GEOSQL: A SKETCH-BASED QUERY LANGUAGE FOR GEOGRAPHICAL DATA

Fernando Ferri, Patrizia Grifoni, Maurizio Rafanelli

Istituto di Ricerche sulla Popolazione e Politiche Sociali, CNR, Via Nizza 128, 00198 Roma
[f.ferri, p.grifoni] @irpps.cnr.it

Istituto di Analisi dei Sistemi ed Informatica, CNR, Viale Manzoni 30, 00185 Roma
rafanelli @iasi.cnr.it

Abstract

Drawing is a simple, common, and expressive way to represent concepts and relationships for geographical databases. However, the possibility to revise the sketch and multiple interpretations of different shapes and their relationships in the sketch makes very difficult to determine the correct interpretation of the user goals. This paper starts from the assumption that the sketched picture is given by an array of pixels and the primitive shapes are represented by black pixels on a white background. It treats ambiguities and their solutions choosing the most correct interpretation for the shapes of the sketch and for their relationships. The correct interpretation is obtained considering both the sketch and temporal information on the sketch evolution.

1. Introduction

A lot of proposals of spatial query languages, and more particularly, of geographical query languages have been advanced during the last ten years. They can be classified according to two different approaches: i) textual (natural language and SQL extension), ii) no textual (tabular, graphic and visual, and more specifically iconic and pictorial languages, etc.). The proposals of the textual approach mainly concern SQL language extensions that enable DBMS to load and retrieve spatial information. Besides the different SQL dialects, some graphic languages have been defined. They cannot be considered as visual languages because they use a graphic approach in order to provide only more powerful functionalities [1], but not to provide a visual representation of the result.

Among the reasons which determined the arising of the first visual and iconic languages [2][3][4], there was the need to provide such query languages with the same dimension of the considered word. Indeed the use of textual languages to express spatial concepts forces a projection from two or more dimensional words to the space of a linear language. Geographical data are intrinsically spatial data and it is very useful to consider them in terms of explicit spatial concepts instead of expressing a query with a lexical approach.
Among the visual query languages for geographical data the sketch-based languages are very interesting. Sketching is an interaction modality, which directly supports the way to express spatial information (generally people use an image to represent spatial configurations). Moreover, using sketching, a graphic feedback of the spatial configuration is immediately produced. Sketching rapidly communicates ideas through approximate visual pictures with low overhead (pencil and paper), easy to correct and revise without necessity of specialized knowledge or precision. When a user draws a visual image, he tends to divide the graphic objects into several primitive shapes. Primitive shapes are basic elements of diagrams, graphs or complex objects. They usually are closed-shape or polylines. Each primitive shape is drawn by means of a sequence of strokes. A stroke is a trajectory of the pen (or mouse) movement between the time in which the pen-tip begins to touch the tablet and the time in which the pen-tip has lifted up from the tablet.

However, using a sketching technique in a query expression, if only configurations perfectly corresponding to the sketch have to be returned, it could happen that no situation is exactly correspondent. Indeed this technique does not permit to easily represent more alternative (OR) spatial relations (they can require to be represented among primitive shapes).

Both the previous problems are faced providing not only results perfectly corresponding to the query, but also results obtained reducing the number of constraints in order to obtain results approximating the sketch. Such a language is, for example, Spatial-Query-by-Sketch [5], where it considers a computational model of similarity for spatial relations, in order to decide the constraints that can be maintained, deleted or modified in that language.

In sketch based interfaces several and different needs of recognizing and ambiguity management can be considered. The sketch-based geographical query languages usually manage only few types of graphical objects and their spatial relationships. In sketch-based geographical query languages the correct recognition process for the different shapes of the sketch requires to take into account both: i) the sketch, ii) as the sketch was designed. It is a very complex activity to identify the correct interpretation of the sketch according to the possibility of modifying the sketch deleting, adding and modifying each its shape. More complex is to give a description using sketches of objects manipulation by means of complex actions.

2. Background

The main characteristic of the geographical information is that data of the database can be distinguished between geographical data describing spatial properties of geographical information and no graphical data (usually referred as alphanumerical) describing characteristics and properties of the geographical information. Moreover abstract data types, such as point, polyline and region, provide an abstraction to model geometric entities in the space, but also to model relations (i.e. inclusion), properties (i.e. area and length) and operator (selection of the area that is intersection of two regions). A point is an abstraction of objects only characterized by its position in the space. A polyline is an abstraction of objects characterized by path and connections (i.e. rivers, roads, …). Finally, a region is an abstraction of objects extended in a bi-dimensional space.

Geographical information systems represent both geometric characteristics of the geographical entities, and spatial relations among these entities. Among the different kinds of spatial relationships introduced in literature, topologic relationships are the most significant. In particular GeoSQL [4] considers the following operators:

- Geo-Disjunction (Dsj): Two geographical entities $\psi_i$ and $\psi_j$ are disjointed if the intersection of their boundaries and the intersection of their interiors side are null.
- Geo-Touching (Tch): The Geo-Touching between two geographical entities $\psi_i$ and $\psi_j$ exists iff the points common to the two $\psi$ are all contained in the union of their boundaries. If this
condition is satisfied, the result of this operation between \( \psi_i \) and \( \psi_j \) is a new geographical entity \( \psi_h \) defined by the set of points common to \( \psi_i \) and \( \psi_j \).

- **Geo-Inclusion (Inc):** A geographical entity \( \psi_i \) includes another geographical entity \( \psi_j \) iff all the points of \( \psi_j \) are also points of \( \psi_i \). The result is a geographical entity \( \psi_h \) which coincides with \( \psi_j \).

- **Geo-Crossing (Crs):** If the geographical entity polyline \( \psi_i \) crosses another geographical entity polyline \( \psi_j \), the intersection of their interior sides will be a point.

- **Geo-Pass-through (Pth):** The Geo-Pass-through operator can be applied only between a polyline and a region. The result is the geographical entity polyline which is internal to the region. If the operator acts on a polyline \( \psi_i \) and a region \( \psi_j \), then \( \psi_i \) Geo-Pass-through \( \psi_j \) iff the polyline is partially included in the geo-region.

- **Geo-Overlapping (Ovl):** Two geographical entities \( \psi_i \) and \( \psi_j \) of the same type have a non-null overlap if both of them, as well their intersection, have the same dimension.

### 3. Ambiguity and Multiple Interpretation

Sketches drawn by the user can have multiple interpretations [6] [7]. This one occurs because, on one hand, a unique space is used to express different kinds of information and, on the other hand, signs on the sketch may not completely represent the semantic of the information related to the sign. In Figure 1A an example of ambiguity is presented. The sketch can have at least the interpretations shown in Figure 1B and Figure 1C. Obviously, if the goal of the user is to sketch a river passing-through a region, the correct interpretation is represented in Figure 1C.

![Fig. 1. A sketch and some of the possible interpretations](image)

It is possible to resolve ambiguities contained in the sketch using different approaches (eventually integrated):
- specifying the correct interpretation by the user;
- considering the application domains and a significant set of sketch models for each domain;
- considering both the sketch and the actions made to draw the sketch in order to have the correct interpretation.

Usually, a sketch is drawn by several steps in which the sketch can be updated with new primitive shapes, or primitive shapes can be modified, partially deleted or completely deleted. It is not possible to define a predefined limit to the number of modifications of the sketch. In order to avoid the management of a very complex model taking into account all the successive modifications in the sketch, according to the interpretation of the user action on the sketch, the adopted approach analyses the sketch after the user finished it. Firstly the interpretation of the
sketch considers the closed-shapes, and successively it considers the polylines (obtained from the sketches after eliminating closed-shapes).

A finest interpretation is obtained considering temporal information related to the primitive shapes of the sketch. Finally, the user can directly correct the eventually wrong interpretations remaining from the previous two steps. Indeed, any criterion chosen to interpret correctly the sketch has necessarily to be based on usage and common sense rules, not necessarily respected by all the users during the interaction with the sketch-based interface (characterized by lack of constrains for the user).

Example: The considered sketch is in Figure 2

![Fig. 2. An example of sketch](image)

Its correct interpretation is: A touch B; C passthrough A; C disjoint B

A rule generally adopted by users is that they sketch each primitive graphical objects minimizing strokes and drawing consecutive strokes (i.e. they cannot be succeeded each other with strokes referring other primitive graphical objects).

According to this criterion the user, in order to have a correct interpretation has, for example, to produce Figure 2, according to the actions sequence of Figure 3.

![Fig. 3. The sequence of actions for the sketch of Figure 4](image)

However, a sketch-based interface cannot constrain the user to adopt the previous sequence to draw the sketch. Therefore, if the user defines the sketch of Figure 2 according to different action sequence presented in Figure 4, a different interpretation corresponds to the same sketch.

![Fig. 4. Another sequence of actions for the sketch of Figure 2](image)
4. The algorithm for vectorizing and recognizing primitive shapes

Three different steps form the algorithm devoted to vectorize and to recognize the primitive shapes of the sketch [8]. A vectorial representation describes sketches in terms of primitive shapes (with their geometry). The transformation must preserve topological features and relationships between initial primitive shapes, i.e. preservation of inclusion, touching, and other relationships.

The algorithm assumes that the input image is given by an array of black and white pixels and uses a representation based on coding of horizontal sequences of black pixels identified in the sketch. Horizontal sequences of black pixels are stored in a record that contains the following fields:
- vv contains the row index,
- valInf contains the index of the column of the pixel which starts the horizontal sequence,
- valSup contains the index of the column of the pixel which ends the horizontal sequence.

Using this kind of representation, the part of a sketch formed by three horizontal sequences of black pixels, shown in Figure 5, could be stored using the following three records:

sequence1.vv:=14; sequence2.vv:=15; sequence3. vv:=15;
sequence1.valInf:=11; sequence2. valInf:=9; sequence3. valInf:=15;
sequence1. valSup:=14; sequence2. valSup:=10; sequence3. valSup:=16;

The first step identifies and stores in a list all the horizontal sequences of black pixels of the sketch according to the order of the pixels in the scanning starting from the top left hand corner, and ending to the bottom right hand corner.

The second step identifies the different shapes of the sketch. The identification of the shapes maximizes the number of closed regions and minimize their area. Moreover, all the remaining not closed shapes are identified as polylines. For example, in the sketch of Figure 4C, which represents the query formulated by the user, the second step of the algorithm recognizes closed-shapes (A, B and C) and polylines (1 and 2) as represented in Figure 6A.

Fig. 6. Closed-shapes s and polylines recognized by the algorithm
Then, after the second step, the query is characterized by the following relationships:

a) A touch B  
b) B touch C  
c) A disjoint C  
d) 1 touch A  
e) 1 disjoint B  
f) 1 disjoint C  
g) 2 touch B  
h) 2 disjoint A  
i) 2 disjoint C

The third step of the algorithm considers the different actions of the user in designing the sketch. The user draws the sketch by means of a set of strokes, each stroke is drawn in a temporal interval. One or more strokes (or parts of strokes) form a primitive shape. For example, the primitive shape of Figure 7C is formed by the left stroke of Figure 7B, and the primitive shape of Figure 7D is formed by a part of the left stroke of Figure 7B and by the entire right stroke of Figure 7B. Each primitive shape can be characterized, from the temporal point of view, by means of a set of time intervals: the intervals in which the strokes (or their parts) forming the shape are drawn. For example, in the sketch of Figure 7A drawn by means of two strokes represented in Figure 7B, the second step of the algorithm recognizes two closed-shapes (Figure 7A), each one characterized by means of the intervals of time of Figure 7C and Figure 7D. In fact, the closed shape of Figure 7C involves just one temporal interval (a), on the contrary, the closed shape of Figure 7D involves two temporal intervals (a’ and b, where a’ is a sub-interval of a), because the two branches composing the closed shape belong to two different strokes drawn by the user.

The use of the temporal information can be very useful in order to identify the user goals. In fact, people tend to draw all of a primitive shape before moving to a new shape and tend to draw all of a graphic object before moving to a new object.

---

This important criterion can be used in two different manners in order to approach the correct interpretation of the sketch:

- minimizing the total amount of breaks between intervals for all primitive shapes; it means that each primitive shape is drawn as a stroke (preferably) without interruption.
- minimizing the total amount of waiting time (of breaks) between intervals for all the primitive shapes; it means that each serie of strokes refers to only one primitive shape. This is very important if the user prefers to draw a primitive shape by means of a series of strokes. Moreover this minimization can be applied for recognizing complex objects.

In order to minimize the total amount of breaks between intervals, it is necessary to reconsider primitive shapes with consecutive intervals. For example the sketch drawn in Figure 8A (by means of the two strokes of Figure 8B) is interpreted during the second step, as shown in Figure 8C. The two primitive shapes can be characterized from the temporal point of view by means of the temporal intervals of Figure 8C. Their couples of intervals (a-b and b-c) are consecutive, then they can be useful to reconsider the interpretation. In fact, in the example of Figure 8B the total amount of breaks is three (two for the primitive shape 1 and one for the primitive shape 2). Considering the union of the two shapes, it is possible to minimize the total amount of breaks that can be reduced to zero by means of a different (and probably more correct) interpretation of Figure 8D. On the contrary, in the example of Figure 7A the total amount of breaks is one (there is just one break between intervals of the primitive shape 2) and it is not possible to reduce the amount of breaks. In fact, in this case the most correct interpretation is that one found after the second step of the algorithm and represented in Figure 7B.

In order to minimize the total amount of waiting time between intervals for all primitive shapes, it is necessary to reconsider how series of strokes match with primitive shapes. When there is that series of strokes regarding two primitive shapes succeed each other, it is necessary to reconsider their interpretation in an analogous manner to the minimization of the total amount of breaks.

![Diagram](image-url)

Fig. 8. Another example of temporal information related to the primitive shapes of the sketch

However one of the main problem to consider in a sketch interpretation is that the sketch is not drawn linearly but the user can delete partially or totally drawn shapes or can modify one or more shape. The approach presented in this paper makes very easy to manage this kind of problem. In fact, deletions do not have impact on two minimization rules and then on the correct interpretation of the sketch, then it is not necessary to consider them. The interpretation that minimizes both the
total amount of breaks and the total amount of waiting time (of breaks) between intervals is the same considering or not deletions. Experimentally this interpretation is the most correct. Regarding modifications of shapes, a problem can occur when a modification concerns two or more shapes of the sketch. In fact, in this case the shape temporally nearest is chosen independently from the correct interpretation. However, experimentally, the given interpretation has the higher probability to be the correct interpretation. After the third step the query is characterized by the following relationships, as represented in Figure 6B:

a) Z touch Y
b) W pass-trough Z
c) W disjoint Y

5. Application of the algorithm to a sketch

The algorithm works following paths on the sketch. When the followed path is closed a closed-shape has been found, when the followed path is not closed one or more polylines have been found. Figure 9 shows the application of the algorithm to the sketch of Figure 1A. A row of Figure 9 refers to a different path and the last figure of a row shows the labels of the new shapes found by the algorithm at the end of the path. In particular, the first row considers the three edges constituting the shape A (Figure 9.1-A, Figure 9.1-B, Figure 9.1-C).

The second row shows a path compound of five edges from Figure 9.2-A to Figure 9.2-E. Because the path does not close, two polylines are detected: shapes 1 and 2 of Figure 9.2-E. The third row shows a path compound of four edges from Figure 9.3-A to Figure 9.3-D. The first edge is the border between shape A and shape B. This closed path allows to detect the shape B. Finally, in the fourth row, the path is constituted of two edges. The first edge is the border between the shape B and the shape C, the second edge is considered in Figure 9.4-B. The third step of the algorithm can perform some simple operations on the detected primitive shapes in order to improve the probability to obtain the correct interpretation. Then, for example, if the correct interpretation of the sketch of figure 1A is Figure 1C and the user drew uninterruptedly
the three primitive shapes, the third step of the algorithm modifies the interpretation of the sketch by means two steps shown in figure 10B and figure 10C.

![Fig. 10. The third step of the algorithm](image)

The first step executes the union of shapes A and B of figure 10A obtaining a new shape: the closed-shape D. Moreover the border between A and B has to be considered as a further new shape: the polyline 3. The second step executes the union of shapes 1, 2 and 3 obtaining only one shape: the polyline 4.

6. Conclusion

The proposed language allows to the user to formulate a geographical query by means of a sketch. The query is interpreted in three steps. In particular, the second step of the algorithm recognizes primitive shapes of the sketch (regions and polylines). In the third step, the algorithm resolves the problems related to multiple interpretations of the sketch and its primitive shapes. In order to improve automatic capabilities to recognize the correct interpretation of the sketch, this step considers the sketch drawing process, described in terms of the starting sketch, and the temporal intervals of all primitive shapes.

References