SPATIAL AND TOPOLOGICAL DATA MODELS

Ying Deng and Peter Z. Revesz

Computer Science and Engineering Department
University of Nebraska - Lincoln
revesz@cse.unl.edu

Abstract: Spatial and topological data models are increasingly important in business applications such as urban development planning, transportation and traffic control, decision support in agriculture, pollution and environment analysis, fire and flood prevention, etc. that require handling spatial and topological data more efficiently and more effectively than older models, for example the relational data model. In this survey we compare several alternative spatial and topological data models: the *Spaghetti Data Model*, the *Vague Region Data Model*, the *Topological Data Model*, *Worboys' Spatiotemporal Data Model* and the *Constraint Data Model*. We first describe how spatial and/or topological data are represented and give examples for each data model. We also illustrate by examples the use of an appropriate query language for each data model.

THE SPAGHETTI DATA MODEL

The Spaghetti data model (Laurini and Thompson, 1992) is a popular model for representing spatial data that occur in for example Computer-Aided-Design (CAD) (Kemper and Wallrath, 1987) and Geographical Information Systems (GIS) (Worboys, 1995; Zeiler, 1997) applications. The reason why this model is so popular is that there are many efficient algorithms for detecting properties in this model (Preparata and Shamos, 1985). In addition, the Spaghetti model is simple to use and offers in most applications a sufficient approximation to reality. There are several extensions of this model, for example the parametric 2-spaghetti (Chomicki and Revesz, 1999) and the parametric rectangles (Cai et al., 2000) models, which we do not review here.

Data Representation

In the Spaghetti model, the information in an n-dimensional space is represented using a set of m-dimensional hyperspaces, with $m < n$. This means that in a two-dimensional plane, we only consider polygons, the boundary of which contain line segments and points. More concretely, we use here (Paredaens, 1995):

- Points, which are represented using their coordinates *(x, y)*;
- Graphs, whose data structure is a finite set of pairs of points;
- Polylines, whose data structure is a finite sequence of points;

- Polygons that are represented by non-selfintersecting closed polylines.
- Complex polygons, that can contain holes, which are again complex polygons (up to a finite level);
- Objects are sets of polygons, points or graphs.

Figure 1.1 shows a city with a park and a university and river that runs through it.

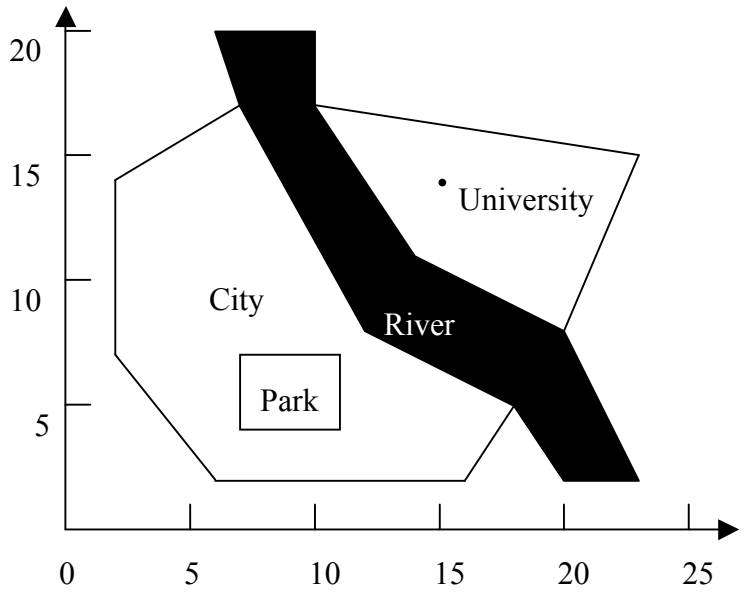


Figure 1.1: An example for Spaghetti model

In the Spaghetti model, the above figure can be represented by the relation in Table 1.1 as follows.

ID	(x, y)'s
University	{ (15, 14) }
River	{ (6, 20), (7, 17), (12, 8), (18, 5), (20, 2), (23, 2), (20, 8), (14, 11), (10, 17), (10, 20), (6, 20) }
Park	{ (7, 4), (11, 4), (11, 7), (7, 7), (7, 4) }
City	{ (6, 2), (16, 2), (18, 5), (12, 8), (7, 17), (2, 14), (2, 7), (6, 2) }
City	{ (10, 17), (14, 11), (20, 8), (23, 15), (10, 17) }

Table 1.1: Spaghetti representation

Note that the polyline is represented by a sequence of points. Each of the polygons is represented by a sequence of its corner points. To distinguish polylines and polygons the first and the last points of a polygon are the same which means this is a closed polygon.

ARC/INFO Queries

Database systems based on the Spaghetti data model usually provide queries that tell whether two polygons overlap, whether a point lies in a polygon or on a line segment, whether two line segments intersect, whether a polyline self-intersects, whether a polygon is contained in another one, etc. The evaluations or implementations of all these queries are solvable in polynomial time complexity (Preparata and Shamos, 1985).

ARC/INFO is currently the primary GIS system available from ESRI. It handles both spatial information and descriptive information based on the spatial-relational data model. The spatial information in ARC/INFO is represented through four classes of basic data components: arcs, nodes, label points, and polygons.

ARC/INFO provides queries that retrieve its non-spatial attribute data as relational databases do, and queries with spatial selection ability that traditional databases do not have. There are three general methods for selecting geography features in ARC/INFO (Zeiler, 1997).

Spatial selection. A user can use the mouse cursor to pick one or more geographic features. The user can collect features within boxes, circles, and polygons interactively. For example, the user can choose all the houses located within a certain polygon or circle.

Logical selection. A user can build simple or compound logical queries to select geographic features. This kind of queries is based on non-spatial attributes, which is expressed as traditional SQL queries. The results of logical selections can be connected with spatial information through internal identifiers. An example would be "Find all the buildings with an area greater than 3200 square feet." The feature 'area' is one of the attributes associated with the polygons that represent the buildings.

Overlap selection. A user can select geographic features such as contained within, overlap, adjacent to, or within a buffer of a certain distance etc. For example, the **RESELECT** and **OVERLAP** commands can be used to "select wells within 2000 feet of a selected stream arc". This query can be expressed as:

RESELECT WELLS POINTS **OVERLAP** STREAMS ARCS 2000

where STREAMS and WELLS are two existed relations in the database, POINTS and ARCS are two feature classes predefined in the system. **RESELECT** and **OVERLAP** are reserved key words of the system and 2000 indicates the distance.

THE VAGUE REGION DATA MODEL

Most spatial data models assume that the boundaries of spatial objects are sharply delimited by points, lines and regions. This is not always a natural assumption, for example when we need to consider a vague region like the region of regular customers of

a supermarket. A particular example of modeling spatial vagueness and querying vague region data models is reviewed in this section based on the model of Erwig and Schneider (1997). Several other proposals use fuzzy logic, for example (Edwards, 1994), that we do not review here.

Data Representation and Operations

In the model proposed by Erwig and Schneider (1997) each *vague region* is a pair of disjoint regions. The first region, called the *kernel* describes the determinate part of the vague region, that is, the area, which is definite and always belongs to the vague region. The second region, called the *boundary*, describes the vague part of the vague region, that is, the area for which we cannot say with any certainty whether it or parts of it belong to the vague region or not. Boundaries need not necessarily be one-dimensional structures but can be regions. Kernels and boundaries may be adjacent, they may have holes which themselves can contain a hierarchy of kernels and boundaries with holes.

We give an example shown as Figure 2.1. The region *p* denotes the areas of the Great Plain in the mid-west of America, and the region *b* denotes the areas that birds usually inhabit in the spring. The gray parts represent boundaries of these two classes of areas, which means that it is not definite that they belong to the Great Plain or not and the birds inhabit or not.

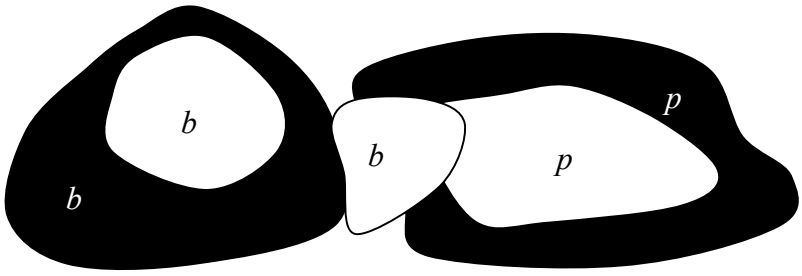


Figure 2.1: Kernels and boundaries of vague regions

Operations and SQL-like Queries

Let us denote the vague region *v* as a pair of disjoint regions *(k, b)* where *k* gives the kernel of *v* and *b* gives the boundary of *v*. The notation $v^k = k$ and $v^b = b$ is employed and the exterior of v is defined as $v^e = \neg(k+b)$ where '+' is the union operation and '¬' is the complement operation in a general sense.

The following figure shows the operations of *union*, *intersection*, *difference* and *complement* on two vague regions. A black circle represents a kernel, a gray circle represents a boundary, and a white circle denotes the exterior. Each field of the tables denotes a possible combination of kernel, boundary and exterior. The label in each field specifies whether the corresponding operation result belongs to kernel, boundary or exterior.

difference	K	B	O
K	O	B	K
B	O	B	B
O	O	O	O

intersection	K	B	O
K	K	B	O
B	B	B	O
O	O	O	O

union	K	B	O
K	K	K	K
B	K	B	B
O	K	B	O

complement	K	B	O
	O	B	K

Figure 2.2: union, intersection, difference and complement operations on vague regions

In addition to the four basic spatial operations, the following operations are defined.

Boundary (v) : = (Ø, v^b)
Kernel (v) : = (v^k, Ø)
Invert (v): = (v^b, v^k)

When answering questions like "Does region u and v intersect?", sometimes we can neither return *true* nor *false* but *maybe* or *unknown*. Therefore, a three-valued logic is used as the range of Boolean predicates. Figure 2.3 shows the definition of the logic operators paralleling the definition of the operations for vague regions. T, F, and M are used as abbreviations for *true*, *false*, and *maybe*.

and	T	M	F
T	T	M	F
M	M	M	F
F	F	F	F

or	T	M	F
T	T	T	T
M	T	M	M
F	T	M	F

not	T	M	F
	F	M	T

Figure 2.3: Three-valued logic operators

A number of numeric operations are defined based on the corresponding functions for regions. The *min-area* of a vague region *v* is the area of the kernel of *v*. The *max-area* of a vague region *v* is the area of the kernel and the boundary of *v*. Similarly the distance between two vague regions is a vague value. The *max-dist* of two vague regions *v* and *u* is the distance between the kernels of *v* and *u*. The min-*dist* of two vague regions *v* and *u* is defined as the distance between the maximal extensions of *v* and *u*, taking kernel and boundary into account.

By assuming a relational data model where tables may contain vague region objects, an SQL-like query language is used to indicate how the operations can play as a part of a spatial query language. We give some examples as follows.

First, let us look at a simple query: “Find all regions where lack of water is a problem for cultivation.” The query can be expressed as

SELECT region **FROM** weather **WHERE** climate = dry

assuming a table *weather* exists having a column named *region* containing vague region values for various climate conditions given by the column *climate*. A similar query would ask for bad soil regions as a hindrance for cultivation. The result of both queries is a set of vague regions.

If now we want to find out regions where cultivation is impossible due to either reason, the query is then expressed using *union* operation as following:

(SELECT SUM(region) **FROM** weather **WHERE** climate = dry)
UNION
(SELECT SUM(region) **FROM** soil **WHERE** quality = bad)

where *sum* is a built-in function which aggregates a set of regions.

By prefixing that any predicate with *maybe* which causes the predicate to fail only if it returns false, the query “Find out all areas where people are definitely or possibley endangered by pollution.” can be expressed as

SELECT areas.name
FROM pollution, areas
WHERE areas.use = inhabited **AND** pollution.region **MAYBE INTERSECTS** areas.region

This model is capable of describing many other aspects of vague spatial objects. Based on the exact spatial modeling concepts, it allows a smooth migration from existing models to vague concepts. Further extensions include modeling vague point and lines, integration of vague regions into other data models and query languages.

THE TOPOLOGICAL DATA MODEL

Topology has been used for modeling spatial data and their composition for a long time. Characteristic of topological properties is that they do not distinguish between two databases that can be obtained from each other by a topological deformation. Such databases are usually called *topologically equivalent*. Queries that only involve topological properties are of interest in this class, such as adjacency, connectivity, and containment. In recent years, there is a growing interest of the topological data model among the spatial database community. In this section, we introduce the PCA-structure, a topological data model based on labeled points, curves and areas.

Data Representation

Spatial databases concerning topological properties are expected to represent information *lossless,* in the sense that two databases that are not topologically equivalent are represented differently. In addition, *topological invariance* is desired in the sense that two topologically equivalent databases are represented identically. (Kuijpers et al., 1995) proposes a data structure, *PCA-structure* that gives an invariant and lossless representation used to represent spatial data in the Euclidean plane $\mathbf{R}^2$.

Conceptually, a *spatial database* in the *topological data model* consists of a finite set of labeled points, a finite set of labeled curves and a finite set of labeled areas. Each point label is assigned to a distinct point in the Euclidean plane $\mathbf{R}^2$. Each curve label is assigned to a distinct non-self-intersecting continuous curve in the plane that starts and ends in a labeled point and does not contain any other labeled points except these. Two curves only intersect in a labeled point. Each area label is assigned to a distinct area formed by the labeled curves. We give an example of such a database as shown in Figure 3.1. Note that this definition allows curves to start and end in the same point, i.e., the database may contain loops.

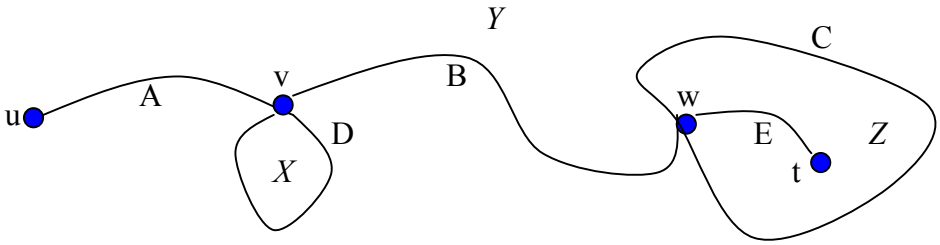


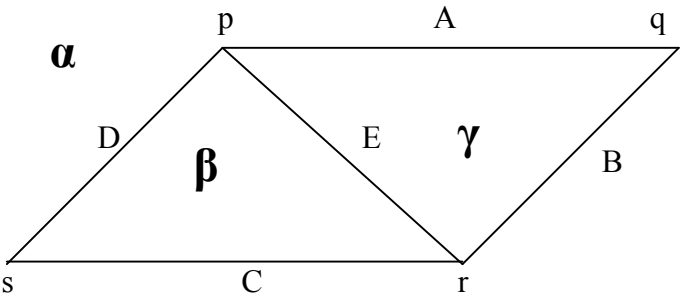
Figure 3.1: An example spatial database in the topological data model

A spatial database is represented by means of a classical database consisting of four relations, R1, R2, R3, and R4, on the labels of points, lines, and areas as follows (Kuijpers et al., 1997).

- R1 gives for every curve its two endpoints;
- R2 gives for every curve its two adjacent areas;
- R3 gives for each area its border of alternative curves and points;
- R4 gives for each point its neighborhood of alternative curves and areas.

R1 indicates that every line has exactly two end-points. R2 indicates that every line is the border between exactly two areas. R3 says that every area is surrounded by an ordered cycle of curves and points indicating the border of an area. R4 says that every point is surrounded by an ordered cycle of curves and areas indicating the neighborhood of the point. Different orders may be used to distinguish outer borders and holes in areas. Neither the exact position of the cells nor their length or surface is given by this representation, which is the general case in topological data models where only the

topological properties are determined. Figure 3.2 illustrates the relations for the depicted spatial database.



R1	R2	R3	R4
A p q	A α β	α p A 1	p A γ
A q p	A β α	α q B 2	p E β
B q r	B α γ	α r C 3	p D α
B r q	B γ α	α s D 4	q B α
C r s	C α β	β p E 1	q A γ
C s r	C β α	β r C 2	r C β
D p s	D α β	β s D 3	r E γ
D s p	D β α	γ p A 1	r B α
E p r	E β γ	γ q B 2	s C α
E r p	E γ β	γ r E 3	s D β

Figure 3.2. The relations R1, R2, R3, and R4 illustrated

For each labeled point in such a spatial database, it is more convenient to make a circular alternating list of area labels and curve labels rather than a set of tuples. The list corresponds respectively to the areas and curves that an observer sees when he makes a clockwise full turn and scans the environment of the point. The observation from an object *p* is denoted by *Obs(p)*. Figure 3.3 illustrates an example for the Obs() representation. The alternating list for the point with label *t* is (X B X A Y A). Note that a point which is isolated from the remainder of the database consists of one single area label.

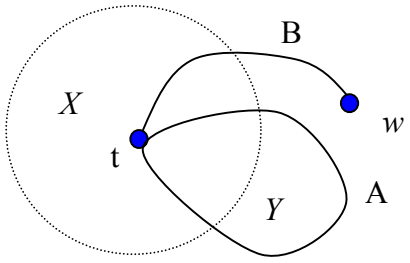


Figure 3.3: An observation of a database from point t

Now the *PCA-structure* is defined as follows. For a given database *D,* the PCA-structure of *D* is the data structure (*P*, *C*, *A*, α^{∞}, Obs()) if *P* is the set of point labels, *C* is the set of curve labels, *A* is the set of area labels, α^{∞} is the name of the unbounded area and Obs() is a function that associates with each element *p* of *P*, the observation of *D* from *p*. Clearly the topological relations or properties can be reconstructed from the observations. The PCA-structure is an invariance and lossless representation of a database (Kuijpers et al., 1995), which allows users to concentrate only on the topological aspects of the spatial data, and on all of them.

The L_{PCA} Query Language

To query spatial databases in the PCA topological data model, the first-order language L_{PCA} is introduced in (Kuijpers et al., 1997). L_{PCA} has three sorts of variables: lower-case characters are used for point variables, such as p, q, r, …; capitals are used for curve variables such as A, B, C, …; and Greek characters are used for area variables such as α, β, γ, …. The language has one constant α^{∞}, the label for the unbounded area. A *term* in L_{PCA} is

- p = q with p and q point variables;
- A = B with A and B curve variables;
- α = β with α and β are area variables;
- $\alpha = \alpha^{\infty}$ with α an area variable;
- AαB ⊂ Obs(p) with p a point variable, A and B curve variable and α an area variable or the area constant α^{∞};
- αA β ⊂ Obs(p) with p a point variable, A a curve variable, α and β area variables or the area constant α^{∞}; or
- α = Obs(p) with p a point variable, α an area variable or the area constant α^{∞}.

An *expression* in L_{PCA} is

- a term;
- a combination of expressions using ∧ , ∨ , ¬, →; or
- (∃p)φ, (∃A)φ and (∃α)φ with φ an expression and p a point variable, A a curve variable and α an area variable.

A *query* expressed in L_{PCA} has the form

$\{(p_1, \ldots, p_n, A_1, \ldots, A_m, \alpha_1, \ldots, \alpha_k) \mid \varphi(p_1, \ldots, p_n, A_1, \ldots, A_m, \alpha_1, \ldots, \alpha_k)\}$,

where $\varphi(p_1, \ldots, p_n, A_1, \ldots, A_m, \alpha_1, \ldots, \alpha_k)$ is an expression in L_{PCA} with free point variables $p_1, \ldots, p_n$, free curve variables $A_1, \ldots, A_m$ and free area variables $\alpha_1, \ldots, \alpha_k$.

Consider the query "Does the spatial database contain a loop?". This is a simple Boolean query with n = 0, m= 0, and k=0. It is expressed by the formula

(∃p)(∃α)(∃β)(∃A)¬α = β ∧ αAβ⊂Obs(p) ∧ βAα⊂Obs(p).

WORBOYS' SPATIOTEMPORAL DATA MODEL

The information, which is referred to space, is often related to time. Space, time and process are closely connected. In recent years, there have been a large number of researches on spatial-temporal databases (Snodgrass, 1992; Worboys, 1994; Chomicki and Revesz, 1997). Generally, there are two classes of time to be handled in an information system, the database time and the event time. The database time is the time when transactions actually take place with the information system. The event time is the time when the events actually occur in the application. Different systems provide supports either to the database time or to the event time, or both. In this section, we introduce a spatial-temporal data model which uses simplicial complexes to model pure spatial information and two orthogonal dimensions to represent database time and event time. A query algebra, similar in some respects to relational algebra is also presented.

Data Representation

The spatial-temporal model represents spatial data based on the simplicial model developed by Worboys (Worboys, 1994) and Egenhofer (Egenhofer et al., 1989). Spatial objects are classified according to their spatial dimension. For each dimension, a minimal object exists, called *simplex*. For example, 0-simplex represents node, 1-simplex stands for edge, 2-simplex stands for triangle. Any n-simplex is composed of (n+1) geometrically independent simplices of dimension (n-1). For example, an edge, a 1-simplex, is bounded by two 0-simplices (two nodes). A *face* of a simplex is any simplex that contributes to the composition of the simplex. For instance, the bounding edges of a triangle are faces of this triangle. The n-simplex is a face of itself. An ordered n-simplex S_n is represented by its vertices in the form $S_n = < x_0, x_1, ..., x_n >$. An *orientation* of a simplex fixes the vertices to lie in a sequence and is defined through the associated ordered simplices. A *simplicial complex* is a finite collection of simplices and their faces. The dimension of a complex is taken to be the largest dimension of the simplices of this complex. If the intersection between two simplices of this collection is not empty, then the intersection is a simplex, which is a face of both simplices. The following Figure 4.1 shows an example in which the line segments represents the border of a piece of land and the shaded area represents a building on this land.

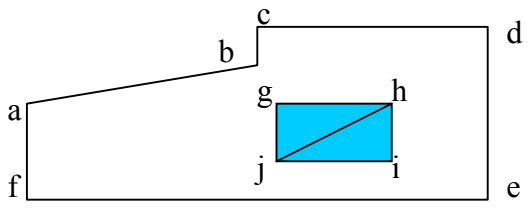
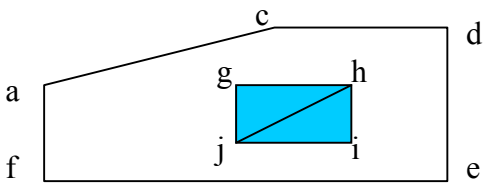


Figure 4.1: Simplicial complex{ab,bc,cd,de,ef,fa,ghj,hij}

A *bitemporal element* or BTE is defined to be the union of a finite set of Cartesian products of intervals of the form $I_D \times I_E$, where I_D is an interval of database time and I_E is an interval of event time. The semantics expressed by a BTE T are that $(t_D, t_E) \in T$ if and

only if at time t_D there is information in the database that the object temporally referenced by T exists as event time t_E. Suppose that the information in Figure 4.1 is of year 1988. In year 1990, the border of the land changed, as shown in Figure 4.2, and the database was updated. The BTE associated with simplex *ac* is illustrated in Figure 4.3.



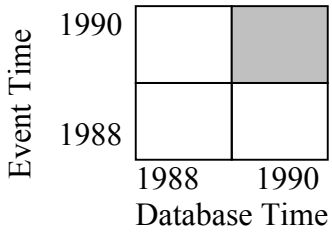


Figure 4.2: The changed simplicial complex

Figure 4.3: An example of BTE

An *ST-simplex* is an ordered pair <S, T>, where S is a simplex and T is a BTE. Intuitively, an ST-simplex is an elemental spatial objects (simplex) to which a bitemopral reference is attached. The projection operator is defined as follows. Let the ST-simplex R = <S, T>. Then $\pi^s(R) = S$ and $\pi^t(R) = T$.

An *ST-complex, C* is a finite set of ST-simplexes satisfying the properties:

1. The spatial projections of ST-simplexes in C are pair wise disjoints. Taken together, they form a spatial simplicial complex.

2. $\forall$ R, R' $\in$ C| $\pi^s(R)$ is a face of $\pi^s(R')$ implies that $\pi^t(R) \geq \pi^t(R')$.

The second condition, for example, ensures that the end points of a line segment are always extant when the line segment itself extant.

Querying Spatial-Temporal Data

Worboys gives a query algebra on the spatial-temporal model. The possible operations on ST-complexes include equals(=), subset($\subset$), ST-union($\cup$), ST-intersection($\cap$), ST-different(-), S-select(δ^s), T-select(δ^t), S-project(π^s), T-project(π^t), boundary(∂), ST-β-product($\times_\beta$) and so on. We discuss some of the above operations, which contribute to our query examples.

The *boundary* operation upon an n-simplex S_n determines all (n-1)-faces of S_n. The boundary of a simplicial complex can be determined as the sum of the boundaries of all its simplices S_n. For ST-complex C, its boundary ∂C is defined as

$$\partial C = \{<S, T> \in C \mid S \in \partial \pi^s(C)\}$$

For two ST-complexes, C and C', we define C$\subset$C' if and only if for each (x, y, z, w)$\in$<S, T>$\in$C, there is <S', T'>$\in$C' such that (x, y, z, w)$\in$<S', T'>.

Let C1 and C2 be two purely spatial simplicail complexes. A *common refinement* of C1 and C2 is a simplicial complex, which has the same planar embedding as the union of the embeddings of C1 and C2. Let β be a Boolean set operation on BTEs. Let simplicial complex R be a common refinement of $\pi^s(C1)$ and $\pi^s(C2)$. Then define $C1 \times_\beta C2$ to be the smallest ST-complex (with respect to the ST-subset relation) which contains the set of ST-simplexes $\{< S, T^s_1 \beta T^s_2 > \mid S \in R\}$ where T^s_1 and T^s_2 are the BTEs associated with the spatially smallest faces of $\pi^s(C1)$ and $\pi^s(C2)$, respectively, which contains S. The *ST-intersection* is defined as

$$C1 \cap C2 = C1 \times \cap C2.$$

Let C be an ST-complex. Then the temporal selection operation $\delta^t{}_\Phi$ is defined to be the smallest ST-complex containing the set of ST-complexes $\{<S, T> \in C \mid \Phi(T)\}$ where Φ is a first-order formula on BTEs.

Suppose we have ST-complexes C1 and C2 representing a path and Jane's house, respectively. Now consider the query "Has Jane's house ever shared a common boundary with the path?". The query can be expressed as

$$\partial C1 \cap \partial C = \emptyset$$

the query "Does the path currently pass through land that was part of Jane's house?" then can be expressed as

$$\pi^s(\delta^t{}_{t \supseteq \text{now DB}}(C1)) \cap \pi^s(C2) = \emptyset$$

where *now DB* indicates a BTE with database time *now* and the intersection operator is purely spatial intersection of spatial complexes.

It is not claimed that the spatial-temporal operations defined by Worboys (Worboys, 1994) are complete in any sense, just as there is complete list of purely spatial operators.

THE CONSTRAINT DATA MODEL

Constraint databases provide a powerful framework to model and retrieve spatial data. A constraint database uses constraints on a specific decidable logical theory both to model and retrieve data. At the data level, constraints are able to finitely represent possibly infinite sets of relational tuples. With respect to data modeling, constraints serve as a unifying data type for the (conceptual) representation of heterogeneous data. In particular, the benefit of this approach is emphasized when complex knowledge (for example, spatial or temporal data) has to be combined with some descriptive non-structured information (such as name or figures), when several types of spatial objects with different dimension (like points, lines, convex polygons and concave polygons) have to be represented. At the query language level, constraints increase the expressive power of simple relational languages by allowing mathematical computation and recursive

expressions. In this section, we review how constraint databases represent and query data with constraints.

Data Representation

A framework for using constraint databases is presented in (Kanellakis et al., 1995). The following three definitions are from (Kanellakis et al., 1995).

A *generalized k-tuple* is a quantifier-free conjunction of constraints on *k* variables ranging over a domain *D*. Each generalized k-tuple represents in a finite way an infinite set of regular k-tuple.

A *generalized relation of arity k* is a finite set of generalized k-tuples with each k-tuple over the same variables. Suppose relation R contains the set of points on the line with slope four. It is impossible for a relational database to enumerate all the points on the line, however the line can be finitely represented by a generalized 2-tuple R(x, y):- y = 4x in a natural sense. A *generalized database* is a finite set of generalized relations.

Constraint databases are parameterized by the type of constraint domains and constraints used. Real polynomial inequality constraints, dense linear order inequality constraints, Boolean equality constraints, and set constraints are typical types attracting a great deal of interest (Revesz, 1998).

It is generally required that a spatial database contain an elegant framework to combine geometric and thematic information, be as general as possible and not be designed for one particular area of application, have a formally defined semantics that is closed under set theoretic, geometric and topological operations, i.e. defined in terms of finite representations and use efficient implementation techniques, especially for the operations on n-dimensional objects.

The polynomial data model is well suited to model spatial objects, which require exact geometrical and geographical information. The task of the spatial database is to store a representation of some geographic areas, which are typically two-dimensional maps. Such geographical information can be described precisely. Higher dimensional spatial objects, unbounded and topologically non-closed geometric figures, which most other models can not handle, can also be represented and manipulated with polynomial inequality constraints. But practical constraint database systems (Brodsky et al., 1999; Grumbach et al., 1998; Revesz et al., 2000) typically use linear constraints for which efficient algorithms are available. For most problems with non-linear polynomial constraints, there are no efficient solutions currently.

The linear data model approximates spatial objects using linear representable objects, e.g. points, line segments, polygons, i.e. only linear inequality constraints are allowed. The simplicity of linear data is very attractive to spatial data modeling and there also exist efficient algorithms to implement the variety of operations on spatial data (Brodsky et al.,

1993; Brodsky and Kornatzky, 1995; Huynh et al., 1990; Lassez, 1990). Models with set Boolean constraints also exist (Revesz, 1998).

Suppose we have the following town map as shown in Figure 5.1, which depicts the border of the town and highways near the town.

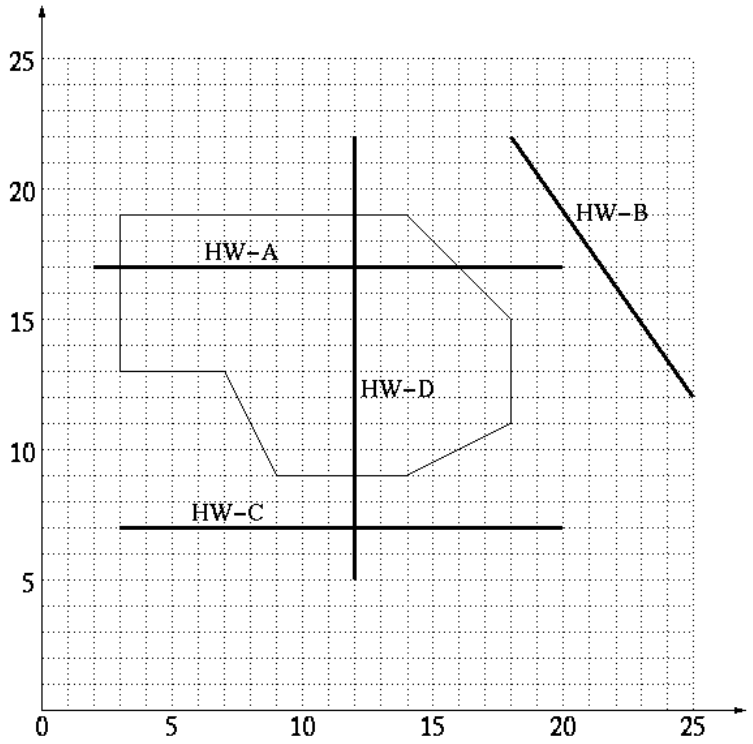


Figure 5.1 the map of a town

The spatial information of this town can be represented using two relations in a constraint database. One relation represents a town region and the other contains the highways around the town. The two relations are as follows.

Name	Geometry
The- Town	$x \geq 3, x \leq 18, y \geq 13, y \leq 19, 7y-6x \geq 49.$
The-Town	$y \geq 9, 7y-6x \leq 49, x \leq 18, y+2x \geq 27, 2y-x \leq 4, x+y \leq 33.$

Table 5.1: the Town relation

Name	Geometry
HW-A	$y=17, x \geq 2, x \leq 20.$
HW-B	$10x+y=234, y \geq 12, y \leq 22.$

HW-C	$y=7, x \geq 3, x \leq 20.$
HW-D	$x=12, y \geq 5, y \leq 22.$

Table 5.2: the Highway relation

The town region is the collection of all the point-sets that satisfy the constraints on the *Geometry* attribute. All the *(x, y)* pairs that satisfy either of the two sets of constraints are in relation *Town*. The *Highway* relation contains four constraint tuples, which together represent the set of points lies on some highway in the map.

The Datalog Query Language

Datalog (Ullman, 1988), the primary example of a deductive query language, utilizes logic as a way to represent knowledge and as a language for expressing operations on relations. Datalog derives new relations from input relations using rules. Each Datalog query consists of a finite set of rules of the form

$$R0(x_1, \ldots, x_k) : - \ R1(x_{1,1} \ldots, x_{1,k1}), \ldots, Rn(x_{n,1}, \ldots, x_{n,kn}).$$

The x's are either variables or constants in the domain. If the variables on the right side are substituted by constants, which make the right side true, then the left side must also be true. In general, rules in Datalog define the true instances of certain predicates, with R0 in the above form, in terms of certain other predicates, R1, …, Rn, where each Ri is either an input relation name or a derived relation name.

The following query finds all the highways that pass through the town.

$Q_{passthrough}$(name):- Town(x, y), Highway(name, x, y).

The query $Q_{connect}$ finds all the highways that are connected to the town directly or indirectly (if intersecting any passing through highway).

$Q_{connect}$(name1):- $Q_{passthrough}$(name1), Highway(name1, x, y).

$Q_{connect}$(name):- $Q_{connect}$(name1), Highway(name1, x, y), Highway(name, x, y).

The relationship between relations with same relation names but different rule bodies, is the logical OR. The comma symbols represent conjunctions between constraints. A query is *recursive* is the body of any of its rules constraints a derived relation. Otherwise, the relation is called *non-recursive*. For instance, the query $Q_{passthrough}$ is a non-recursive query and $Q_{connect}$ is a recursive query.

Syntactically, Datalog is a fragment of predicate calculus extending relational calculus with intentionally defined relations. In relational calculus each query defines a single output relation which is not named explicitly and all other relations are input relations. In

Datalog, a query may define several output relations, which are referred to by names within the query, that is, built-in relations are allowed in Datalog. There are expressions in recursive Datalog, such as the $Q_{connect}$ query that cannot be expressed in relational algebra or relational calculus. The relational calculus underlying most commercial relational query languages is a form a logic that can be obtained from a non-recursive Datalog program with the substitution of logical OR of the rule bodies. Since most basic operations of computational geometry can be described in Datalog with real polynomial constraints, this implies the potential capability of constraint databases to be used in spatial applications.

CONCLUSION

No one data model can be claimed satisfying all general requirements for a wide range of applications. The Spaghetti data model and queries in the ARC/INFO system cannot handle both spatial and temporal information. The vague region model is limited in modeling regions exclusively. Moreover, the integration of vague concepts into existing models might cause trouble and be tedious since it either requires a redefinition of the data types or a redefinition of the operations. The topological data model overcomes the problem of the finiteness of computers and their numbering systems, recording topological properties explicitly and allows efficient queries on topological relations. But it needs to be complemented by qualitative analysis of the semantics of the spatial objects to provide intelligent spatial reasoning. In Worboys' spatial-temporal data model, the temporal and the spatial arguments are independent. Hence we cannot describe for example the relationship that exists between time and the area covered by an incoming tide (Chomicki and Revesz, 1999). The constraint data model uses constraints to model and retrieve data, which has shown powerful expressiveness, strong abilities to handle both spatial and temporal information and extend to n-dimensional space in a natural sense.

It is necessary to examine tradeoffs utilizing a specific set of usage-based criteria so that the overall quality or suitability of a specific data model can be evaluated within a particular context. The general criteria include completeness, robustness, versatility, efficiency and ease of generation. Current and anticipated spatial data volumes have generated a two-faceted problem. The first one is that existing data structures are inefficient and inflexible to meet current requirements. The second one is that format conversions between different data structures to satisfy the current range of required applications produces significant processing overhead. As the size of any database becomes very large, several important problems rise which must be dealt with efficiency, heterogeneity, accuracy, and security (Adam and Gangopadhyay, 1997; Floriani et al., 1993; Peuquet, 1988). Further research to meet all these requirements is scientifically exciting and commercially important.

REFERENCES

N. R. Adam, A. Gangopadhyay, Database Issues in Geographic Information Systems, Kluwer Academic Publishers, 1997.

A. Brodsky, J. Jaffar, M. J. Maher. Toward Practical Constraint Databases. *Proc. 19th VLDB*, pp. 322-331, 1993.

A. Brodsky, Y. Kornatzky. The *Lyric* Language: Querying Constraint Objects. *Proc. ACM SIGMOD*, pp. 35-46, 1995.

A. Brodsky, V. E. Segal, J. Chen, P. A. Exarkhopoulo, The CCUBE Constraint Object-Oriented Database System. *Proc. ACM SIGMOD*, pp.577-579, 1999.

M. Cai, D. Keshwani, P. Z. Revesz, Parametric Rectangles: A Model for Querying and Animation of Spatiotemporal Databases, *Proc. Seventh Conference on Extending Database Technology*, Springer LNCS 1777, pp. 430-444, 2000.

J. Chomicki, P. Z. Revesz, Constraint-based Interoperability of Spatiotemporal Databases, *Geoinformatica*, vol. 3, no. 3, pp. 211-243, 1999.

G. Edwards, Characterizing and Maintaining Polygons with Fuzzy Boundaries in GIS. *6th Inter. Symp. on Spatial Data Handling*, pp. 223-239, 1994.

M. J. Egenhofer, et al., a Topological Data Model for Spatial Databases, *Springer LNCS 409*, pp. 271-284, 1989.

M. J. Egenhofer, D. M. Mark, Modeling Conceptual Neighborhoods of Topological Line-Region Relations, *Inter. J. of GIS*, vol. 9, no. 5, pp. 555-565, 1995.

M. Erwig, M. Schneider, Vague Regions, *Advances in Spatial Databases*, Springer LNCS 1262, pp.298-320, 1997.

L. D. Floriani, et al., Spatial Queries and Data models, *Spatial Information Theory*, Springer LNCS 716, 1993.

S. Grumbach, P. Rigaux, L. Segoufin, The DEDALE System for Complex Spatial Queries. *Proc. ACM SIGMOD*, pp. 213-224, 1998.

T. Huynh, C. Lassez, J.-L. Lassez. Fourier Algorithm revisited. *Springer LNCS 463*, pp. 117-131, 1990.

P. C. Kanellakis, G. M. Kuper, P. Z. Revesz, Constraint Query Languages. *Journal of Computer and System Sciences*, vol. 51, no. 1, pp. 26-52, 1995.

B. Kuijpers, J. Paredaens, J. Van den Bussche, Lossless Representation of Topological Spatial Data, *Springer LNCS 951,* 1995.

B. Kuijpers, J. Paredaens, L. Vandeurzen, Semantics in Spatial Databases, *Advances in Spatial Databases*, Springer LNCS 1262, pp. 114-135, 1997.

A. Kemper, M. Wallrath, An Analysis of Geometric Modeling in database systems, *Proc. ACM Computer Surveys*, vol. 19, no. 1, 1987.

J.-L. Lassez, Querying Constraints, *Proc. 9th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database systems*, pp.288-298, 1990.

R. Laurini, D. Thompson, *Fundamentals of Spatial Information Systems*, Academic Press, 1992.

J. Paredaens, Spatial Databases, The Final Frontier, *International Conference on Database Theory*, Prague, Springer LNCS 893, 1995.

D. J. Peuquet, Issues Involved in Selecting Appropriate Data Models for Global Databases, *Building Databases for Global Science*, 1988.

F. Preparata, M. Shamos, *Computational Geometry: An Introduction*, Springer, 1985.

P. Z. Revesz, The Evaluation and the Computational Complexity of Datalog Queries of Boolean Constraint Databases, *International Journal of Algebra and Computation*, vol. 8, no. 5, pp. 553-574, 1998.

P. Z. Revesz, R. Chen, P. Kanjamala, Y. Li, Y. Liu, Y. Wang, The MLPQ/GIS Constraint Database System, *Proc. ACM SIGMOD*, 2000.

R. T. Snodgrass, Temporal Databases, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, pp.22-64, Springer, Berlin, 1992.

J. D. Ullman, *Database and Knowledge-Base Systems, volume I*, Computer Science Press, 1988.

M. F. Worboys, A Unified Model for Spatial and Temporal Information, *The Computer Journal*, vol. 37, no. 1, pp.26-34, 1994.

M. F. Worboys, *GIS: A Computing Perspective*, Taylor & Francis, 1995.

M. Zeiler, *Inside ARC/INFO* (revised edition), Onword Press, 1997.