Topological Spatial Relations and Operators

Zhexue Huang

Environmental and Natural Resources Information Systems
Department of Photogrammetry
Royal Institute of Technology
100 44 Stockholm, Sweden
huang@fmi.kth.se

Abstract:

There is a growing interest in investigating the underlying topological relations of spatial objects, and defining corresponding operators to check these relations. The investigation has recently resulted in a complete set of eight different spatial relationships between two regions. To check these eight spatial relations between two polygons, six fundamental topological spatial operators are required. More operators for checking specific spatial relations between two objects can be created by combining the fundamental operators with other operators. These operators provide a sound basis for designing a spatial query language and the query language with implementation of such spatial operators can become a useful tool for spatial querying and analysis.

KEY WORDS: GIS, Spatial, Theory.

1. Introduction

A geographical information system (GIS) can be considered as a spatial database system on which a set of application programs operates. These application programs provide tools for the GIS user to make spatial queries and analyses about objects in the database. The object-oriented (OO) approach has proved to be a powerful tool for designing spatial databases, especially heterogeneous spatial databases (Oosterom et al 1989, Kemp 1990).

An OO spatial database consists of a collection of spatial objects, which, together with their (spatial) relations, represent spatial information about reality (Molenaar 1991). An important feature of an OO database system is that each object has an identity which allows the user to distinguish and address it (Unland et al 1990). Besides its identity a spatial object also carries two kinds of data: spatial data, which describe the location and geometry of the object in space, and attribute data, which represent non-spatial properties of the object.

Spatial relations of objects are an important part of the spatial information. The complexity of spatial relations among objects creates difficulties to explicitly represent all kinds of spatial relations in a database. An alternative is to define spatial functions in the database query language to discover spatial relations of objects.

There is a growing interest in investigating the underlying topological relations of spatial objects (Egenhofer et al 1990, 1991, Kainz 1989, 1990), and defining corresponding operators to check these relations (Svensson et al 1991). The investigation of topological spatial relations of objects in a topological space has recently resulted in a complete set of eight different spatial relationships between two regions (Egenhofer et al 1990, 1991). To check these eight spatial relations between two polygons, six fundamental topological spatial operators are required. The terminology used for polygons can also be adopted for defining spatial operators for points and lines and combinations of these. These operators provide a sound basis for designing a spatial query language and the query language with implementation of such spatial operators can become a useful tool for spatial querying and analysis (Svensson et al 1991).

This paper is organized as follows. Section 2 discusses types of spatial objects. Spatial data types are proposed for handling different types of objects in databases. Functions operating on the spatial data types are given in section 3. Section 4 discusses topological spatial relations between two objects. How to use given spatial functions and operators to define new operators to detect detailed spatial relations is demonstrated in section 5. Section 6 gives some examples showing the use of these operators in querying a spatial database. Some conclusions are drawn in section 7.

2. Object types and spatial data types

Any object that is related to a location in space is said to be spatial. Some spatial objects are complex, for example, a road network. Others are simple, for example, oil wells which are usually presented as points in maps. The complex objects are composed of simple objects.

Three basic types of spatial objects exist in the two dimensional space. They are points, lines, areas, which will be called polygons from now, and each of these three basic types has a primitive form. Any point is primitive. A line is primitive if it has no loop between its two ends, excluding the simply closed line whose two ends coincide. A polygon is

primitive if it is simple (Preparata et al 1985). A primitive polygon may contain holes which are also simple polygons. Examples of primitive and non-primitive lines and polygons are shown in Figure 1.

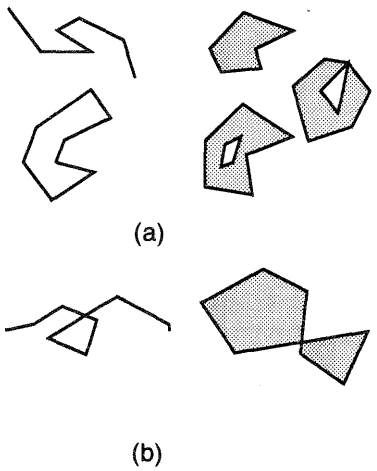


Figure 1. (a) Primitive lines and polygons.
(b) Non-primitive lines and polygons.

Non-primitive objects can be decomposed into a set of primitive objects.

To efficiently handle spatial objects in spatial databases, spatial data types are used. A spatial data type can be viewed as a data structure for storing spatial data of objects. Each instance of the spatial data types is a specific object which is uniquely identified by its identifier. Operators upon the data types are defined to alter or retrieve some information from the structure. Spatial data types can be either system-defined or user-defined.

The semantics of spatial data types can be described using set and binary relation concepts (Stanat et al 1977). The structure of the data type POINT is a pair $< p,q >$, where p and q are real numbers representing coordinates.

The data type POINTSET is defined as a countable set of points $(p_0,p_1,...p_{n-1})$.

The data type LINESEGMENT is a pair $<p_s,p_e>$, where p_s, p_e represent the start and end points of a line segment.

The data type LINE is defined as

$$< POINTSET, R >$$

where R is a binary relation on POINTSET representing a set of line segments connected to form a primitive line.

The data type LINESET is a set of lines ($l_0, l_1,...,l_{k-1}$).

The data type POLYGON is defined by

$$<l_o, L_h >$$

where l_o is a simply closed line representing the outer boundary and L_h is a set of simply closed lines representing holes of a primitive polygon.

Spatial data types can be used in a way similar to other data types such as INTEGER, REAL, TEXT. For example, relations can be defined by spatial data types in an extended relational database management system (ERDBMS) (Huang et al 1992). These relations become spatial because they contain spatial objects. Since each object in the database is uniquely identified, columns defined by spatial data types can be used as a key to the relations.

Generally speaking, spatial objects are stored independently in tuples in the spatial relations. Relationships between objects are not explicitly described. Spatial relationships, however, are detected by spatial operators defined in the spatial query language.

Some of the fundamental functions and operators needed in the spatial query language will be discussed in the following sections.

3. Functions on spatial data types

Based on the structures of spatial data types, a number of necessary functions can be defined (Svensson et al 1991). Some extract subsets of data or components of objects, such as extracting coordinates of a point or the outer boundary of a polygon. Some compute new data from the existing data set of an object instance, such as the length of a line. This section introduces some functions which are used in the following discussions.

XCOORD(p) and YCOORD(p) are functions to extract values of x,y coordinates of point p. p can be a specific identifier or the name of a relation column defined by the data type POINT.

SP(l) and EP(l) return the start and end points of line l.

LENGTH(l) returns a real number representing the length of line l.

AREA(pg) returns a real number representing the area of polygon pg.

BOUNDARY(pg) returns a closed line representing the boundary of polygon pg.

New functions can be built from compositions of the functions or combinations of the functions by logical operators.

For example, the function

PERIMETER (pg) =
LENGTH(BOUNDARY(pg))

computes the perimeter of polygon *pg* and the function

CLOSED(*l*) =
(XCOORD(SP(*l*)) = XCOORD(EP(*l*)))
AND
(YCOORD(SP(*l*)) = YCOORD(EP(*l*)))

returns the logical value TRUE if and only if line *l* is a closed line.

4. Topological spatial relations

Topological relations are such spatial relations that are invariant under topological transformations between two topologically equivalent spaces (Armstrong 1979). Adjacency, overlapping, and containment are typical examples. It is known that totally eight different topological relations exist between two regions (Egenhofer et al 1990).

By disregarding orders of containment and coverage and taking point and line objects into account, six topological relations between two spatial objects can be named (Svensson et al 1991). Table 1 presents the topological relations with different combinations of object types.

In some combinations of object types, one topological relation may be implied by different names. For example, MEETS, EQUALS, COVERS, OVERLAPS indicate the same topological relation of two points. In order to avoid ambiguity, only EQUALS is defined.

Table 1. Topological relations between objects of different types

topological relations	combination of object types					
	P-P	P-L	P-Pg	L-L	L-Pg	Pg-Pg
DISJOINT	Y	Y	Y	Y	Y	Y
MEETS	-	-	-	Y	Y	Y
EQUALS	Y	-	-	Y	-	Y
CONTAINS	-	-	Y	-	Y	Y
COVERS	-	Y	Y	Y	Y	Y
OVERLAPS	-	-	-	Y	Y	Y

Y/- means the topological relationship exists/undefined.

Visualization of topological relationships between two polygons is given in (Egenhofer et al 1990).

5. Topological operators

Spatial relations vary in the same topological relations. For example, when two polygons meet, they may meet at boundaries (meets-1), or they may meet at a corner (meets-0). It is necessary to specify the cases because the merging of two primitive polygons which meet at boundaries produces a primitive polygon, whereas the merging of two primitive polygons which meet at a corner results in a non-primitive polygon.

Spatial operators are needed to detect topological spatial relations of two objects. In order to define spatial operators, specific cases of spatial relations should be investigated. Some operators, called fundamental operators, must be defined and implemented at the system level. Others can be expressed by using fundamental operators and other given functions.

Below, we use some examples to argue the necessary fundamental operators needed for detecting some general and specified topological relations, and to show how to create new operators by combinations of those fundamental operators and other functions and operators.

Since the disjoint relation exists in any combination of spatial object types (see Table 1), the operator DISJOINT is defined as a fundamental operator.

Example 1: Operators for checking topological relations between two points *p* and *q*.

There are only two kinds of topological relations between two points. They are either disjoint or not disjoint. We define the operator EQUALS to describe the not-disjoint relation.

p EQUALS *q* **iff** NOT(*p* DISJOINT *q*)

EQUALS is a non-fundamental operator defined by the fundamental operator DISJOINT. EQUALS can also be defined as

p EQUALS *q*
iff (XCOORD(*p*) = XCOORD(*q*)) AND
(YCOORD(*p*) = YCOORD(*q*))

This example shows that non-fundamental operators can sometimes be defined in several ways.

Example 2: Function for checking closedness of a line *l*.

CLOSED(*l*) **iff** SP(*l*) EQUALS EP(*l*)

This example shows that spatial operators can be used to define spatial functions.

Example 3: Operators for checking spatial relations between a point *p* and a line *l*.

Two general topological relations are disjoint and not disjoint. We describe the not-disjoint relation by the operator COVERS defined as

l COVERS *p* **iff** NOT(*p* DISJOINT *l*)

Under the COVERS relation, two cases must be distinguished. The point *p* is either located at the end points, or on the other place of the line *l*. We define the operator P-AT-ENDS to describe the first case.

p P-AT-ENDS *l* **iff** (*l* COVERS *p*) AND
((*p* EQUALS SP(*l*)) OR

(*p* EQUALS EP(*l*)))

Example 4: Operators for checking spatial relations between a point *p* and a polygon *pg*. *pg* is a primitive polygon without holes.

There are three kinds of spatial relations. The point *p* is either disjoint from the polygon *pg*, or on the boundary of, or inside of the polygon *pg*. The first kind is checked by the operator DISJOINT. We define the operator COVERS to check the second kind.

pg COVERS *p*
iff BOUNDARY(*pg*) COVERS *p*

The third kind can be checked by the operator CONTAINS defined as

pg CONTAINS *p*
iff NOT(*pg* DISJOINT *p*) AND
NOT(*pg* COVERS *p*)

Example 5: Operators for checking spatial relations between two lines *li* and *lk*.

When two lines are not disjoint, they can be topologically connected in different ways. Figure 2 shows some simple, but fundamental examples. In group 1, two lines are connected at a point. We define the operator MEETS as a fundamental operator to detect these relations. To distinguish the relation 1a, the operator INTERSECTS is defined as

li INTERSECTS *lk*
iff (*li* MEETS *lk*) AND
(*li* DISJOINT SP(*lk*)) AND
(*li* DISJOINT EP(*lk*)) AND
(*lk* DISJOINT SP(*li*)) AND
(*lk* DISJOINT EP(*li*))

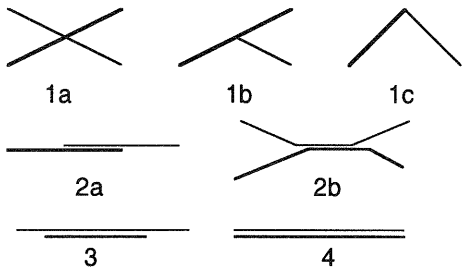


Figure 2. Topological relations between two lines.

The relation 1c can be checked by the operator MEETS-AT-ENDS defined as

li MEETS-AT-ENDS *lk*
iff (*li* MEETS *lk*) AND
((SP(*li*) EQUALS SP(*lk*)) AND
NOT (SP(*li*) EQUALS EP(*lk*))) OR
((SP(*li*) EQUALS EP(*lk*)) AND
NOT (SP(*li*) EQUALS SP(*lk*))) OR
((EP(*li*) EQUALS SP(*lk*)) AND
NOT (EP(*li*) EQUALS EP(*lk*))) OR
((EP(*li*) EQUALS EP(*lk*)) AND
NOT (EP(*li*) EQUALS SP(*lk*)))

We consider the situations in group 2 as the overlapping relation in which two lines partly intersect. The operator OVERLAPS is defined as a fundamental operator to detect the overlapping relation. Specific cases of the overlapping relation can be distinguished by combining the operator OVERLAPS with other predefined operators. The relation 2a is detected by the operator

li OVERLAPS-AT-ENDS *lk*
iff (*li* OVERLAPS *lk*) AND
((*li* COVERS SP(*lk*)) OR
(*li* COVERS EP(*lk*))) AND
((*lk* COVERS SP(*li*)) OR
(*lk* COVERS EP(*li*)))

The relation 2b is detected by the operator

li OVERLAPS-IN-MIDDLE *lk*
iff (*li* OVERLAPS *lk*) AND
(*li* DISJOINT SP(*lk*)) AND
(*li* DISJOINT EP(*lk*)) AND
(*lk* DISJOINT SP(*li*)) AND
(*lk* DISJOINT EP(*li*))

We call the situations in group 3 and 4 the covering relation. The operator COVERS is a fundamental operator to check the covering relation.

The group 4 is a special case of covering relation in which two lines are equal. To detect this special relation, the operator EQUALS is defined as

li EQUALS *lk*
iff (*li* COVERS *lk*) AND
((SP(*li*) EQUALS SP(*lk*)) AND
(EP(*li*) EQUALS EP(*lk*))) OR
((EP(*li*) EQUALS SP(*lk*)) AND
(SP(*li*) EQUALS EP(*lk*)))

Example 6: Operators for checking spatial relations between a line *l* and a polygon *pg*.

Spatial relations under the not-disjoint condition can be classified into four groups (Figure 3). We define operators MEETS, CONTAINS, COVERS and OVERLAPS as fundamental operators to detect the spatial relations of the respective groups.

The relations in group 1 can be distinguished by the following operators.

l MEETS-AT-END *pg*
iff (*l* MEETS *pg*) AND
(*l* MEETS BOUNDARY(*pg*))

pg MEETS-PART *l*
iff (*pg* MEETS *l*) AND
(BOUNDARY(*pg*) OVERLAPS *l*)

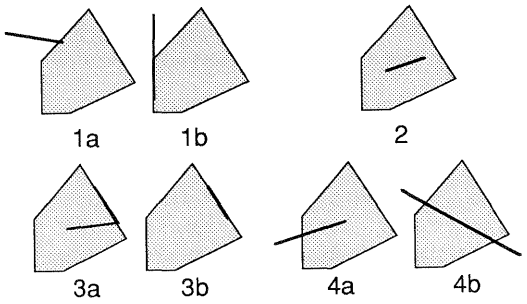


Figure 3. Topological relations between a line and a polygon.

The relation 3a is described by the operator

pg COVERS-PART *l*
iff (*pg* COVERS *l*) AND
(BOUNDARY(*pg*) OVERLAPS *l*)

and the relation 3b is detected by the operator

pg BOUND-COVERS *l*
iff (*pg* COVERS *l*) AND
(BOUNDARY(*pg*) COVERS *l*)

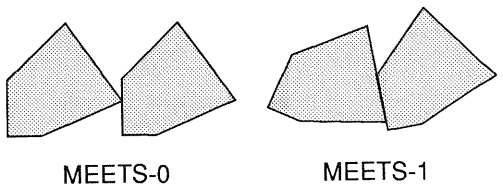
The relations in group 4 can be respectively distinguished by the operators ENTERS and PASSING defined as

l ENTERS *pg*
iff (*pg* OVERLAPS *l*) AND
((*pg* CONTAINS SP(*l*)) OR
(*pg* CONTAINS EP(*l*)))

l PASSING *pg*
iff (*pg* OVERLAPS *l*) AND
(*pg* DISJOINT SP(*l*)) AND
(*pg* DISJOINT EP(*l*))

Example 7: Operators for checking topological relations between two polygons *pg1* and *pg2*.

All the topological relations in table 1 are defined as fundamental operators for two polygons. Two polygons, however, can meet, cover, or overlap in different ways. These specific relationships can be distinguished by combinations of fundamental operators and other predefined operators and functions. The following is the meets-0 and meets-1 examples (Figure 4).



Figrue 4. Two specific cases of the MEETS relation.

pg1 MEETS-0 *pg2*
iff (*pg1* MEETS *pg2*) AND
(BOUNDARY(*pg1*) MEETS
BOUNDARY(*pg2*))

pg1 MEETS-1 *pg2*
iff (*pg1* MEETS *pg2*) AND
(BOUNDARY(*pg1*) OVERLAPS
BOUNDARY(*pg2*))

6. Spatial queries

Suppose that a spatial database consists of three spatial relations:

```
Towns(Pt:POINT,
      Name:TEXT,
      Population:INTEGER)

Roads(Li:LINE,
      Class:INTEGER,
      No:INTEGER)

Provinces(Pg:POLYGON,
          Name:TEXT,
          Population:INTEGER)
```

Here, POINT, LINE and POLYGON are spatial data types.

Query 1: Find towns inside province B passed by second class roads.

```
Roads2 <- *(a:Roads,b:Provinces)
          WHERE[(a.Class = 2) AND
                (b.Name = 'B') AND
                ((b.Pg OVERLAPS a.Li) OR
                (b.Pg COVERS a.Li) OR
                (b.Pg CONTAINS a.Li))]

                [Li:a.Li,No:a.No];

Towns1 <- *(a:Towns,b:Provinces)
          WHERE[(b.Name = 'B') AND
                (b.Pg CONTAINS a.Pt)]

          [Pt:a.Pt,Name:a.Name,
           Population:a.Population];

Towns2 <- *(a:Towns1,b:Roads2)
          WHERE[b.Li COVERS a.Pt]

          [Pt:a.Pt,Name:a.Name,
          Population:a.Population];
```

Relation Towns2 contains information about towns in province B which are passed by second class roads.

Query 2: Find towns in province B which are passed by second class road No. 3.

```
Towns3 <- *(a:Towns2,b:Roads2)
            WHERE[(b.No = 3) AND
                    (b.Li COVERS a.Pt)]

            [Pt:a.Pt,Name:a.Name,
             Population:a.Population];
```

7. Conclusions

Spatial information is carried by spatial objects and their relations. Points, lines and polygons are three distinct types of spatial objects in the two dimensional space. Spatial data types are a useful mechanism for describing spatial objects in spatial databases.

Information about instances of spatial data types are extracted by using functions defined on the data types. These functions are used either to extract subsets of data or to calculate new data from the instances. The composition and combination of existing functions can form new functions.

To check the spatial relations of two objects in the database, a few system-defined fundamental topological operators are necessary. More operators can be built from the fundamental operators and other functions. By including spatial operators in the query language, spatial analyses can be performed as a number of queries to the spatial database.

Implementation of operators and functions discussed here as well as others included in the spatial query language GeoSAL (Svensson et al 1991) is going on at the National Defence Research Establishment in Stockholm, Sweden.

Acknowledgements

The author is grateful for valuable comments from Prof. Dr. Friedrich Quiel at the Royal Institute of Technology in Stockholm, and Dr. Per Svensson and Dr. Karsten Jöred at the National Defence Research Establishment in Stockholm, Sweden

References

Armstrong, M.A., 1979. Basic topology. McGraw-Hill.

Egenhofer, M. J. & Herring, J. R., 1990. A mathematical framework for the definetion of topological relationships. In: Proceedings of the 4th Int. Symposium on Spatial Data Handling, Zurich, Switzerland, pp. 803-813.

Egenhofer, M. & Franzosa, R., 1991. Point-set topological spatial relations. Int. Journal of Geographical Information Systems, 5(2):161-174.

Huang, Z., Svensson, P. & Hauska, H., 1992. Solving spatial analysis problems with Geo-SAL, a spatial query language. to appear In: Proceedings of the 6th Int. Working Conf. on Scientific and Statistical Database Management, Switzerland, June 9-11, 1992.

Kainz, W., 1989. Order, topology and metric in GIS. In: Proceedings of ASPRS/ACSM, 4:154-160.

Kainz, W., 1990. Spatial relationships-topology versus order. In: Proceedings of the 4th Int. Symposium on Spatial Data Handling, Zurich, Switzerland, pp. 814-819.

Kemp, Z., 1990. An object-oriented model for spatial data. In: Proceedings of the 4th Int. Symposium on Spatial Data Handling, Zurich, Switzerland, pp. 659-668.

Molenaar, M., 1991. Status and problems of geographical information systems. The necessity of a geoinformation theory. ISPRS Journal of Photogrammetry and Remote Sensing, 46:85-13.

Oosterom, P. & Bos, J., 1989. An object-oriented approach to the design of geographical information systems. In: Design and Implementation of Large Spatial Databases, ed. by Buchmann, A. et al, Lecture Notes in Computer Science, Vol. 409:255-269, Springer-Verlag.

Stanat, D. F. & McAllister, D. F., 1977. Discrete Mathematics in Computer Science. Prentice-Hall.

Svensson, P. & Huang, Z., 1991. Geo-SAL: A query language for spatial data analysis. In: Advances in Spatial Databases, ed. by O. Gunther & H.-J. Schek, Lecture Notes in Computer Science, Vol. 525:119-140, Springer-Verlag.

Unland, R. & Schlageter, G., 1990. Object-oriented database systems: concepts and perspectives. In Database Systems of the 90's, ed. by Blaser, A., Lecture Notes in Computer Science, Vol. 466:154-197, Springer-Verlag.