Fuzzy Spatial Objects
An Algebra Implementation in SECONDO

Thomas Behr and Ralf Hartmut Güting
LG Datenbanksysteme für neue Anwendungen
Fachbereich Informatik, Fernuniversität Hagen
D-58084 Hagen, Germany

***Abstract*— This paper describes a data model for fuzzy spatial objects implemented as an algebra module in SECONDO. Furthermore, the display functionalities and their implementation strategies are also depicted.**

I. INTRODUCTION

Over the years, spatial data modeling has implicitly assumed that the extent and hence the borders of spatial phenomena are precisely determined, homogeneous, and universally recognized. From this perspective, spatial phenomena are typically represented by sharply described points, lines, and regions [1]. We will call this kind of entities crisp or determinate spatial objects.

Increasingly, however, researchers are realizing that the current mapping of spatial phenomena to crisp spatial objects is an insufficient abstraction of the real world, as the feature of spatial vagueness or spatial indeterminacy is inherent to many geographic data. Examples include the air pollution area caused by a power plant and the water pollution section of a river caused by a paper-making factory. We can observe that the pollution caused by the power plant covers a certain region. However, the pollution in the center of the region is more serious than that at the boundary. The same property is with the line entity corresponding to the polluted river. Points which have similar properties are also imaginable. We call this kind of entities fuzzy spatial objects.

For a determinate spatial object, we can use a boolean function to check whether a given point $p \in \mathbb{R}^2$ is contained in it. However, we cannot define such a "true or false" function for a fuzzy object. Instead, we should define a function to return a value $d \in [0, 1]$ which indicates to what extent the point belongs to the fuzzy object. We call the value d "degree of affinity", or "degree" for short. It is obvious that we can consider a determinate spatial object as a special case of fuzzy objects. A detailed discussion of fuzzy spatial objects can be found in [2].

II. PROGRAMMING ENVIRONMENT

The fuzzy spatial objects together with their operations are designed as an algebra module, called "FuzzyAlgebra", in SECONDO. SECONDO is a new generic environment supporting the implementation of database systems for a wide range of data models and query languages [3], [4]. It is developed as a research prototype at the Fernuniversität Hagen. The implementation of each algebra in SECONDO is based on the concept of *second-order signature* [5] with the first signature describing *type constructors* and the second signature describing *operations* on these type constructors. An algebra can be plugged into SECONDO with the central part of the SECONDO code unchanged. After recompiling SECONDO, we can use the newly added algebra. The type constructors of the FuzzyAlgebra are discussed in Section III and the operations are discussed in Section IV.

III. DATA TYPES

The FuzzyAlgebra contains three data types, `fpoint`, `fline`, and `fregion`, which correspond to their determinate counterparts. An `fpoint` value consists of a set of different positions in the plane with a degree of affinity in $(0, 1]$. An `fline` value is the union of a set of quasi disjoint curves in $\mathbb{R}^2$. The degree of the point set of this `fline` value can be zero only in a finite number of positions. The representation of the `fregion` value is based on the concept of the regular point sets. Such an area $R \subseteq \mathbb{R}^2$ is extended by an affinity function $f : R \to [0, 1]$, where only in single curves or in single points the value zero can arise.

The three defined types are extended by a scale factor $sf \in \mathbb{R}^+ \backslash \{0\}$. This is needed if we want to compare different objects which describe similar properties. For example, suppose r_1, r_2 are two `fregion` objects which describe the air pollution of the power stations S_1 and S_2 respectively. In the center (the position of the power station itself) the degree of both objects must be 1, because this point is 100% included in the area of air pollution. But the pollution of S_1 and S_2 can be different, e.g. because S_1 has much more output than S_2. When we ask for the air pollution of both stations together, we can use the scale factor to indicate this difference.

IV. OPERATIONS

The FuzzyAlgebra contains more than 30 operations. One reason is that many operations appear twice, once as "scaled" and once as "non-scaled". The non-scaled version ignores the scale factor described in Section III. This is needed to handle objects with different properties. As an example we can ask for the effects of the air pollution of the power station S to the nature reserve N, which is a non-fuzzy region. Here the scale factor makes no sense and hence it should be ignored.

In order to make it more clear, we list some typical operations special for the fuzzy spatial objects in the following.

The *alpha_cut* operator is a selection to the part of the fuzzy object with a degree greater than the given parameter alpha. The point set operators *union* and *intersection* (as well as their scaled versions) are based on the *max* and *min* operations for the degree.

The operator *add* is the contrary to the *difference* operator. Both use the corresponding mathematical operations + and - applied to the degree. The *add* operator is new for fuzzy objects meaning it does not exist for crisp objects. It makes it possible to compute the pollution of several power stations together.

A few operations compute some characteristics of a given fuzzy spatial object:

- *area*: the area of an fregion weighted by degree (volume)
- *basicarea*: the area of the covered territory of a fuzzy region
- *area3d*: the area of the surface of a fuzzy region considered as an object in three dimensions
- *length*, *basiclength*: the weighted (unweighted) length of a fuzzy line
- *length3d*: the length of the 3D structure for a fuzzy line
- *cardinality*, *basiccard*: the weighted (unweighted) number of positions contained in a fuzzy point

boundary and *contour* compute the boundary of a fuzzy region, where the operator *contour* ignores the holes within this region.

We also proposed an operator *value_at* to replace the classical operator *contains*, which returns the degree for a given position. On the basis of the current implementation (see Section V), this operator is not implementable. To solve this problem, we have implemented three operators *max_value_at*, *min_value_at* and *mid_value_at*.

Other operations are: *similar*, *basic_similar*, *common_lines*, *common_points* and *holes*.

V. Concepts of the Implementation

The goal of the implementation was to build a general model for fuzzy spatial objects. It should enable the representation of complex `fpoint`-, `fline`-, and `fregion`-objects, which means:

- all objects can be empty
- an instance of `fpoint` can contain zero, one or more positions (a finite number)
- an `fline` object can have an arbitrary finite number of endpoints and can contain separated components
- an `fregion` value can consist of several non-connected faces with possible holes
- the function describing the degree should be piecewise continuous

To implement this, we have proposed a method to deassemble the $\mathbb{R}^2$ in a raster composed of a set of homogeneous triangles. In [6] a rectangular partition of the Euclidean plane is used where each rectangle refers to a single degree. This does not allow continuous functions of the degree. Here, we use the sides and corner points of the triangles as well as the triangles themselves as the basic elements. Furthermore, the basic elements together with a degree for each basic point are the building blocks for fuzzy spatial objects. A fuzzy spatial object consists of a set of building blocks defined here, where the corresponding basic sets are disjoint. Additionally such an object contains the scale factor described in Section III. We can model discontinuities referring to the same position of two different neighbouring basic elements to a different degree. This explains the necessity to have different operators returning the value of degree for a given position.

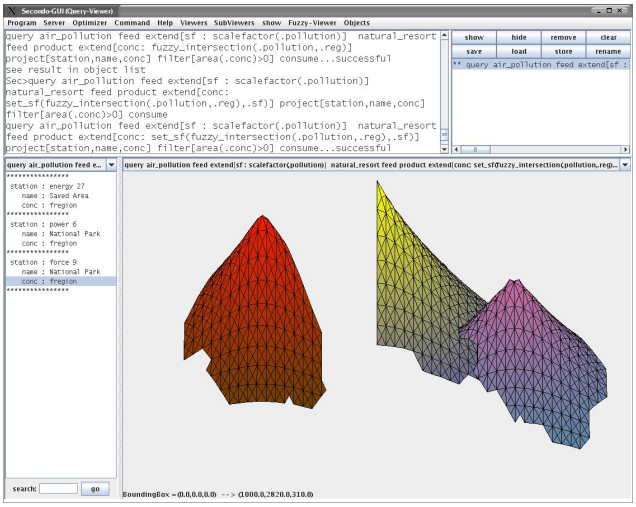

Fig. 1. Natural resorts concerned by air pollution

VI. Graphical Representation

A fuzzy object cannot be drawn with the typical methods for spatial objects. The reason is that these methods cannot depict the degree information of the fuzzy object. There are essentially two possibilities to draw a fuzzy spatial object. The first one is to colorize the object with a linear distribution of colors. For segments (triangles) we can use a single color given by the middle degree value of the two (three) end- (corner-) points. We can also colorize the points in the interior of such a building block. The second possibility is to draw the fuzzy objects in three dimensions with the value $degree * scalefactor$ used as the third dimension. In this case we can also use the coloring method in addition.

For the graphical representation of the objects of the FuzzyAlgebra we use the interface "Javagui" of Secondo. This interface uses viewers to display objects. The objects are not restricted to be graphical. There are a lot of viewers to display different object types. A new viewer can be plugged in at runtime without changing any code of this framework. We have implemented two viewers for drawing fuzzy spatial objects corresponding to the two possibilities described above. In both viewers an object can be colored continuously by defining the colors for degree 0 and for degree 1. By assigning different colors to diffent objects we can make them distinguishable. In the three dimensional representation, the position of the eye, the view reference point as well as the desired window are changeable.

Compared with 3D pictures, the 2D pictures are more exact. There are two reasons for this. First, in the 3D view there exists a distortion which comes from the projection from 3D space into the 2D plane. The second reason is that the painter's algorithm used in the 3D viewer is not as exact as the z-buffer algorithm used in the 2D viewer. This means that some part of the object can be incorrectly covered, for instance, when two triangles intersect. The disadvantage of the 2D viewer is that the parts of a fuzzy object covered by another one cannot be displayed. In the 3D viewer we can look from the bottom to view the covered parts. With the help of another viewer, "QueryViewer", it is possible to view relations containing fuzzy objects. This viewer can display relations containing arbitrary objects and uses a sub-viewer to display those objects.

VII. Example Scenario

To demonstrate the functionalities of the FuzzyAlgebra we have constructed a little scenario. In this example we have only included fuzzy regions because we think that this type is the most important one. We have several relations with the following schemas:

`air_pollution`

no: int	station: string	output: real	pollution: fregion

`territory`

owner: string	territory: fregion

`natural_resort`

name: string	reg: fregion

`living_room`

species: string	reg: fregion

The `fregion` attributes for the relations `territory` and `natural_resort` are determinate. The scale factor here makes no sense and holds the value 1.0.

With these relations we can execute some queries. In the sequel all queries are given as query plans at the executable level of SECONDO. That is, all used operations are given directly, and most of them are in postfix notation. Aided by the optimizer of SECONDO we could also use an SQL-like syntax, but this is not needed to demontrate the facilities of the FuzzyAlgebra.

The following query can be used to compare the pollutions caused by the related power plants. Besides we compute the rate between pollutions:

```
query air_pollution feed
    extend[volume : area(.pollution)]
    project[station, output, volume]
    extend[rate : .volume / .output]
    sortby[rate asc]
consume
```

The *feed* operator takes the relation `air_pollution` and converts it to a stream of tuples. With the *extend* operator we can add a new attribute into the tuples. The functionalities of *project* and *sortby* are self-explanatory from their names. The operator *consume* collects all tuples of the input stream into a relation which is the the result of the query.

The next query computes the air polluted area affected by a power plant:

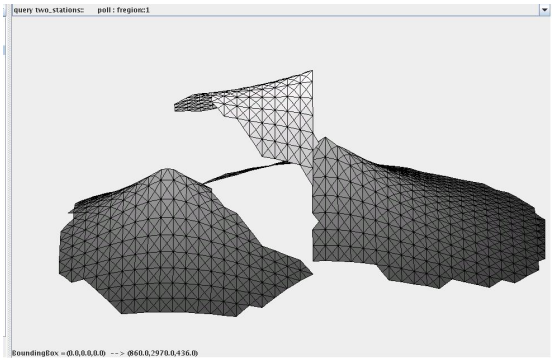

Fig. 2. The air pollution of two stations together

```
query air_pollution feed
 extend[sf : scalefactor(.pollution)]
 natural_resort feed product
 extend[conc: set_sf(
       fuzzy_intersection(.pollution,.reg),.sf)]
 filter[area(.conc)>0] project[station,name,conc]
consume
```

The result of this query is shown in Figure 1. In this figure there are three fuzzy regions. The query is processed like this. First the scale factor of the fuzzy regions is added as an attribute to the relation since we want to compare the pollutions. After that the `air_pollution` relation is combined with the natural resorts and the intersection of the fuzzy regions is computed. From the resulting stream all tuples are excluded where the area of the intersection has no extension. Because we do not need the whole information, we project to the attributes of interest.

We can also compute the air pollution caused by the power stations "energy 27" and "power 6" together. Because we want to reuse the result, we store it as a new object.

```
let two_stations = air_pollution feed
  filter[.station="energy 27"]
  air_pollution feed filter[.station="power 6"] {r}
  product
  extend[poll : scaled_add(.pollution,.pollution_r)]
  project[station,station_r,poll]
consume
```

The result `query two_stations` (see Figure 2) shows that there is an area where the sum of the pollutions is greater than the highest pollution of each single power station.

We can define a "dangerous" area as a region where the pollution has a value greater than 250. We can compute such a region by the following query:

```
query two_stations feed
  extend[sf : scalefactor(.poll)]
  extend[danger :
         sharp(alphacut(.poll,250.0/.sf,TRUE))]
extract[danger]
```

The result is the region at the upper part of Figure 2 but now as a crisp region. This means that every point contained in this region has the degree 1.0. We can use this result to find the plot of land from Mr. Smith, which can be used agrarian. This is a simple *difference* between the involved regions and omitted here.

When Mr. Smith wants to fence his land, he can compute how much fence is needed to do that. The used query is:

```
query territory feed
  filter[.owner="Smith"]
  extend[L: length(boundary(.reg))]
sum[L]
```

The answer will be a real number. Note that the used operations can also be defined for crisp spatial objects. Normally we should take the *basic_length* operator for this computation, but because the *boundary* operator returns a crisp object, the result will be the same.

Furthermore we can ask where doves are threatened by hawks. This is a simple intersection of the living areas of both species:

```
query scaled_intersection(
  living_room feed filter[.species="Hawk"]
  extract[room] ,
  living_room feed filter[.species="Dove"]
  extract[room]
)
```

VIII. What Will be Demonstrated

All the capabilities of the FuzzyAlgebra described above, as well as the related functionalities of the SECONDO system and the graphical user interface, can be demonstrated, for example:

- queries including operators and objects of the FuzzyAlgebra
- queries on relations containing fuzzy objects
- displaying fuzzy spatial objects in two dimensions
- showing fuzzy spatial objects in a three dimensional scene
  - colorizing objects for distinguishing them
  - changing the view

References

[1] M. Schneider, "Uncertainty management for spatial data in databases: Fuzzy spatial data types," in *6th Int.Symp. on Advances in Spatial Databases (SSD)*, ser. LNCS 1651. Springer Verlag, 1999, pp. 330–351.

[2] T. Behr, "Modellierung, Implementierung und Visualisierung unscharfer räumlicher Objekte," Master's thesis, Fernuniversität Hagen, 2002.

[3] S. Dieker and R. H. Güting, "Plug and play with query algebras: SECONDO-a generic DBMS development environment," in *International Database Engineering and Application Symposium*, 2000, pp. 380–392.

[4] R. H. Güting, T. Behr, V. Almeida, Z. Ding, F. Hoffmann, and M. Spiekermann, "Secondo: An extensible dbms architecture and prototype," FernUniversität in Hagen, Tech. Rep., 2004.

[5] R. H. Güting, "Second-order signature: A tool for specifying data models, query processing, and optimization," in *SIGMOD*, 1993.

[6] D. Altman, "Fuzzy Set Theoretic Approaches for Handling Imprecision in Spatial Analysis," *International Journal of Geographical Information Systems*, vol. 8, no. 3, pp. 271–289, 1994.