable but finite. The algebra cannot represent infinitely many membership values, for example, by a finite number of membership intervals. Second, the term "plateau" indicates that a membership value remains constant in a line component or region component of a plateau line object or plateau region object. This means that fuzzy spatial objects with continuously increasing or decreasing membership functions cannot be modeled by this approach. Third, at the boundaries of line components and region components, the membership functions of plateau line objects and plateau region objects are not differentiable since we have abrupt membership value changes at these locations.

3.3 Formal Definition of Spatial Plateau Data Types

Based on the considerations in Section 3.2 we are now able to give a formal definition of the fuzzy spatial data types *fpoint* for fuzzy point objects represented as plateau point objects, *fline* for fuzzy line objects represented as plateau line objects, and *fregion* for fuzzy region objects represented as plateau region objects. We aim at a universal and generic definition of these types since the structure of their objects are very similar and since the only difference consists in the possible relationships between component objects. For this purpose, we define a type constructor ϕ that takes as input a spatial data type $\alpha \in \{point, line, region\}$ (Section 2.3) and produces as output a fuzzy spatial data type as a spatial plateau data type $\phi(\alpha)$. That is, $fpoint := \phi(point)$, $fline := \phi(line)$, and $fregion := \phi(region)$. We define a spatial plateau data type $\phi(\alpha)$ as follows (the symbol $\bigoplus$ in Condition (vi) denotes geometric union):

$\phi(\alpha) = \{po_1, \ldots, po_{k_\alpha} \mid$
(i) $\alpha \in \{point, line, region\}, k_\alpha \in \mathbb{N}$
(ii) $\forall 1 \leq i \leq k_\alpha : po_i = \langle (o_{i,1}, m_{i,1}), \ldots, (o_{i,n_i}, m_{i,n_i}), o_i \rangle$
(iii) $\forall 1 \leq i \leq k_\alpha : n_i \in \mathbb{N} \cup \{0\}$
(iv) $\forall 1 \leq i \leq k_\alpha \; \forall 1 \leq j \leq n_i : o_{i,j} \in \alpha$
(v) $\forall 1 \leq i \leq k_\alpha \; \forall 1 \leq j \leq n_i : m_{i,j} \in]0,1]$
(vi) $\forall 1 \leq i \leq k_\alpha : o_i = \bigoplus_{j=1}^{n_i} o_{i,j} \in \alpha$
(vii) $\forall 1 \leq i \leq k_\alpha \; \forall 1 \leq j < l \leq n_i :$
if $\alpha = point$ then $disjoint_c(o_{i,j}, o_{i,l})$
else if $\alpha = line$ then $disjoint_c(o_{i,j}, o_{i,l}) \vee meet_c(o_{i,j}, o_{i,l}) \vee$
$(overlap_c(o_{i,j}, o_{i,l}) \wedge |o_{i,j} \cap o_{i,l}|$ is finite)
else $disjoint_c(o_{i,j}, o_{i,l}) \vee meet_c(o_{i,j}, o_{i,l})$
(viii) $\forall 1 \leq i \leq k_\alpha \; \forall 1 \leq j < l \leq n_i : m_{i,j} < m_{i,l}$
(ix) if $\alpha \in \{point, region\}$ then $\forall 1 \leq i \leq k_\alpha \; \forall 1 \leq j \leq n_i \; \forall p \in o_{i,j}^\circ : \mu(p) = m_{i,j}$
(x) if $\alpha = region$ then $\forall 1 \leq i \leq k_\alpha \; \forall p \in \bigcup_{j=1}^{n_i} \partial o_{i,j} : \mu(p) = \max\{m_{i,j} \mid 1 \leq j \leq n_i, p \in \partial o_{i,j}\}$
(xi) if $\alpha = line$ then $\forall 1 \leq i \leq k_\alpha \; \forall p \in \bigcup_{j=1}^{n_i} o_{i,j} : \mu(p) = \max\{m_{i,j} \mid 1 \leq j \leq n_i, p \in o_{i,j}\}\}$

The representable spatial plateau objects of each of the three spatial plateau data types can be enumerated since their number is finite due to the finiteness of computer systems (Condition (i)). Each spatial plateau object is represented as a finite sequence of pairs and a single entity at the end, that is, as a compact storage representation (Condition (ii)). The number of pairs depends on each individual spatial plateau object and can thus be different for different spatial plateau objects (Condition (iii)). If the number of pairs is equal to zero, we obtain the *empty spatial plateau object*, that is, the *empty plateau point object*, the *empty plateau line object*, and the *empty plateau region object* respectively. Each pair consists of an object of the underlying complex spatial data type (Condition (iv)) and a non-zero membership value (Condition (v)) indicating the degree of belonging of the crisp spatial object (called *component object*) to the spatial plateau object. For example, a plateau region object consists of a finite number of pairs where each pair consists of a crisp component region and an assigned membership value larger than 0 and less than or equal to 1. At the end of each spatial plateau object, we store the geometric union of all its component objects (Condition (vi)). The reason for this is more based on convenience and performance issues than on necessity. We make use of the union objects later for a more efficient execution of spatial plateau operations.

Condition (vii) determines the permitted topological relationships between the component objects of a spatial plateau object. These relationships depend on the underlying complex spatial data type and are expressed by *topological cluster predicates* [44] indicated by the subscript c. A topological cluster predicate summarizes several *basic* topological predicates that characterize *similar* spatial configurations (for example, all basic topological predicates that describe a meeting situation of two spatial objects). In case of plateau point objects, we require that the component point objects are disjoint. In case of plateau line objects and plateau region objects, we require that the corresponding component objects are disjoint from each other or meet. Any overlap of the interiors of component objects would be semantically contradictory. In case of plateau line objects, we additionally allow that component line objects overlap in finitely many points (point intersections). We discussed the need for allowing this in detail in Section 3.2. Condition (viii) requires that all membership values are different and that all pairs of the sequence are ordered by increasing membership values. This caters for a unique representation of a plateau region.

Conditions (ix) to (xi) take care of a precise assignment of membership values to the points of a spatial plateau object. The $^\circ$ operator and the ∂ operator used in the conditions are point-set topological operators that determine all interior points and boundary points, respectively, of a point set. Precise definitions of the interior and boundary of complex point, line, and region objects can be found in [44]. The interior of a point

object is the point object itself; its boundary is empty. The interior of a line object contains all its points except for the end points which form the boundary. The boundary of a region object separates its interior points from its exterior points. Condition (ix) states that all interior points of a component point object or component region object, respectively, obtain the membership value of the respective component object. Thus, they define a plateau. Condition (x) expresses that each boundary point of a plateau region object obtains the highest membership value of all boundaries of component region objects to which it belongs. Condition (xi) states that each point of a plateau line object obtains the highest membership value of all component line objects to which it belongs. As we have seen in Section 3.2, multiple membership values for the same point can arise if different component line objects intersect, meet, or touch each other in that point (see Figure 4). Note that the Conditions (ix) to (xi) are semantical conditions that are not explicitly stored in the structural sequence representation given in Condition (ii). An operation that asks for the membership value of a point of a spatial plateau object would have to check Conditions (ix) to (xi). If Condition (x) or (xi) should hold, the whole sequence of components objects would have to be traversed, and the highest membership value assigned to that point would have to be returned.

Another, equivalent characterization of spatial plateau objects is based on spatial α-cut objects. Let $po \in \phi(\alpha)$ with $\alpha \in \{point, line, region\}$ be a spatial plateau object, and let $m_i \in]0,1]$ with $1 \leq i \leq n$ be n membership values such that $m_1 > m_2 > \ldots > m_{n-1} > m_n$. Then we obtain the spatial α-cut objects $po_{m_1} \subseteq po_{m_2} \subseteq \ldots \subseteq po_{m_{n-1}} \subseteq po_{m_n}$ from po with $po_{m_i} \in \alpha$. A representation of po by means of these spatial α-cut objects is not recommendable for three main reasons. First, due to the containment relationship between the n spatial α-cut objects, a point can belong to several of them. This means that a point of po can be represented up to n times. This leads to representation and storage redundancy since certain areas (for example, areas whose points have the membership value 1) are represented multiple times. Second, it is time-consuming to determine the membership of a point since the spatial α-cut object with the highest membership value has to be found in which the point is located. Third, one can show that spatial plateau operations become more time-consuming due to the larger size of spatial α-cut objects.

However, it is possible to characterize a spatial plateau object po by spatial α-cut objects. A spatial plateau object corresponds to the sequence of pairs

$$\langle (o_i, m_i) \,|\, o_i = po_{m_i} \ominus po_{m_{i-1}} \text{ for } 2 \leq i \leq n \rangle$$

where $\ominus$ denotes the geometric difference operation. For all $p \in o_i$ holds that $\mu_{po}(p) = m_i$. This means that each point of a spatial plateau object po belongs to exactly one of the *component objects* o_i and has a unique membership value m_i.

4 Spatial Plateau Operations for Implementing Fuzzy Geometric Operations

Spatial or geometric operations enable the construction of new spatial objects from existing ones. In this section, we give formal specifications of some *spatial plateau operations* that are geometric operations on spatial plateau objects, implement corresponding fuzzy geometric operations, and are based on geometric operations on corresponding crisp spatial objects. The latter aspect enables the executable specification of the spatial plateau operations. That is, the specification of these operations corresponds to their implementation. Section 4.1 deals with the important class of *spatial plateau set operations* that include the operations *plateau union*, *plateau intersection*, and *plateau difference* and implement their fuzzy counterparts. Section 4.2 introduces some other spatial plateau operations like different *slice* operations that construct new spatial plateau objects based on constraints on the membership values as well as other intersection operations that compute the intersection between mixed spatial plateau data types. Section 4.3 shows that the determination of the membership values in the Sections 4.1 and 4.2 is not the only option for the result of

each spatial plateau operation. That is, not only the membership functions of fuzzy spatial objects (and thus spatial plateau objects) but also the operations on them are *context-dependent*. This leads to a class of parameterized operation instances for each spatial plateau operation. Section 4.4 provides operations to map spatial plateau objects of the Spatial Plateau Algebra to spatial objects of our Spatial Algebra 2D for two-dimensional geometric data, and vice versa.

4.1 Spatial Plateau Set Operations for Implementing Fuzzy Geometric Set Operations

In general, the geometric set operations *intersection*, *union*, and *difference* applied to spatial objects belong to the most important operations that construct new spatial objects, that is, geometries. In this section, we describe and give formal definitions of the geometric set operations on fuzzy spatial objects on the basis of our spatial plateau data types. Section 4.1.1 informally reviews the abstract definition of fuzzy geometric set operations that are formally based on fuzzy set theory and fuzzy topology. Section 4.1.2 discusses the implementation of these fuzzy geometric operations in terms of the Spatial Plateau Algebra. Finally, Section 4.1.3 provides a formal definition of the spatial plateau set operations.

4.1.1 Review: Fuzzy Geometric Set Operations

Unsurprisingly, fuzzy geometric set operations are spatial operations that operate on fuzzy spatial objects and produce new fuzzy spatial objects. As an example, we consider the situation that in some European countries, as in Switzerland and France, people who speak German and people who speak French reside very close to each other. This means that language borders are not as strict as state or country borders; they are fluent. Such a situation can be described by two fuzzy regions representing the two language zones and the degree to which German and French, respectively, are spoken in the two zones. Both regions have indeterminate boundaries and blurred interiors since at many locations neither French nor German are spoken solely but together. Let us now assume that a government is interested in a study of the population and wants to find those regions whose residents speak both languages. Such a query can be answered by the *fuzzy geometric intersection* of both fuzzy regions. A point shared by both language zones will belong to the result region and obtain the lower membership value of this point from one of the two zones since it represents the minimum extent to which German *and* French are spoken there. If we are interested in those regions whose residents speak either French or German, the *fuzzy geometric union* can answer this query. A point shared by any of the two language zones will contribute to the result and obtain the higher membership value of this point from one of the two zones since it represents the maximum extent to which French *or* German are spoken there. Further, if we are interested in the region whose residents speak more French than German and if we want to quantify the French ascendancy, we obtain the result by computing the *fuzzy geometric difference* of the French language zone and the German language zone. A point of the result object contains the membership of the French language zone diminished by the membership value of the same point in the German language zone.

We now describe the fuzzy geometric set operations in more detail but informally. As we have seen in Section 3.1, at an abstract level, fuzzy set theory and fuzzy (point set) topology are deployed to formally define the fuzzy spatial data types *fpoint*, *fline*, and *fregion*. The fuzzy geometric set operations *fintersection*, *funion*, and *fdifference* have the same formal basis and the signature $\alpha \times \alpha \to \alpha$ with $\alpha \in \{fpoint, fline, fregion\}$. That is, these data types are closed under the fuzzy geometric set operations.

The operation *funion* for *fuzzy geometric union* assigns the membership value $\mu_{\tilde{C}}(p) = \max(\mu_{\tilde{A}}(p), \mu_{\tilde{B}}(p))$ to each point $p \in \mathbb{R}^2$. If for a point p holds that $\mu_{\tilde{A}}(p) = \mu_{\tilde{B}}(p) = 0$, then $\mu_{\tilde{C}}(p) = 0$ follows, and p does not belong to the result object $\tilde{C}$. In all other cases, point p belongs to $\tilde{C}$ with the larger membership value provided by one of the two fuzzy spatial objects. The operation *fintersection* for *fuzzy geometric intersection* assigns the membership value $\mu_{\tilde{C}}(p) = \min(\mu_{\tilde{A}}(p), \mu_{\tilde{B}}(p))$ to each point $p \in \mathbb{R}^2$. Only if

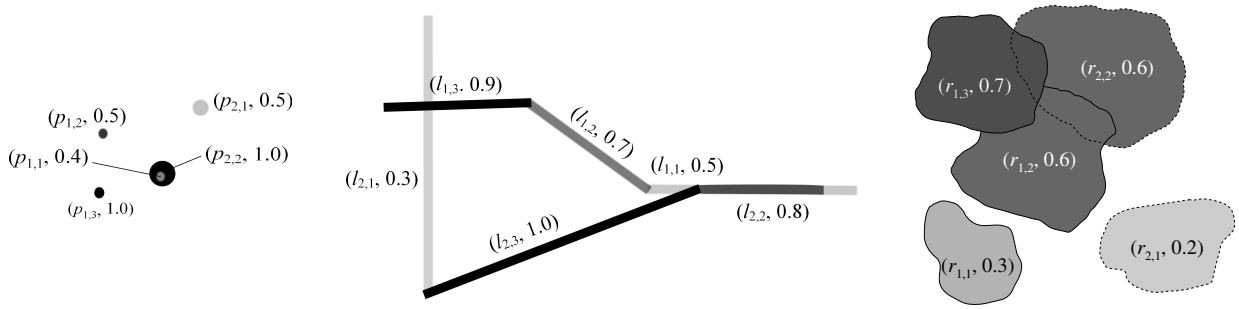


Figure 5: Two plateau point objects $pp_1 = \langle (p_{1,1}, 0.4), (p_{1,2}, 0.5), (p_{1,3}, 1.0), p_1 \rangle$ and $pp_2 = \langle (p_{2,1}, 0.5), (p_{2,2}, 1.0), p_2 \rangle$ (a), two plateau line objects $pl_1 = \langle (l_{1,1}, 0.5), (l_{1,2}, 0.7), (l_{1,3}, 0.9), l_1 \rangle$ and $pl_2 = \langle (l_{2,1}, 0.3), (l_{2,2}, 0.8), (l_{2,3}, 1.0), l_2 \rangle$ (b), and two plateau region objects $pr_1 = \langle (r_{1,1}, 0.3), (r_{1,2}, 0.6), (r_{1,3}, 0.7), r_1 \rangle$ and $pr_2 = \langle (r_{2,1}, 0.2), (r_{2,2}, 0.6), r_2 \rangle$ (c), as they are used for illustrating the three spatial plateau set operations

$\mu_{\tilde{A}}(p) > 0$ and $\mu_{\tilde{B}}(p) > 0$ hold, p belongs to $\tilde{C}$ with the lower membership value provided by the two fuzzy spatial objects. The operation *fdifference* for *fuzzy geometric difference* assigns the membership value $\mu_{\tilde{C}}(p) = \mu_{\tilde{A}}(p) \dot{-} \mu_{\tilde{B}}(p)$ to each point $p \in \mathbb{R}^2$. For $a, b \in \mathbb{R}$, we define that $a \dot{-} b = a - b$ if $a > b$, and $a \dot{-} b = 0$ otherwise. Note that this geometric operation is defined differently than the fuzzy set difference operation on normal fuzzy sets [56]. Further, the operations *funion* and *fintersection* are subject to a fuzzy regularization process in order to avoid geometric anomalies like dangling lines, punctures, and cuts. The formal definition of all fuzzy geometric set operations can be obtained from [18, 42]. Crisp geometric set operations [41, 44] are special cases of fuzzy geometric set operations.

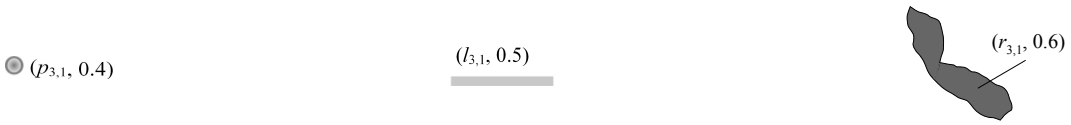
4.1.2 Spatial Plateau Set Operations

In this subsection, we informally show and motivate how the fuzzy geometric set operations *fintersection*, *funion*, and *fdifference* can be represented by corresponding spatial plateau set operations on the basis of our spatial plateau data types. The spatial plateau data types *fpoint*, *fline*, and *fregion* are closed under the spatial plateau set operations and have the signature $\alpha \times \alpha \rightarrow \alpha$ with $\alpha \in \{fpoint, fline, fregion\}$ (see Section 4.1.3). We will see that for the different combinations of two plateau point objects, or two plateau line objects, or two plateau region objects, each spatial plateau set operation works very similar and thus in a generic manner. This means that the meaning (and later the specification) of all three operations is similar for all three type combinations. For illustration purposes, we will take the spatial plateau objects presented in Figure 5.

The discussion and specification of the spatial plateau set operations requires and leverages a local view and/or a global view on the two operand spatial plateau objects. Having a *local view* on the two operand objects means that putting all component objects of the first operand object into relationship to all component objects of the second operand object is sufficient for the evaluation of an operation. Having a *global view* on the two operand objects means that an operation has to take into account the union objects stored in the spatial plateau objects at the end of their representations.

Spatial Plateau Intersection

Intersecting two spatial plateau objects $po_1, po_2 \in \alpha \in \{fpoint, fline, fregion\}$ means that each crisp component object of po_1 must be geometrically intersected with each crisp component object of po_2 and that their smaller membership value is assigned to the resulting non-empty crisp component object. Hence, this operation only requires a local view on the operand spatial plateau objects. The reason is that if a point of



(a) (b) (c)

Figure 6: Spatial plateau intersection: plateau point object $pp_3 = \mathit{fintersection}(pp_1, pp_2) = \langle (p_{3,1}, 0.4), p_3 \rangle$ (a), plateau line object $pl_3 = \mathit{fintersection}(pl_1, pl_2) = \langle (l_{3,1}, 0.5), l_3 \rangle$ (b), and plateau region object $pr_3 = \mathit{fintersection}(pr_1, pr_2) = \langle (r_{3,1}, 0.6), r_3 \rangle$ (c)

the Euclidean plane belongs to both po_1 and po_2, then it belongs to exactly one component object of po_1 and exactly one component object of po_2. Computing the intersection of crisp complex spatial objects is well known [3, 40, 41] and can therefore be assumed. In our example in Figure 5a, we obtain $3 \cdot 2 = 6$ point component pairs that have to be intersected since pp_1 contains three point components and pp_2 contains two components. Similarly, we obtain $3 \cdot 3 = 9$ line component pairs in Figure 5b and $3 \cdot 2 = 6$ region component pairs in Figure 5c. If the geometric intersection of two component objects is empty, we discard this result. Otherwise, the computed component object is part of the resulting spatial plateau object, and we assign the smaller membership value of both operand component objects to it. The reason is that only the smaller membership value guarantees that the points of the intersection belong to both component objects. For example, in Figure 5c, the intersection of the component regions $r_{1,3}$ with its membership value 0.7 and $r_{2,2}$ with its membership value 0.6 leads to a non-empty component region that is assigned the membership value 0.6 as the minimum of both input membership values.

Since the creation of component object pairs and their intersection is a local operation, it can happen that different resulting component objects are labeled with the same membership value. For example, in Figure 5c, the intersection of $r_{1,3}$ and $r_{2,2}$ will obtain the membership value 0.6. Similarly, the intersection of $r_{1,2}$ and $r_{2,2}$ will obtain the same membership value. Since according to the plateau region definition in Section 3.3 all components of a plateau region must have different membership values, we have to compute the geometric union of both region components obtained so far and assign the common membership value 0.6 to it. Figure 6 shows the result of the plateau intersection for our examples in Figure 5.

Spatial Plateau Union

This operation requires both the local view and the global view on its operand spatial plateau objects. The local view refers to the common parts of two component objects. The global view refers to the remaining part of a component object that is not shared with the other spatial plateau object. In all cases, computed component objects with the same membership value are always geometrically merged in the resulting spatial plateau object. We detail this strategy in the following. Forming the union of two objects po_1 and po_2 of the same spatial plateau data type principally means that each component object of po_1 must be geometrically merged with each component object of po_2, that common parts obtain the larger membership value of the two input component objects, and that the remaining parts get the membership values of their corresponding input component objects. The intersection object of two component objects gets their larger membership value since at least one of them can guarantee the higher degree of belonging.

The union of the spatial plateau objects in Figure 5 is illustrated in Figure 7. Two main kinds of spatial configurations can be distinguished. If the intersection of two component objects is empty, both component objects are added with their respective membership values to the resulting spatial plateau object. For example, this is the case for the component lines $l_{1,3}$ and $l_{2,2}$ in Figure 5b and the component regions $r_{1,1}$ and $r_{2,1}$ in Figure 5c. In their union, they can be found again as the component lines $l_{4,5}$ and $l_{4,4}$ (see Figure 7b) as well as the component regions $r_{4,2}$ and $r_{4,1}$ (see Figure 7c) with their original membership values.

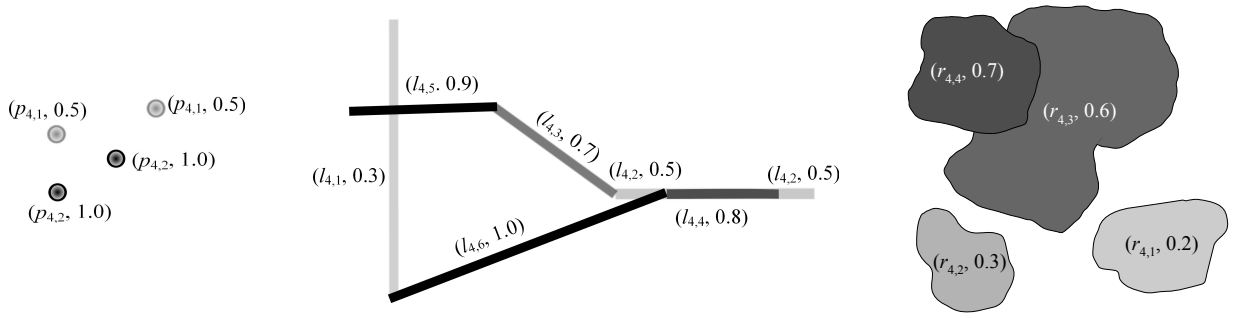


Figure 7: Spatial plateau union: plateau point object $pp_4 = \mathit{funion}(pp_1, pp_2) = \langle (p_{4,1}, 0.5), (p_{4,2}, 1.0), p_4 \rangle$ (a), plateau line object $pl_4 = \mathit{funion}(pl_1, pl_2) = \langle (l_{4,1}, 0.3), (l_{4,2}, 0.5), (l_{4,3}, 0.7), (l_{4,4}, 0.8), (l_{4,5}, 0.9), (l_{4,6}, 1.0), l_4 \rangle$ (b), and plateau region object $pr_4 = \mathit{funion}(pr_1, pr_2) = \langle (r_{4,1}, 0.2), (r_{4,2}, 0.3), (r_{4,3}, 0.6), (r_{4,4}, 0.7), r_4 \rangle$ (c)

Otherwise, the two component objects intersect, and up to three new component objects are stored in the resulting spatial plateau object. The local view relates to the common parts of both component objects. Here we have to distinguish three cases. First, if the two component objects are geometrically equal, a copy with the larger membership value of both equal component objects is added to the resulting spatial plateau object. Second, if one component object is contained in the other one, the contained component object annotated with the larger membership value of both component objects is added to the resulting spatial plateau object. For example, in Figure 5b, the component line $l_{2,2}$ with the membership value 0.8 is contained in the component line $l_{1,1}$ with the membership value 0.5 and can be found again as the component line $l_{4,4}$ with the higher membership value 0.8 in the spatial plateau union in Figure 7b. Third, if both component objects have a proper intersection, the common part is added to the resulting spatial plateau object with the larger membership value of both component objects. For example, the intersection of the component regions $r_{1,3}$ with the membership value 0.7 and $r_{2,2}$ with the membership value 0.6 in Figure 5c is contained as part of the component region $r_{4,4}$ with the higher membership value 0.7 (see Figure 7b).

The handling of the remaining unshared parts of each component object requires a global view. We illustrate this with an example. In Figure 5c, we consider the component object pairs $r_{1,2}$ and $r_{2,2}$ as well as $r_{1,3}$ and $r_{2,2}$. Let $\otimes$, $\oplus$, and $\ominus$ denote the generic geometric intersection, union, and difference operations on crisp spatial data types [41]. The remaining unshared component objects are $r_{1,2} \ominus r_{2,2}$, $r_{2,2} \ominus r_{1,2}$, $r_{1,3} \ominus r_{2,2}$, and $r_{2,2} \ominus r_{1,3}$. If we took the local view, we would, for example, store the tuples $(r_{2,2} \ominus r_{1,2}, 0.6)$ and $(r_{2,2} \ominus r_{1,3}, 0.6)$ into the resulting spatial plateau object. Since component objects with equal membership values are merged, we would obtain $((r_{2,2} \ominus r_{1,2}) \oplus (r_{2,2} \ominus r_{1,3}), 0.6) = (r_{2,2} \ominus (r_{1,2} \otimes r_{1,3}), 0.6) = (r_{2,2}, 0.6)$ since $r_{1,2} \otimes r_{1,3} = \varnothing$. Obviously, this is not the expected result. Instead, the expected result is $(r_{2,2} \ominus (r_{1,2} \oplus r_{1,3}), 0.6)$. That is, from $r_{2,2}$ we have to subtract all component regions of the first operand object pr_1 that intersect $r_{2,2}$. This cannot be locally performed. Since we cannot find out easily which component objects of pr_1 intersect $r_{2,2}$, we leverage the global view provided by the union objects, and compute $(r_{2,2} \ominus r_1, 0.6)$ or, as an optimization, $(r_{2,2} \ominus (r_1 \otimes r_2), 0.6)$ instead of $(r_{2,2} \ominus (r_{1,2} \oplus r_{1,3}), 0.6)$. The original membership value is always maintained in the result. We apply this principle to all component objects of both operand spatial plateau objects for all combinations of equal spatial plateau data types.

Spatial Plateau Difference

This operation also requires both the local view and the global view on its operand spatial plateau objects. The local view refers to the common parts of two component objects. The global view refers to the remaining part of a component object of the first spatial plateau object that is not shared with the second spatial plateau

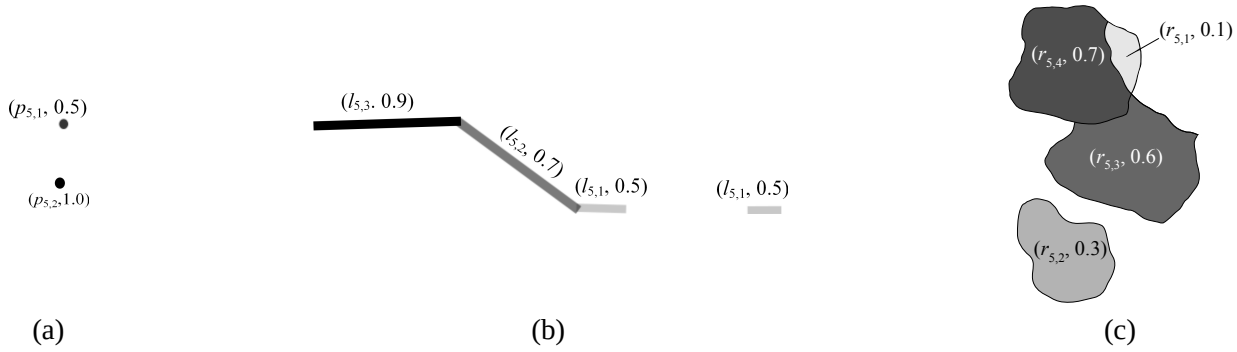


Figure 8: Spatial plateau difference: plateau point object $pp_5 = \mathit{fdifference}(pp_1, pp_2) = \langle (p_{5,1}, 0.5), (p_{5,2}, 1.0), p_5 \rangle$ (a), plateau line object $pl_5 = \mathit{fdifference}(pl_1, pl_2) = \langle (l_{5,1}, 0.5), (l_{5,2}, 0.7), (l_{5,3}, 0.9), l_5 \rangle$ (b), and plateau region object $pr_5 = \mathit{fdifference}(pr_1, pr_2) = \langle (r_{5,1}, 0.1), (r_{5,2}, 0.3), (r_{5,3}, 0.6), (r_{5,4}, 0.7), r_5 \rangle$ (c)

object. In all cases, computed component objects with the same membership value are always geometrically merged in the resulting spatial plateau object. Forming the difference of two objects po_1 and po_2 of the same spatial plateau data type principally means that each component object of po_2 must be geometrically subtracted from each component object of po_1 and that the membership value of the latter component object is diminished by the membership value of the former component object.

The difference of the spatial plateau objects in Figure 5 is illustrated in Figure 8. Two main kinds of spatial configurations can be distinguished. If the two component objects do not intersect, then the component object of po_1 is copied with its membership value into the resulting spatial plateau object. This is, for example, the case for the component region pair $r_{1,1}$ and $r_{2,1}$ in Figure 5c; the result is shown as the component region $r_{5,2}$ in Figure 8c with the original membership value of $r_{1,1}$.

Otherwise, the two component objects intersect, and up to two new component objects are stored in the resulting spatial plateau object. The local view relates to the common parts of both component objects. Here we can distinguish three cases to which we react in the same manner. Either the two component objects are geometrically equal, or the first component object is contained in the second component object, or both component objects have a proper intersection. In all three cases, if the membership value of the first component object is larger than the membership value of the second component object, the intersection is added as a new component object with the positive difference of both membership values to the resulting spatial plateau object. As an example for the latter case, we consider the component regions $r_{1,3}$ with the membership value 0.7 and $r_{2,3}$ with the membership value 0.6 in Figure 5c. The intersection of $r_{1,3}$ and $r_{2,3}$ is added as a new component region $r_{5,1}$ with the membership value $0.7 - 0.6 = 0.1$ to the resulting spatial plateau object (see Figure 8c).

The handling of the remaining unshared parts of each component object of the first spatial plateau object requires a global view and is performed in the same way as for the spatial plateau union operation. That is, from each component object of the first spatial plateau object we subtract the union object of the second spatial plateau object and assign the original membership value to the result of the subtraction.

4.1.3 Formal Definition of the Spatial Plateau Set Operations

Based on the informal descriptions of the spatial plateau set operations in Section 4.1.2, we now present their formal definitions. These definitions rest on the aforementioned, available, and well known geometric set operations *intersection* ($\otimes$), *union* ($\oplus$), and *difference* ($\ominus$) on the crisp spatial data types *point*, *line*, and *region* [41, 44]. The three geometric set operations are generic, that is, polymorphic, and the three spatial data types are closed under them. That is, the operations have the signature $\gamma : \alpha \times \alpha \to \alpha$ for $\gamma \in \{\otimes, \oplus, \ominus\}$

and $\alpha \in \{point, line, region\}$. This helps us ensure that the spatial plateau data types are closed under the spatial plateau set operations.

The formal definition of the spatial plateau set operations requires an auxiliary construction operator $\odot$ that enables us to insert a pair $(c,m) \in \alpha \times [0,1]$ of a component object and its membership value into the ordered representation of a spatial plateau object $po = \langle (o_1,m_1),\ldots,(o_n,m_n),o\rangle \in \phi(\alpha)$ for some $n \in \mathbb{N}$. We define:

$$po \odot (c,m) \;=\; \begin{cases} po & \text{if } c = \varnothing \text{ or } m = 0 \\ \langle (c,m), c\rangle & \text{if } po = \langle\rangle \text{ and } c \neq \varnothing \text{ and } m > 0 \\ \langle (o_1,m_1),\ldots,(o_i \oplus c, m_i),\ldots,(o_n,m_n), o \oplus c\rangle & \\ & \text{if } c \neq \varnothing \text{ and } n \geq 1 \text{ and } \exists i \in \{1,\ldots,n\} : m_i = m \\ \langle (o_1,m_1),\ldots,(o_i,m_i),(c,m),(o_{i+1},m_{i+1}),\ldots,(o_n,m_n), o \oplus c\rangle & \\ & \text{if } c \neq \varnothing \text{ and } n \geq 2 \text{ and } \exists i \in \{1,\ldots,n-1\} : m_i < m < m_{i+1} \\ \langle (c,m),(o_1,m_1),\ldots,(o_n,m_n), o \oplus c\rangle & \\ & \text{if } c \neq \varnothing \text{ and } n \geq 1 \text{ and } 0 < m < m_1 \\ \langle (o_1,m_1),\ldots,(o_n,m_n),(o,m), o \oplus c\rangle & \\ & \text{if } c \neq \varnothing \text{ and } n \geq 1 \text{ and } m > m_n \end{cases}$$

The interesting aspects are the third case in which a new component object c is merged with an existing component object o_i if their membership values are equal, and the merging of the union object o at the end of the representation with c. Note that $\odot$ is left-associative, that is, $po \odot (o_1,m_1) \odot (o_2,m_2) = (po \odot (o_1,m_1)) \odot (o_2,m_2)$. For $po \odot (o_1,m_1) \odot \ldots \odot (o_n,m_n)$ we also write $po \odot \bigodot_{i=1}^{n} (o_i,m_i)$.

We are now prepared to provide the formal definitions of the three spatial plateau set operations. We specify them in a polymorphic manner with the signature $\gamma_\phi : \phi(\alpha) \times \phi(\alpha) \to \phi(\alpha)$ for $\gamma_\phi \in \{fintersection, funion, fdifference\}$ and $\alpha \in \{point, line, region\}$. This means that the meaning and specification of the three operations is the same for all three type combinations. The construction operator $\odot$ ensures that the three spatial plateau data types are closed under them by maintaining the structural constraints of their definition given in Section 3.3. We assume two spatial plateau objects $po_1, po_2 \in \phi(\alpha)$ with $po_k = \langle (o_{k,1},m_{k,1}),\ldots,(o_{k,n_k},m_{k,n_k}), o_k\rangle$ for $k \in \{1,2\}$. Then the operation *fintersection* is defined as

$$fintersection(po_1,po_2) = \langle\rangle \odot \bigodot_{\substack{1 \leq i \leq n_1 \\ 1 \leq j \leq n_2}} (o_{1,i} \otimes o_{2,j}, \min(m_{1,i}, m_{2,j}))$$

This definition (like the others) uses an incremental strategy by starting with the empty spatial plateau object $\langle\rangle$ and incrementally adding local results from the intersections of component object pairs. As said before, this local view is sufficient here since each point of the result can be only obtained by the intersection of exactly one pair of component objects.

The operation *funion* is defined as

$$\begin{aligned} funion(po_1,po_2) \;=\; & \langle\rangle \odot \bigodot_{\substack{1 \leq i \leq n_1 \\ 1 \leq j \leq n_2}} (o_{1,i} \otimes o_{2,j}, \max(m_{1,i}, m_{2,j})) \\ & \odot \bigodot_{1 \leq i \leq n_1} (o_{1,i} \ominus (o_1 \otimes o_2), m_{1,i}) \\ & \odot \bigodot_{1 \leq j \leq n_2} (o_{2,j} \ominus (o_1 \otimes o_2), m_{2,j}) \end{aligned}$$

The first subexpression of the three-part expression represents the local view on the common part (intersection) of any pair of component objects. Each common part is annotated with the higher membership

value of the component objects. The second and third subexpressions represent the global view. From each component object we subtract the intersection of both union objects and maintain its membership value.

The operation *fdifference* is defined as

$$\begin{aligned} \mathit{fdifference}(po_1, po_2) \quad = \quad & \langle\rangle \odot \bigodot_{\substack{1 \le i \le n_1 \\ 1 \le j \le n_2}} (o_{1,i} \otimes o_{2,j}, m_{1,i} \dot{-} m_{2,j}) \\ & \odot \bigodot_{1 \le i \le n_1} (o_{1,i} \ominus (o_1 \otimes o_2), m_{1,i}) \end{aligned}$$

The first subexpression represents the local view on the common part of any pair of component objects. Each common part is annotated with the difference of the membership values of both component objects. The operator $\dot{-}$ has been defined in Section 4.1.1. If $o_{1,i} \otimes o_{2,j} = \varnothing$ or $m_{1,i} \dot{-} m_{2,j} = 0$ should hold, we obtain an empty component object, and the construction operator $\odot$ will prevent its insertion into the resulting spatial plateau object. The second subexpression represents the global view and determines the unshared part of each component object of the first spatial plateau object by preserving the original membership value.

4.2 Other Spatial Plateau Operations

Apart from the spatial plateau set operations in Section 4.1, a few other spatial plateau operations are of interest. Some of these operations cannot be found in the purely crisp domain but only in the fuzzy domain. We classify them into *fuzzy spatial object construction operations* (Section 4.2.1), *fuzzy spatial range operations* (Section 4.2.2), *containment and overlap predicates* (Section 4.2.3), and *membership value changing operations* (Section 4.2.4). Further, we define an order relation $<$ on the spatial data types: *point* $<$ *line* $<$ *region*.

4.2.1 Fuzzy Spatial Object Construction Operations

The operations of this category construct a new spatial plateau object from one or two existing spatial plateau objects. We first discuss a variation of the operation *fintersection* that is here applied to heterogeneous type combinations. Let $\alpha \in \{\mathit{point}, \mathit{line}\}$, $\beta \in \{\mathit{line}, \mathit{region}\}$, and $\alpha < \beta$. The operations have the signature $\mathit{fintersection} : \phi(\alpha) \times \phi(\beta) \to \phi(\alpha)$ and share the same definition as the operation *fintersection* in Section 4.1.3. The $\otimes$ operation takes a component point or line object as well as a component line or region object as operands and computes their geometric intersection. From the two membership values, it assigns the smaller one to the resulting component object.

The operation *fcontour* takes a plateau region object as an operand and stores its fuzzy boundary as a plateau line object. It has the signature $\mathit{fcontour} : \mathit{fregion} \to \mathit{fline}$. Its definition makes use of the crisp spatial operation $\mathit{contour} : \mathit{region} \to \mathit{line}$ [41] that yields the boundary of a crisp region object as a crisp line object. Let $pr \in \mathit{fregion}$ with $pr = \langle (r_1, m_1), \ldots, (r_n, m_n), r \rangle$. Then the operation *fcontour* is defined as

$$\mathit{fcontour}(pr) = \langle\rangle \odot \bigodot_{1 \le i \le n} (\mathit{contour}(r_i), m_i)$$

The operation *fvertices* returns the fuzzy corner points (fuzzy vertices) of a plateau line or region object and keeps them in a plateau point object. It has the signature $\mathit{fvertices} : \phi(\alpha) \to \mathit{fpoint}$ for $\alpha \in \{\mathit{line}, \mathit{region}\}$. We begin with the case that the operand object is a plateau line object $pl = \langle (l_1, m_1), \ldots, (l_n, m_n), l \rangle$. Each line object l_i consists of a set of linear segments defined by two end points or *vertices*. A main problem that impedes the definition of this operation is that only collecting the fuzzy vertices of *pl* is insufficient. The reason is that according to our definition of plateau line objects in Section 3.3, the component lines of *pl* may meet or overlap in single points. Since different component lines have different membership values, we must cater for assigning the highest membership value to each common meeting or overlapping point.

In a meeting situation, we have to distinguish two cases. The first case is that two fuzzy end points (p, m) and (q, m') of different component lines coincide. That is, $p = q$ and $m \neq m'$ holds. We handle this

situation by keeping both points (p,m) and (p,m') temporarily in a sequence (collecting phase represented by the auxiliary function *collectFPoints*$_1$ below) and by removing the pair with the lower membership value later (reducing phase represented by the auxiliary function *reduce* below). For this purpose, we leverage the crisp spatial operation *vertices* : $\alpha \rightarrow$ *point* for $\alpha \in \{line, region\}$ [41] that yields the corner points of a crisp line or region object. The second case is that a fuzzy end point of one component line touches a fuzzy interior point of another component line. The fuzzy interior point is not a vertex or corner point of the second component line. Hence, we use the operation *commonPoints* : *line* $\times$ *line* $\rightarrow$ *point* [41] to identify all touching points.

In an overlap situation, we consider intersection points as vertices and hence as part of the result. Again we use the operation *commonPoints* to compute them. All intersection points are collected (see function *collectFPoints*$_1$ below), and duplicate single points with different membership values are removed (see function *reduce* below).

More formally, collecting fuzzy vertices, touching points, and intersection points is performed by the function *collectFPoints*$_1$ as follows:

$$\begin{aligned} \mathit{collectFPoints}_1(pl) &= pp = \langle (p_1,m_1),\ldots,(p_k,m_k),p\rangle \\ &= \langle\rangle \odot \bigodot_{1\le i\le n} (\mathit{vertices}(l_i),m_i) \\ &\quad \odot \bigodot_{\substack{1\le i\le n-1 \\ i+1\le j\le n}} (\mathit{commonPoints}(l_i,l_j),\min(m_i,m_j)) \end{aligned}$$

Note that the result of the function *collectFPoints*$_1$ is, in general, *not* a plateau point object. Let $pp = \langle (p_1,m_1),\ldots,(p_i,m_i),\ldots,(p_k,m_k),p\rangle$ be the result of this function. The $\odot$ operator already ensures that duplicate (p_i,m_i) elements delivered by both functions *vertices* and *commonPoints* are merged and that $m_i < m_j$ holds for all $1 \le i < j \le k$. However, *pp* may still keep elements (p_i,m_i) and (p_j,m_j) with $i \neq j$ such that $p_i \otimes p_j \neq \varnothing$. This means there are single points that are annotated with different membership values. This contradicts the definition of the data type *fpoint*. Only the highest membership value of a point should be maintained in *pp*. The function *reduce* solves this problem and eliminates for a single component point (p,m) all those single component points (p,m') in *pp* with $m' < m$. It is defined as

$$\mathit{reduce}(pp) = \bigtimes_{\substack{1\le i\le n-1 \\ i+1\le j\le n}} \text{if } p_i \otimes p_j \neq \varnothing \text{ then } \mathit{replace}(pp, m_i, p_i \ominus (p_i \otimes p_j))$$

The result is now a correct plateau point object. The function makes use of an auxiliary function *replace* that replaces elements in a sequence *in situ*. The replacement function is defined as

$$\mathit{replace}(pp, m_i, p_i') = \langle (p_1,m_1),\ldots,(p_i',m_i),\ldots,(p_k,m_k),p\rangle$$

We are now able to define the operation *fvertices* on a plateau line object *pl* as follows:

$$\mathit{fvertices}(pl) = \mathit{reduce}(\mathit{collectFPoints}_1(pl))$$

The definition of the operation *fvertices* on a plateau region object *pr* is given as

$$\mathit{fvertices}(pr) = \mathit{fvertices}(\mathit{fcontour}(pr))$$

The next spatial plateau operation we discuss is the operation *fcommonPoints*. It takes two plateau line objects, or two plateau region objects, or a plateau line object and a plateau region object as operands and computes their shared single fuzzy points as a plateau point object. That is, it has the signature *fcommonPoints* : $\phi(\alpha) \times \phi(\beta) \rightarrow$ *fpoint* for $\alpha,\beta \in \{line, region\}$ with $\alpha \le \beta$. We first consider the case of two plateau line objects. Let $pl_1 = \langle (l_{1,1},m_{1,1}),\ldots,(l_{1,n},m_{1,n}),l_1\rangle, pl_2 = \langle (l_{2,1},m_{2,1}),\ldots,(l_{2,s},m_{2,s}),l_2\rangle \in$ *fline*. We collect all fuzzy touching points and intersection points by the function *collectFPoints*$_2$ as follows:

$$\begin{aligned} \mathit{collectFPoints}_2(pl_1, pl_2) &= pp = \langle (p_1, m_1), \ldots, (p_k, m_k), p \rangle \\ &= \langle\rangle \odot \bigodot_{\substack{1 \le i \le n \\ 1 \le j \le s}} (\mathit{commonPoints}(l_{1,i}, l_{2,j}), \min(m_{1,i}, m_{2,j})) \end{aligned}$$

The result of the function *collectFPoints*$_2$ is, in general, *not* a plateau point object. The reason is the same as above for the function *collectFPoints*$_1$. Let *pp* be the result of this function. We use the function *reduce* from above to preserve only single point results with their highest membership value. This enables us to define the operation *fcommonPoints* on $pl, pl_1, pl_2 \in \mathit{fline}$ and $pr, pr_1, pr_2 \in \mathit{fregion}$ as follows:

$$\begin{aligned} \mathit{fcommonPoints}(pl_1, pl_2) &= \mathit{reduce}(\mathit{collectFPoints}_2(pl_1, pl_2)) \\ \mathit{fcommonPoints}(pl, pr) &= \mathit{fcommonPoints}(pl, \mathit{fcontour}(pr)) \\ \mathit{fcommonPoints}(pr_1, pr_2) &= \mathit{fcommonPoints}(\mathit{fcontour}(pr_1), \mathit{fcontour}(pr_2)) \end{aligned}$$

The operation *fcommonBorder* computes the shared boundary line parts of two plateau region objects as a plateau line object. That is, it has the signature $\mathit{fcommonBorder} : \mathit{fregion} \times \mathit{fregion} \to \mathit{fline}$. Let $pr_1, pr_2 \in \mathit{fregion}$. Then the operation *fcommonBorder* is defined as

$$\mathit{fcommonBorder}(pr_1, pr_2) = \mathit{fintersection}(\mathit{fcontour}(pr_1), \mathit{fcontour}(pr_2))$$

4.2.2 Fuzzy Spatial Range Operations

The operations of this category take a single spatial plateau object and perform a *fuzzy spatial range operation* based on chosen membership values. Hence, they compute a *fuzzy spatial selection*. For this purpose, based on a type *real* for representable real numbers (for example, float or double values), we define the data type $mv = \{m \mid m \in \mathit{real}, 0 < m \le 1\}$ to denote single membership values.

The simplest operation of this category is named *fslice* and has the signature $\mathit{fslice} : \phi(\alpha) \times mv \to \phi(\alpha)$ for $\alpha \in \{\mathit{point}, \mathit{line}, \mathit{region}\}$. For a given spatial plateau object *po* and a membership value *m*, it returns the component object of *po* whose membership value is equal to *m*. This makes it possible to answer a query like "Find all regions where English speaking residents account for 50% of the entire population". Let $po = \langle (o_1, m_1), \ldots, (o_n, m_n), o \rangle \in \phi(\alpha)$ for some $n \in \mathbb{N}$. Then the operation *fslice* is defined as

$$\mathit{fslice}(po, m) = \begin{cases} \langle (o_i, m_i), o_i \rangle & \text{if } \exists\, 1 \le i \le n : m_i = m \\ \langle\rangle & \text{otherwise} \end{cases}$$

The next operation we consider is an extension of the operation *fslice* to a range or interval of membership values. The operation is called *fsliceByRange* and has the signature $\mathit{fsliceByRange} : \phi(\alpha) \times mv \times mv \times \mathit{bool} \times \mathit{bool} \to \phi(\alpha)$ for $\alpha \in \{\mathit{point}, \mathit{line}, \mathit{region}\}$. The two values of type *mv* indicate the start and end value of the membership interval. The two Boolean values indicate whether the start value and the end value, respectively, are included (*true*) or excluded (*false*) from the membership interval. This operation helps a user answer a query like "Find all regions where English speaking residents account for 50% to 70% of the entire population". The operation *fsliceByRange* is defined as

$$
\begin{aligned}
&\mathit{fsliceByRange}(po, m_1, m_2, b_1, b_2) = \\
&\begin{cases}
\langle (o_{i_1}, m_{i_1}), \ldots, (o_{i_k}, m_{i_k}), o' \rangle & \text{if the following conditions hold:} \\
& \text{(i)} \quad \exists 1 \leq k \leq n \; \exists i_1, \ldots, i_k \in \{1, \ldots, n\} : 1 \leq i_1 < \ldots < i_k \leq n \\
& \text{(ii)} \quad \text{if } b_1 \wedge b_2 \text{ then } m_1 \leq m_{i_1} < \ldots < m_{i_k} \leq m_2 \\
& \qquad \text{else if } \neg b_1 \wedge b_2 \text{ then } m_1 < m_{i_1} < \ldots < m_{i_k} \leq m_2 \\
& \qquad \text{else if } b_1 \wedge \neg b_2 \text{ then } m_1 \leq m_{i_1} < \ldots < m_{i_k} < m_2 \\
& \qquad \text{else } m_1 < m_{i_1} < \ldots < m_{i_k} < m_2 \\
& \text{(iii)} \quad \forall j \in \{1, \ldots, n\} - \{i_1, \ldots, i_k\} : \\
& \qquad \text{if } b_1 \wedge b_2 \text{ then } m_j < m_1 \vee m_j > m_2 \\
& \qquad \text{else if } \neg b_1 \wedge b_2 \text{ then } m_j \leq m_1 \vee m_j > m_2 \\
& \qquad \text{else if } b_1 \wedge \neg b_2 \text{ then } m_j < m_1 \vee m_j \geq m_2 \\
& \qquad \text{else } m_j \leq m_1 \vee m_j \geq m_2 \\
& \text{(iv)} \quad o' = \bigoplus_{i_1 \leq j \leq i_k} o_j \\
\langle \rangle & \text{otherwise}
\end{cases}
\end{aligned}
$$

Note that $\mathit{fslice}(po, m) = \mathit{fsliceByRange}(po, m, m, \mathit{true}, \mathit{true})$ holds. This shows that the operation *fslice* is a specialization of the operation *fsliceByRange*.

Finally, we propose another specialization of the operation *fsliceByRange*. This operation is called *fplateauCut* and has the signature $\mathit{fplateauCut} : \phi(\alpha) \times mv \times \mathit{bool} \times \mathit{bool} \rightarrow \phi(\alpha)$. It is defined as

$$
\mathit{fplateauCut}(po, m, u, b) = \begin{cases} \mathit{fsliceByRange}(po, m, 1, b, \mathit{true}) & \text{if } u \\ \mathit{fsliceByRange}(po, 0, m, \mathit{false}, b) & \text{if } \neg u \end{cases}
$$

This means that this operation takes a membership value m as a separation value and returns either the complete upper part ($u = \mathit{true}$) or the complete lower part ($u = \mathit{false}$) of a spatial plateau object *po* by including ($b = \mathit{true}$) or excluding ($b = \mathit{false}$) the object *po* at the membership value m. In the case of $u = \mathit{true}$ and $b = \mathit{true}$, we obtain the concept of an *α-cut* in fuzzy set theory [56, 42]. If $u = \mathit{true}$ and $b = \mathit{false}$ holds, we obtain a *strong α-cut*. In our spatial context, we speak about (*strong*) *spatial α-cuts*.

4.2.3 Containment and Overlap Predicates

This category only includes the two topological predicates *fin* for *fuzzy containment* testing and *fisect* for fuzzy intersection testing. Both predicates are defined in an ad hoc manner[3]. The predicate *fin* checks whether a spatial plateau object is located within another spatial plateau object. Its signature is $\mathit{fin} : \phi(\alpha) \times \phi(\beta) \rightarrow \mathit{bool}$ for all $\alpha, \beta \in \{\mathit{point}, \mathit{line}, \mathit{region}\}$ with $\alpha \leq \beta$. We assume two spatial plateau objects $po_1 \in \phi(\alpha)$ and $po_2 \in \phi(\beta)$ with $po_k = \langle (o_{k,1}, m_{k,1}), \ldots, (o_{k,n_k}, m_{k,n_k}), o_k \rangle$ for $k \in \{1, 2\}$. Then the operation *fin* is defined as

$$
\mathit{fin}(po_1, po_2) = (\forall 1 \leq i \leq n_1 \; \exists 1 \leq j \leq n_2 : o_{1,i} \; \mathit{inside}_c \; o_{2,j} \wedge m_{1,i} \leq m_{2,j})
$$

This operation makes use of the *topological cluster predicate* $\mathit{inside}_c : \alpha \times \beta \rightarrow \mathit{bool}$ on complex spatial objects [44] with α and β as defined and constrained above. The predicate inside_c checks here whether a crisp component object is located within another crisp component object; it is independent of the spatial data type combination considered. The predicate *fin* further requires that for each containment relationship

[3] We will present thorough and systematically developed concepts of *fuzzy topological relationships* (like *fuzzy overlap*, *fuzzy meet*) and *fuzzy directional relationships* (like *fuzzy north*, *fuzzy southeast*) in future publications.

found for two component objects, the membership value of the first component object must not be larger than the membership value of the second component object.

The predicate *fisect* tests whether any two spatial plateau objects intersect geometrically. Its signature is $\mathit{fisect} : \phi(\alpha) \times \phi(\beta) \to \mathit{bool}$ for all $\alpha, \beta \in \{\mathit{point}, \mathit{line}, \mathit{region}\}$ with $\alpha \leq \beta$. We assume two spatial plateau objects po_1 and po_2 as defined above. Then the operation *fisect* is defined as

$$\mathit{fisect}(po_1, po_2) = \mathit{overlap}_c(o_1, o_2)$$

The predicate $\mathit{overlap}_c : \alpha \times \beta \to \mathit{bool}$ is also a topological cluster predicate on complex spatial objects. It tests here whether the union objects o_1 and o_2 intersect. Then the two spatial plateau objects po_1 and po_2 intersect too. Membership values do not play a role in the definition. The reason is that a test for intersection implies the existence of a common part shared by both spatial plateau objects. For such a common part, the membership values must be larger than 0 in both spatial plateau objects. If the intersection of the union objects is non-empty, this requirement is ensured. Hence, alternatively, we could define that $\mathit{fisect}(po_1, po_2) = (o_1 \otimes o_2 \neq \varnothing)$ must hold. The degree of existence is not relevant here.

4.2.4 Membership Value Changing Operations

The operation *relabel* with the signature $\mathit{relabel} : \phi(\alpha) \times \mathit{mv} \times \mathit{mv} \to \phi(\alpha)$ makes it possible to change the membership value of a component object of a spatial plateau object. If the new membership value is not among the membership values of the other component objects, the old membership value of the component object is replaced, and the order of the component objects has to be updated if necessary. However, if the new membership value already exists among the other membership values, a geometric union of the respective component objects has to be performed. Let $po \in \phi(\alpha)$ with $po = \langle (o_1, m_1), \ldots, (o_n, m_n), o \rangle \in \phi(\alpha)$ for some $n \in \mathbb{N}$, and let $m, m' \in \mathit{mv}$. Then we define the operation *relabel* as

$$\mathit{relabel}(po, m, m') = \begin{cases} \langle (o_1, m_1), \ldots, (o_{i-1}, m_{i-1}), (o_{i+1}, m_{i+1}), \ldots, (o_n, m_n), o \ominus o_i \rangle \odot (o_i, m') & \text{if } \exists 1 \leq i \leq n : m_i = m \\ po & \text{otherwise} \end{cases}$$

4.3 Context-Dependent Spatial Plateau Operations

Several binary spatial plateau operations in the previous sections map two membership values stemming from two component objects of different spatial plateau objects into a single membership value. For this purpose, they apply the *standard fuzzy membership operations min*, *max*, and $\dot{-}$. The operation *fintersection* uses the minimum function *min*. The operations *funion*, *fvertices*, and *fcommonPoints* leverage the maximum function *max*. Finally, the operation *fdifference* makes use of the "positive subtraction" function $\dot{-}$. The standard fuzzy membership operations behave precisely as the corresponding operations for crisp sets when the range of membership values is restricted to the set $\{0, 1\}$. In this way, the standard fuzzy membership operations are generalizations of the classical set operations.

However, the presented generalizations are not the only possible interpretations of fuzziness, contrary to their crisp counterparts. For each of the three standard fuzzy membership operations above, there exists a broad class of functions whose members qualify as fuzzy generalizations of the classical operations as well and could be of interest for spatial applications. Since the three standard fuzzy membership operations are not unique, different functions may be suitable to represent these operations in different contexts. This means that not only the membership functions of fuzzy sets but also the operations on fuzzy sets are *context-dependent*. Functions that qualify as fuzzy membership intersections and fuzzy membership unions are usually referred to in the literature as *t-norms* and *t-conorms* respectively [32]. But their application-specific selection and usage requires expert knowledge.

A *t-norm* T is a function $T : [0,1] \times [0,1] \to [0,1]$ with the following four properties for all $a, b, d \in [0,1]$:

(i)	$T(a,1) = a$	(*boundary condition, identity element*)
(ii)	$T(a,b) = T(b,a)$	(*commutativity*)
(iii)	$b \leq d$ implies $T(a,b) \leq T(a,d)$	(*monotonicity*)
(iv)	$T(a,T(b,d)) = T(T(a,b),d)$	(*associativity*)

Condition (i) ensures that the fuzzy membership intersection becomes the classical set intersection when the interval $[0,1]$ is reduced to the crisp set $\{0,1\}$ since we obtain $T(0,1) = 0$ and $T(1,1) = 1$. Condition (ii) states that the fuzzy membership intersection is symmetric, that is, indifferent to the order in which the sets to be combined are considered. Further, we can conclude that $T(1,0) = 0$ holds. Condition (iii) expresses that a decrease in the degree of membership cannot lead to an increase in the degree of membership in the intersection. Incorporating Condition (ii), T is non-decreasing in both arguments. Further, we can see that $0 \leq 1$ implies $T(0,0) \leq T(0,1) = 0$ and $T(0,0) = 0$. Condition (iv) allows us to take the intersection of any number of membership values in any order of pairwise grouping desired; this enables us to extend the fuzzy membership intersection operation to more than two membership values.

The following examples show some t-norms that are frequently used as fuzzy membership intersections (each defined for all $a,b \in [0,1]$).

$$
\begin{array}{rcll}
T_m(a,b) & = & \min(a,b) & (\textit{standard intersection, minimum t-norm, Gödel t-norm}) \\
T_b(a,b) & = & \max(0,a+b-1) & (\textit{bounded difference, Lukasiewicz t-norm}) \\
T_p(a,b) & = & ab & (\textit{algebraic product, product t-norm}) \\
T^*(a,b) & = & \begin{cases} a & \text{if } b = 1 \\ b & \text{if } a = 1 \\ 0 & \text{otherwise} \end{cases} & (\textit{drastic intersection, drastic t-norm})
\end{array}
$$

One can easily show that for all $a,b \in [0,1]$ holds that

$$T^*(a,b) \leq T_b(a,b) \leq T_p(a,b) \leq T_m(a,b)$$

In fact, if T is any t-norm, then for all $a,b \in [0,1]$ holds that

$$T^*(a,b) \leq T(a,b) \leq T_m(a,b)$$

Hence, a geoscientist could make use of one of the aforementioned t-norms, or apply other t-norms like

$$
\begin{array}{rcll}
T_{nm}(a,b) & = & \begin{cases} \min(a,b) & \text{if } a+b > 1 \\ 0 & \text{otherwise} \end{cases} & (\textit{nilpotent minimum}) \\
T_w(a,b) & = & 1 - \min(1,[(1-a)^w + (1-b)^w]^{1/w}) \quad (w > 0) & (\textit{Yager t-norms}) \\
T_s(a,b) & = & (\max(0,a^p + b^p - 1))^{1/p} \quad (p \neq 0) & (\textit{Schweizer \& Sklar t-norms}) \\
T_{Hp}(a,b) & = & \begin{cases} 0 & \text{if } a = b = 0 \\ \frac{ab}{a+b-ab} & \text{otherwise} \end{cases} & (\textit{Hamacher product})
\end{array}
$$

or invent own t-norms tailored to respective spatial applications. We define a set *tnorm* that contains the shortcuts of the names of all listed t-norms:

$$tnorm \quad = \quad \{T_m, T_b, T_p, T^*, T_{nm}, T_w, T_s, T_{Hp}\}$$

T-conorms are dual to t-norms under the order-reversing operation which assigns $1-x$ to x on the interval $[0,1]$. Given a t-norm T and the complementary t-conorm C, we obtain the following two dual relationships:

$$
\begin{array}{rcl}
T(a,b) & = & 1 - C(1-a,1-b) \\
C(a,b) & = & 1 - T(1-a,1-b)
\end{array}
$$

It follows that a t-conorm satisfies similar conditions like a t-norm. These conditions can be used for an equivalent definition of t-conorms independently of t-norms. A *t-conorm* C is a function $C : [0,1] \times [0,1] \to [0,1]$ with the following four properties for all $a, b, d \in [0,1]$:

(i)	$C(a,0) = a$	(*boundary condition, identity element*)
(ii)	$C(a,b) = C(b,a)$	(*commutativity*)
(iii)	$b \leq d$ implies $C(a,b) \leq C(a,d)$	(*monotonicity*)
(iv)	$C(a,C(b,d)) = C(C(a,b),d)$	(*associativity*)

A comparison of these conditions with the conditions for t-norms shows that they only differ in the boundary condition. Conditions (i) to (iii) ensure that the fuzzy membership union becomes the classical set union when the interval $[0,1]$ is reduced to the crisp set $\{0,1\}$. We obtain $C(0,0) = 0$, $C(0,1) = C(1,0) = C(1,1) = 1$. Otherwise, the conditions have the same rationale as those for t-norms.

The following examples show some t-conorms that are frequently used as fuzzy membership unions (each defined for all $a, b \in [0,1]$).

$$
\begin{array}{lcll}
C_m(a,b) & = & \max(a,b) & (\textit{standard union, maximum t-conorm}) \\
C_b(a,b) & = & \min(a+b,1) & (\textit{bounded sum}) \\
C_p(a,b) & = & a+b-ab & (\textit{algebraic sum, probabilistic sum}) \\
C^*(a,b) & = & \begin{cases} a & \text{if } b = 0 \\ b & \text{if } a = 0 \\ 1 & \text{otherwise} \end{cases} & (\textit{drastic union, drastic t-conorm})
\end{array}
$$

T_m and C_m, T_b and C_b, T_p and C_p, and T^* and C^* are pairwise dual. One can easily show that for all $a, b \in [0,1]$ holds that

$$C_m(a,b) \leq C_p(a,b) \leq C_b(a,b) \leq C^*(a,b)$$

If C is any t-conorm, then for all $a, b \in [0,1]$ holds that

$$C_m(a,b) \leq C(a,b) \leq C^*(a,b)$$

Also the other aforementioned t-norms T_{nm}, T_w, T_s, and T_{Hp} have dual t-conorms C_{nm}, C_w, C_s, and C_{Es}:

$$
\begin{array}{lcll}
C_{nm}(a,b) & = & \begin{cases} \max(a,b) & \text{if } a+b < 1 \\ 1 & \text{otherwise} \end{cases} & (\textit{nilpotent minimum}) \\
C_w(a,b) & = & \min(1, (a^w + b^w)^{1/w}) \quad (w > 0) & (\textit{Yager t-conorms}) \\
C_s(a,b) & = & 1 - (\max(0, (1-a)^p + (1-b)^p - 1))^{1/p} \quad (p \neq 0) & (\textit{Schweizer \& Sklar t-conorms}) \\
C_{Es}(a,b) & = & \frac{a+b}{1+ab} & (\textit{Einstein sum})
\end{array}
$$

We define a set *tconorm* that contains the shortcuts of the names of all listed t-conorms:

$$\textit{tconorm} \quad = \quad \{C_m, C_b, C_p, C^*, C_{nm}, C_w, C_s, C_{Es}\}$$

For fuzzy membership difference, we have so far specified the membership function $\dot{-}$. An alternative is the membership function *mdiff* defined for all $a, b \in [0,1]$ as

$$\textit{mdiff}(a,b) \quad = \quad \min(a, 1-b)$$

This function can be explained as follows: For two fuzzy sets $\tilde{A}$ and $\tilde{B}$ on $\mathbb{R}^2$, the complement of $\tilde{A}$ is $c\tilde{A} = \{(p, \mu_{c\tilde{A}}(p)) \,|\, p \in \mathbb{R}^2, \mu_{c\tilde{A}}(p) = 1 - \mu_{\tilde{A}}(p)\}$, and the fuzzy difference between $\tilde{A}$ and $\tilde{B}$ is $\tilde{A} - \tilde{B} = \tilde{A} \cap c\tilde{B} = \{(p, \mu_{\tilde{A} \cap c\tilde{B}}(p)) \,|\, p \in \mathbb{R}^2, \mu_{\tilde{A} \cap c\tilde{B}}(p) = \min(\mu_{\tilde{A}}(p), 1 - \mu_{\tilde{B}}(p))\}$. We collect the two alternatives for fuzzy membership difference in the set $\textit{diff} = \{\dot{-}, \textit{mdiff}\}$.

Besides the t-norms, the t-conorms, and the *diff* functions, we also provide the two additional special functions $\textit{first}, \textit{second} : [0,1] \times [0,1] \to [0,1]$ that give priority either to the first or to the second membership argument. They are defined for all $a, b \in [0,1]$ as

$$\begin{aligned} \mathit{first}(a,b) &= a \\ \mathit{second}(a,b) &= b \end{aligned}$$

For example, assume that a polluted river modeled as a fuzzy line flows across a city modeled as a fuzzy region, and an environmental science expert wants to know the length of the river inside this city in order to calculate the amount of decontamination liquid that needs to be poured into the river to cope with the polluted water. Then the membership values of the fuzzy line are relevant and should be preserved while the membership values of the fuzzy region can be ignored. Hence, the function *first* would be used.

For the extension of the operations *fintersection*, *funion*, *fvertices*, *fcommonPoints*, and *fdifference*, we introduce the three types *itype* (intersection type), *utype* (union type), and *dtype* (difference type). Note that these three types do not contain the functions themselves but only function identifiers. Depending on a selected function identifier, the corresponding membership function is called in one of the five spatial plateau operations.

$$\begin{aligned} \mathit{itype} &= \mathit{tnorm} \cup \{\mathit{first}, \mathit{second}\} \\ \mathit{utype} &= \mathit{tconorm} \cup \{\mathit{first}, \mathit{second}\} \\ \mathit{dtype} &= \mathit{diff} \cup \{\mathit{first}, \mathit{second}\} \end{aligned}$$

Besides the standard form of each operation given in Section 4.1.3, we offer each operation also in a version with one additional parameter of type *itype*, *utype*, or *dtype* so that we obtain the following extended signatures for $\alpha, \beta \in \{\mathit{point}, \mathit{line}, \mathit{region}\}$ with $\alpha \leq \beta$:

$$\begin{aligned} &\mathit{fintersection}: && \phi(\alpha) \times \phi(\alpha) \times \mathit{itype} && \rightarrow && \phi(\alpha) \\ &\mathit{funion}: && \phi(\alpha) \times \phi(\alpha) \times \mathit{utype} && \rightarrow && \phi(\alpha) \\ &\mathit{fvertices}: && \phi(\alpha) \times \mathit{utype} && \rightarrow && \mathit{fpoint} \\ &\mathit{fcommonPoints}: && \phi(\alpha) \times \phi(\beta) \times \mathit{utype} && \rightarrow && \mathit{fpoint} \\ &\mathit{fdifference}: && \phi(\alpha) \times \phi(\alpha) \times \mathit{dtype} && \rightarrow && \phi(\alpha) \end{aligned}$$

The definition of the operations is as before but depending on the last parameter the corresponding *t*-norm, *t*-conorm, *diff* function, *first* function, or *second* function respectively is selected and evaluated.

4.4 Mapping Operations between the Spatial Plateau Algebra and the Spatial Algebra 2D

The Spatial Plateau Algebra, as it has been defined so far, is closed under operations. This means that its operations do not leave the algebra or type system. But this can be useful in applications. In particular, the mapping of objects between the Spatial Plateau Algebra and our Spatial Algebra 2D is of interest. The Spatial Algebra 2D provides a comprehensive collection of spatial data types, spatial operations, and spatial (topological and directional) predicates for two-dimensional geometric data. The Spatial Plateau Algebra offers the operations *label*, *select*, and *detach* to move from one algebra to the other.

The operation *label* has the signature $\mathit{label} : \alpha \times \mathit{mv} \rightarrow \phi(\alpha)$ for $\alpha \in \{\mathit{point}, \mathit{line}, \mathit{region}\}$. This means it takes a spatial object and a membership value as operands and constructs a corresponding spatial plateau object. Let $o \in \alpha$ and $m \in \mathit{mv}$. Then we define the operation *label* as

$$\mathit{label}(o,m) = \langle (o,m), o \rangle \in \phi(\alpha)$$

By using the operation *funion*, larger spatial plateau objects can be assembled from spatial objects. For example, for $o_1, o_2 \in \alpha$ and $m_1, m_2 \in \mathit{mv}$, the term $\mathit{funion}(\mathit{label}(o_1,m_1), \mathit{label}(o_2,m_2))$ constructs the spatial plateau object $\langle (o_1 \ominus o_2, m_1), (o_2, m_2), o_1 \oplus o_2 \rangle$ if $m_1 < m_2$, $\langle (o_2 \ominus o_1, m_2), (o_1, m_1), o_1 \oplus o_2 \rangle$ if $m_2 < m_1$, or $\langle (o_1 \oplus o_2, m_1), o_1 \oplus o_2 \rangle$ if $m_1 = m_2$. Note that o_1 and o_2 could intersect.

The operation *component* has the signature $\phi(\alpha) \times \mathit{mv} \rightarrow \alpha$. It takes a spatial plateau object and a membership value as operands and returns the component object with this membership value if it exists. Let $po \in \phi(\alpha)$ with $po = \langle (o_1,m_1), \ldots, (o_n,m_n), o \rangle \in \phi(\alpha)$ for some $n \in \mathbb{N}$ and $m \in \mathit{mv}$. Then we define the operation *component* as

$$component(po, m) = \begin{cases} o_i & \text{if } \exists 1 \leq i \leq n : m_i = m \\ \langle\rangle & \text{otherwise} \end{cases}$$

The operation *detach* has the signature $detach : \phi(\alpha) \to \alpha$. It removes all labels of a spatial plateau object and returns the union object obtained from the geometric union of the component objects as a spatial object. Geometric operations and predicates can then be applied to this spatial object. Let $po \in \phi(\alpha)$ with $po = \langle (o_1, m_1), \ldots, (o_n, m_n), o \rangle \in \phi(\alpha)$ for some $n \in \mathbb{N}$. Then we define the operation *detach* as

$$detach(po) = o \in \alpha$$

5 Embedding Spatial Plateau Algebra Concepts into Query Languages

In the previous section we have defined a number of operations on fuzzy spatial objects that are represented as spatial plateau objects. In this section, we will show how spatial plateau data types can be integrated into a database schema and how spatial plateau operations can be embedded into an extension of the Structured Query Language (SQL), which is the standard query language for object-relational databases. In the following, we present three simplified scenarios that illustrate Spatial Plateau Algebra concepts and their possible embedding into table schemas and SQL-like queries.

5.1 Scenario 1: Homeland Security

Secret services are interested in the prevention of terroristic activities. Important information about each terrorist is which refuges they have had, which routes they have taken, and which areas have been their focus of operation. It is evident that this information is afflicted with spatial fuzziness or vagueness since the knowledge of the terrorist's presence at a certain location is often not fully known but incomplete. We use fuzzy spatial objects to represent the inherent vagueness of these locations. From a database perspective we deploy our spatial plateau data types *fpoint*, *fline*, and *fregion* as *attribute data types* in the same way as we use standard data types like *integer* or *date* as attribute data types. This allows us to store the above information about terrorists in all database models that support the concept of attributes to describe properties of entities, and leads thus to a database model independence. Examples of such models are the relational, nested relational, object-oriented, and object-relational data models. In a relational table schema, we can model the information about terrorists as follows:

terrorist(id: *integer*, name: *string*, refuge: *fpoint*, route: *fline*, active_area: *fregion*)

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

Looking at the table schema, we observe the equal treatment of all standard and non-standard data types. This is possible for non-standard data types since they are modeled as *abstract data types*. This means that the internal, complex structure of fuzzy spatial objects (spatial plateau objects) is hidden from the user and that information about these objects can only be obtained by high-level operations, which are deployed in fuzzy spatial queries.

The first query asks for the locations where any two terrorists have definitely or possibly taken the same refuge. It makes use of the topological predicate *fisect* and the spatial plateau set operation *fintersection*. Common locations are stored with the minimum degree of membership in the new attribute *common_refuge*.

select A.id, B.id, **fintersection**(A.refuge, B.refuge) **as** common_refuge
from terrorist **as** A, terrorist **as** B
where A.id $\neq$ B.id **and fisect**(A.refuge, B.refuge)

The following query determines the names of terrorists and the locations where their routes have definitely or possibly crossed each other. The term $\langle\rangle$ denotes the empty spatial plateau object (here empty spatial plateau point).

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

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

select **label**(**convex_hull**(**detach**(**fplateauCut**(**aggr_funion**(refuge), 1.0, *true*, *true*))), 1.0)
as known_sphere
from terrorist

The function *aggr_funion* is a new *fuzzy spatial aggregation function* that here computes the fuzzy geometric union of a collection of spatial plateau point objects from the column *refuge*. Since we are only interested in the known refuges having the membership value 1.0, we use the operation *fplateauCut* with appropriate parameters to identify these. We interpret the convex hull of the remaining refuge locations as the known sphere where the terrorists have been definitely active. For a given set of points, its convex hull is the smallest convex polygon that encloses all points. Since the operation *convex_hull* is a crisp spatial operation, we first detach all membership values from the spatial plateau point object obtained from the operation *fplateauCut*. We use the operation *label* to lift the crisp region object obtained from the operation *convex_hull* to a spatial plateau region object of type *fregion*.

The final query for this scenario asks for the areas of activity where *all* terrorists have been with a membership value between 0.12 and 0.78.

select **aggr_fintersection**(restricted_area) **as** common_restricted_area
from (**select fsliceByRange**(active_area, 0.12, 0.78, *true*, *true*) **as** restricted_area **from** terrorist)

For each terrorist, by using the operation *fsliceByRange*, the nested SQL query determines the area of activity with membership values between 0.12 and 0.78 as a spatial plateau region. The outer SQL query applies the new fuzzy spatial aggregation operation *aggr_fintersection* that here computes the fuzzy geometric intersection of a collection of spatial plateau region objects from the column *restricted_area*. Two main cases are possible. First, the nested SQL query produces a tuple with an empty spatial plateau region $\langle\rangle$. Then the aggregation will inevitably produce the empty spatial plateau region $\langle\rangle$. Second, all tuples have non-empty spatial plateau regions. Then the fuzzy spatial aggregation function yields either an empty (not all terrorists have been at the same location) or a non-empty spatial plateau region (there is a common area of terroristic activity).

5.2 Scenario 2: Ecological Application

This scenario assumes an ecological database about weather and soil data with the following table schemas:

weather(climate: *string*, area: *fregion*)
soil(quality: *string*, area: *fregion*)

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

The first query asks for the driest areas with a membership value of more than 95% where a lack of water is a problem for cultivation.

select **fplateauCut**(area, 0.95, *true*, *false*) **as** extreme_dry_area
from weather
where climate = "dry"

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

select **funion**(dry_area, bad_soil) **as** bad_region
from (**select aggr_funion**(area) **as** dry_area **from** weather **where** climate = "dry"),
(**select aggr_funion**(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 regions and bad soil quality regions respectively. Each table contains a single tuple with a single attribute value of type *fregion*. In the *select* clause, we compute the fuzzy union of the two attribute values of both tuples. The larger membership values prevail for each point.

For each kind of climate, the last query for this scenario determines the numerical area measures of those weather zones that we can definitely classify and that we can only vaguely classify.

select climate,
area(**detach**(**fslice**(**aggr_funion**(area), 1.0))) **as** definite_area,
area(**detach**(**fplateauCut**(**aggr_funion**(area), 1.0, *false*, *false*))) **as** vague_area
from weather
group by climate

Note that the operation *area* is a crisp operation on the spatial data type *region* and that for the empty crisp region $\langle\rangle$ holds that $area(\langle\rangle) = 0$. Further, for the term **detach**(**fslice**(**aggr_funion**(area), 1.0)) we could also write the simpler term **component**(**aggr_funion**(area), 1.0).

5.3 Scenario 3: Environmental Application

Pollution is nowadays a central environmental problem and causes an increasing number of environmental damages. Important examples are air pollution, oil soiling, and radioactive contamination. 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: *fregion*)
land use(use: *string*, area: *fregion*)

The first query asks for inhabited areas which are air polluted. In order to accentuate heavily polluted areas and largely inhabited areas and to understate hardly polluted areas and hardly inhabited areas, we make use of the product *t*-norm (see Section 4.3).

select **fintersection**(**aggr_funion**(zone), **aggr_funion**(area), T_p)
from pollution, land_use
where use = "inhabited" **and** type = "air"

Finally, we pose the query to determine all areas where people are endangered by pollution.

select **aggr_funion**(**fintersection**(zone, area, T_p)) **as** endangered_area
from pollution, land_use
where use = "inhabited" **and fisect**(zone, area)

6 Implementation

Due to our concept of executable specifications (see Section 3.2), the implementation of the Spatial Plateau Algebra only requires high level calls to an underlying crisp spatial type system implementation. This minimizes the needed implementation effort. Available crisp spatial algebra implementations are, for example, ESRI's Spatial Database Engine (ArcSDE), Oracle Spatial, Informix's Geodetic Data Blade, DB2's Spatial Extender, PostGIS, MySQL Spatial Support, the Java Topology Suite (JTS), and the Geometry Engine Open Source (GEOS) (see Section 2.3). However, the concepts, interfaces, functional range, and implementations of these spatial type systems as well as the storage techniques for spatial objects are rather different from each other and depend on the database system used. This means that different implementations of the Spatial Plateau Algebra would be required for different database systems with their specific spatial type systems. Some of the aforementioned spatial type systems additionally offer interfaces based on the *OpenGIS Simple Features Specifications For SQL* of the Open Geospatial Consortium (OGC). But even these interfaces are not generic and unique since they are only informally specified.

We have therefore decided to go another, longer way and aim at a novel and general solution to complex object management in databases that enables the *type system implementer* (*TSI*) to define, implement, and integrate new type systems for *complex application objects* into database systems on her own. These type systems and the whole framework that enables their implementation are database independent. That is, they are reusable and can be embedded into several database systems. The framework that enables their implementation is based on two pillars. The first pillar is a generalized method, named *type structure specification* (*TSS*), for representing and interpreting the structure of complex application objects. Such a specification provides an interface for the TSI to describe the structure of complex objects at a conceptual and high level of abstraction by means of special TSS grammars. In addition, it provides a generic interface for high-level retrieval, storage, and update operations on complex objects and their components.

The second pillar is a generalized framework, named *intelligent binary large objects* (*iBLOBs*), for the efficient and high-level storage, retrieval, and update of hierarchically structured complex objects in databases. iBLOBs store complex objects by utilizing the unstructured storage capabilities of a DBMS and provide component-wise access to them. In this sense, they serve as a communication bridge between the high-level abstract type system and the low-level binary storage. Low-level binary storage is usually provided by the built-in type for *binary large objects* (*BLOBs*). BLOBs themselves are not well suited for structured object management. They have originally been designed for storing unstructured data as byte

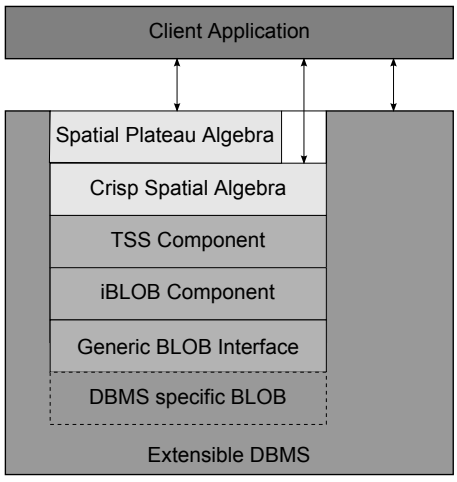


Figure 9: Integration of the Spatial Plateau Algebra into an extensible database system.

sequences and offer a low-level interface for simple read/write access to byte ranges. Thus BLOBs do not understand the semantics of the internal structure of the application objects stored in them and therefore do not include methods to access internal components of them. This makes the access to a component of an application object rather expensive since the entire object needs to be loaded into main memory to understand its structural semantics and get access to the component of interest. Further, BLOBs typically allow data to be appended, truncated, and modified through the overwriting of bytes. However, general data insertions and deletions are not supported unless the user explicitly shifts data, which is expensive. Since the implementations and access operations of the BLOB type vary largely between different database systems, we have designed a *Generic BLOB Interface* that has to be implemented for each database specific BLOB implementation.

Figure 9 illustrates the proposed system architecture. A more detailed description of the TSS concept and the iBLOB concept, which are beyond the scope of this article, can be found in [10]. This research effort is ongoing work. We are currently implementing the TSS component and the iBLOB component and are beginning with the TSS specification and implementation of a crisp spatial type system called *Spatial Algebra*. The implementation of the Spatial Plateau Algebra on top of the Spatial Algebra will follow these implementation efforts.

7 Conclusions and Future Work

In this article, we have dealt with the problem of implementing spatial objects and operations afflicted with the feature of spatial vagueness in a database context. *Spatial vagueness* or *spatial fuzziness* is inherent to many database applications in the geosciences and in geographical information systems. Our solution presented in this article is the Spatial Plateau Algebra that implements a fuzzy spatial algebra (like one of those described in [18, 42]) and can be embedded into any extensible and commercially or publicly available database system and its query language SQL. This algebra is a type system that provides the spatial plateau data types *fpoint*, *fline*, and *fregion* together with a comprehensive collection of spatial plateau operations. A special characteristic of our approach is that these data types and operations rest on well known concepts, data structures, algorithms, and implementations of their crisp counterparts. This leads to executable specifications that can be directly implemented with minimal effort. Further, spatial plateau data types are closed under spatial plateau operations. To the author's knowledge, the Spatial Plateau Algebra provides the first approach to an implementation of a fuzzy spatial algebra in general and in a database context in particular.

For future work, we plan to extend the Spatial Plateau Algebra by *metric operations* and *topological predicates on spatial plateau objects* as the implementation of their counterparts on fuzzy spatial objects.

References

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

[2] B. C. Arabacioglu. Using Fuzzy Inference System for Architectural Space Analysis. *Applied Soft Computing*, 10(3):926–937, 2010.

[3] M. de Berg, M. van Krefeld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition edition, 2000.

[4] D. G. Brown. Mapping Historical Forest Types in Baraga County Michigan, USA as Fuzzy Sets. *Plant Ecology*, 134:97–111, 1998.

[5] J. J. Buckley and E. Eslami. Fuzzy Plane Geometry I: Points and Lines. *Fuzzy Sets and Systems*, 86:179–187, 1997.

[6] J. J. Buckley and E. Eslami. Fuzzy Plane Geometry II: Circles and Polygons. *Fuzzy Sets and Systems*, 87:79–85, 1997.

[7] P. A. Burrough. *Natural Objects with Indeterminate Boundaries*, pp. 3–28. In Burrough and Frank [8], 1996.

[8] P. A. Burrough and A. U. Frank, editors. *Geographic Objects with Indeterminate Boundaries*. GISDATA Series, vol. 2,. Taylor & Francis, 1996.

[9] P. A. Burrough, P. F. M. van Gaans, and R. A. Macmillan. High-Resolution Landform Classification Using Fuzzy k-Means. *Fuzzy Sets and Systems*, 113:37–52, 2000.

[10] T. Chen, A. Khan, M. Schneider, and G. Viswanathan. iBLOB: Complex Object Management in Databases Through Intelligent Binary Large Objects. *3rd Int. Conf. on Objects and Databases*, pp. 85–99, 2010.

[11] T. Cheng, M. Molenaar, and H. Lin. Formalizing Fuzzy Objects from Uncertain Classification Results. *Int. Journal of Geographical Information Science*, 15:27–42, 2001.

[12] E. Clementini and P. Di Felice. A Model for Representing Topological Relationships between Complex Geometric Features in Spatial Databases. *Information Sciences*, 90(1-4):121–136, 1996.

[13] E. Clementini and P. Di Felice. A Spatial Model for Complex Objects with a Broad Boundary Supporting Queries on Uncertain Data. *Data & Knowledge Engineering*, 37:285–305, 2001.

[14] A. G. Cohn and N. M. Gotts. *The 'Egg-Yolk' Representation of Regions with Indeterminate Boundaries*, pp. 171–187. In Burrough and Frank [8], 1996.

[15] R. de Caluwe, G. de Tré, and G. Bordogna, editors. *Spatio-Temporal Databases–Flexible Querying and Reasoning*. Springer-Verlag, 2004.

[16] J. de Gruijter, D. Walvoort, and P. Vangaans. Continuous Soil Maps – A Fuzzy Set Approach to Bridge the Gap between Aggregation Levels of Process and Distribution Models. *Geoderma*, 77:169–195, 1997.

[17] G. de Tré, R. de Caluwe, and A. Hallez. *The Applicability of Generalized Constraints in Spatio-Temporal Database Modeling and Querying*, pp. 127–158. In de Caluwe et al. [15], 2004.

[18] A. Dilo, R. D. de By, and A. Stein. A System of Types and Operators for Handling Vague Spatial Objects. *Int. Journal of Geographical Information Science*, 21(4):397–426, 2007.

[19] S. Dragićević. *Fuzzy Sets for Representing the Spatial and Temporal Dimensions in GIS Databases*, pp. 11–27. In de Caluwe et al. [15], 2004.

[20] S. Du, Q. Qin, Q. Wang, and B. Li. Fuzzy Description of Topological Relations I: A Unified Fuzzy 9-Intersection Model. *1st Int. Conf. on Advances in Natural Computation*, LNCS 3612, pp. 1261–1273. Springer-Verlag, 2005.

[21] S. Dutta. Qualitative Spatial Reasoning: A Semi-Quantitative Approach Using Fuzzy Logic. *1st Int. Symp. on the Design and Implementation of Large Spatial Databases*, LNCS 409, pp. 345–364. Springer-Verlag, 1989.

[22] S. Dutta. Topological Constraints: A Representational Framework for Approximate Spatial and Temporal Reasoning. *2nd Int. Symp. on Advances in Spatial Databases*, LNCS 525, pp. 161–180. Springer-Verlag, 1991.

[23] G. Edwards. Characterizing and Maintaining Polygons with Fuzzy Boundaries in GIS. *6th Int. Symp. on Spatial Data Handling*, pp. 223–239, 1994.

[24] M. J. Egenhofer. Spatial SQL: A Query and Presentation Language. *IEEE Trans. on Knowledge and Data Engineering*, 6(1):86–94, 1994.

[25] ESRI Spatial Database Engine (SDE). Environmental Systems Research Institute, Inc., 1995.

[26] Geometry Engine Open Source (GEOS). `http://trac.osgeo.org/geos/wiki`.

[27] R. H. Güting. Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems. *Int. Conf. on Extending Database Technology*, pp. 506–527, 1988.

[28] H. Hendricks Franssen, A. van Eijnsbergen, and A. Stein. Use of Spatial Prediction Techniques and Fuzzy Classification for Mapping Soil Pollutants. *Geoderma*, 77:243–262, 1997.

[29] IBM. Informix Geodetic DataBlade Module: User's Guide, 2002.

[30] IBM. DB2 Spatial Extender and Geodetic Data Management Feature – User's Guide and Reference, 2006.

[31] Java Topology Suite (JTS). `http://www.vividsolutions.com/jts/JTSHome.htm`.

[32] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall, 1995.

[33] V. J. Kollias and A. Voliotis. Fuzzy Reasoning in the Development of Geographical Information Systems. *Int. Journal of Geographical Information Systems*, 5(2):209–223, 1991.

[34] P. Lagacherie, P. Andrieux, and R. Bouzigues. *Fuzziness and Uncertainty of Soil Boundaries: From Reality to Coding in GIS*, pp. 275–286. In Burrough and Frank [8], 1996.

[35] A. Morris and F. E. Petry. Design of Fuzzy Querying in Object-oriented Spatial Data and Geographic Information Systems. pp. 165–169, 1998.

[36] MySQL Spatial Support. `http://dev.mysql.com/doc/refman/5.0/en/gis-introduction.html`.

[37] Oracle Corporation. Oracle Spatial User's Guide and Reference 10g Release 2, 2005.

[38] A. Pauly and M. Schneider. VASA: An Algebra for Vague Spatial Data in Databases. *Information Systems*, 35(1):111–138, 2010.

[39] PostGIS. `http://postgis.refractions.net/`.

[40] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer Verlag, 1991.

[41] M. Schneider. *Spatial Data Types for Database Systems - Finite Resolution Geometry for Geographic Information Systems*, volume LNCS 1288. Springer-Verlag, 1997.

[42] M. Schneider. Uncertainty Management for Spatial Data in Databases: Fuzzy Spatial Data Types. *6th Int. Symp. on Advances in Spatial Databases*, LNCS 1651, pp. 330–351. Springer-Verlag, 1999.

[43] M. Schneider. Design and Implementation of Finite Resolution Crisp and Fuzzy Spatial Objects. *Data & Knowledge Engineering*, 44:81–108, 2003.

[44] M. Schneider and T. Behr. Topological Relationships between Complex Spatial Objects. *ACM Trans. on Database Systems*, 31(1):39–81, 2006.

[45] M. J. Somodevilla and F. E. Petry. *Fuzzy Minimum Bounding Rectangles*, pp. 237–263. In de Caluwe et al. [15], 2004.

[46] A. Sözer, A. Yazici, H. Oğuztüzün, and F. E. Petry. Querying Fuzzy Spatiotemporal Databases: Implementation Issues. In J. Kacprzyk, F. E. Petry, and A. Yazici, editors, *Uncertainty Approaches for Spatial Data Modeling and Processing – A Decision Support Perspective*, volume 271 of *Studies in Computational Intelligence*, pp. 97–116. Springer-Verlag, 2010.

[47] E. L. Usery. *A Conceptual Framework and Fuzzy Set Implementation for Geographic Features*, pp. 71–85. In Burrough and Frank [8], 1996.

[48] J. Verstraete, G. de Tré, R. de Caluwe, and A. Hallez. Field Based Methods for the Modeling of Fuzzy Spatial Data. In F. E. Petry, V. B. Robinson, and M. A. Cobb, editors, *Fuzzy Modeling with Spatial Information for Geographic Problems*, pp. 41–69. Springer-Verlag, 2005.

[49] J. Verstraete, A. Hallez, and G. de Tré. Bitmap Based Structures for the Modelling of Fuzzy Entities. *Control Cybernetics*, 35(1):147–164, 2006.

[50] F. Wang. Towards a Natural Language User Interface: An Approach of Fuzzy Query. *Int. Journal of Geographical Information Systems*, 8(2):143–162, 1994.

[51] F. Wang and G. B. Hall. Fuzzy Representation of Geographical Boundaries in GIS. *Int. Journal of Geographical Information Systems*, 10(5):573–590, 1996.

[52] F. Wang, G. B. Hall, and Subaryono. Fuzzy Information Representation and Processing in Conventional GIS Software: Database Design and Application. *Int. Journal of Geographical Information Systems*, 4(3):261–283, 1990.

[53] M.F. Worboys and P. Bofakos. A Canonical Model for a Class of Areal Spatial Objects. *3rd Int. Symp. on Advances in Spatial Databases*, LNCS 692, pp. 36–52. Springer-Verlag, 1993.

[54] A. Yazici and K. Akkaya. Conceptual Modeling of Geographic Information System Applications. In G. Bordogna and G. Pasi, editors, *Recent Issues on Fuzzy Databases*, volume 53 of *Studies in Fuzziness and Soft Computing*, pp. 129–151. Physica-Verlag, 2000.

[55] L. Ying-Ming and L. Mao-Khang. *Fuzzy Topology*. World Scientific, 1997.

[56] L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

[57] F. B. Zhan. Approximate Analysis of Binary Topological Relations between Geographic Regions with Indeterminate Boundaries. *Soft Computing*, 2:28–34, 1998.