Semantics in Spatial Databases

Bart Kuijpers[1], Jan Paredaens[1], and Luc Vandeurzen[2]

[1] University of Antwerp (UIA), Dept. Math. & Computer Sci.,
Universiteitsplein 1, B-2610 Antwerp, Belgium
Email: {kuijpers, pareda}@uia.ua.ac.be
[2] University of Limburg (LUC), Dept. WNI,
Universitaire Campus, B-3590 Diepenbeek, Belgium
Email: lvdeurze@alpha.luc.ac.be

Abstract. In this paper we discuss two data models for spatial database systems: the *linear data model* and the *topological data model*. Both can be used to model a wide range of applications. The linear data model is particularly suited to model spatial database applications in which exact geometrical information is required and in which this information can be approximated by linear geometrical spatial objects. The topological model on the other hand is suitable for applications in which rather than exact geometrical information the relative position of spatial objects is of importance.
We will specify in each case which types of spatial data and spatial databases are under consideration. A semantics for both data models is formally defined in terms of finite representations of spatial databases in the data models. We also present languages to query spatial databases in both models and briefly investigate their expressiveness.

1 Introduction

The number of computer applications in which database systems are used to store and manage spatial or geometric information has grown rapidly during the last decades. CAD/CAM, VLSI-design, robotics, geographical information systems, and medical imaging are only a few examples of applications that use (sometimes large amounts of) two-dimensional or three-dimensional spatial information.

Initially, spatial database management systems for such applications were build by extending traditional database management systems by introducing rather trivial spatial data types and by extending SQL in an application-dependent way. Current efforts in spatial database systems, however, aim at developing systems that are specifically suited to deal with spatial information, but which are nevertheless application-independent. This latter goal was set out in the following quote of [12]:

> "The challenge for the developers of DBMSs with spatial capabilities lies not so much in providing yet another special-purpose data structure that is marginally faster when used in a particular application, but in defining abstractions and architectures to implement systems that offer generic spatial

data management capabilities and that can be tailored to the requirements of a particular domain".

Apart from the fact that spatial database management systems are now designed as general as possible rather than for one particular application, a number of other and more theoretical concerns dominate the current research in spatial database systems:

- theoretical models are needed which support an elegant combination of spatial (or geometrical) information and non-spatial (or classical) information;
- a formally defined semantics is needed that is closed under set-theoretic, geometric and topological operations and that is defined in terms of a finite representation;
- efficient implementations of operations on n-dimensional spatial objects are desired;
- the connection with visual interfaces and multimedia must be investigated.

In this paper we describe two data models both of which are suited to model a wide class of spatial database applications: the *linear data model* and the *topological data model*. The linear data model is suited to model spatial database applications in which exact geometrical or geographical information on spatial objects is required and in which this information can be approximated by linear geometrical spatial objects. The topological model on the other hand is suitable for applications in which rather than exact geometrical information the relative position of spatial objects is of importance.

The remainder of the paper is structured as follows. In Section 2, we describe the linear data model. First, we look at a number of applications for which this data model is suited: geographical information systems, CAD/CAM and linear decision problems. We analyze the spatial data requirements of these different applications. Next, we formalize the types of data that occur in these applications into the mathematical framework of semi-linear sets in an n-dimensional Euclidean space. We conclude this section by briefly illustrating how the representation scheme of the linear data model can be extended to a language to query spatial databases in the linear data model.

In Section 3, we discuss the topological data model. First, we look at a typical example application of spatial databases in the topological data model and at the information that is characteristically contained in such spatial databases. We formally define what we mean by spatial data and spatial databases in this data model: conceptually, spatial databases consist here of points, curves between these points, and areas formed between these curves in the two-dimensional Euclidean plane. We describe a representation of spatial databases by means of a finite data structure which contains exactly the topological information of the spatial database. We conclude by giving a language to query spatial databases in the topological data model and illustrate the expressiveness of this language.

2 Linear Data Model

In this section, we concentrate on spatial database models for applications that require the knowledge of the exact geometric position in space of the spatial data they manipulate. We focus on linear spatial data, although some applications typically handle more general kinds of spatial data. The representation of a coast line, for instance, requires fractal curves. The restriction to linear data is justified however, for the following reasons:

1. at this moment there are few efficient algorithms known to implement the variety of spatial operations on curved data;
2. linear figures are perfectly suitable to approximate more general data with; and
3. the simplicity of linear data will offer us several desirable properties.

As an introduction, we discuss three different kinds of spatial applications. We briefly analyze the kind of spatial data each application requires to detect all necessary and sufficient data requirements. All these requirements together will motivate the choice of our final spatial data type.

We propose a general and natural data type that, in our opinion, covers all possible data requirements for the intended applications. The representation scheme based on this data type is complete, in the sense that it can deal with every n-dimensional, mathematically definable linear geometric figure. Once the representation of the spatial data is known, we describe formally the *linear data model* and explain its syntax and semantics. We present possible linear database instances for the discussed applications to show the effectiveness of the linear model. Finally, we conclude with extending the representation tool of linear figures to a linear spatial query language.

2.1 Data Requirements

In this section, we analyze the spatial data requirements of three totally different spatial applications. In the first example, we discuss the kind of spatial data required in geographical information systems. Next, we investigate the various spatial objects used in CAD/CAM. Finally, we show how linear (decision) problems can be interpreted as intersections, unions, differences, and projections of n-dimensional linear figures.

Geographical information systems [21] are one of the first applications that demanded a spatial database system. The task of the spatial database is to store a representation of some geographical area, e.g., a road map, together with text and number information as, for instance, speed limits and one way directions.

Representations of geographical areas are typically two-dimensional maps graphically visualizing the relevant information. Figure 1 shows a simplified representation of Belgium. On this map, we have displayed the two most important rivers, some cities, and the three regions of Belgium. Unavoidably, we grap at the polygon data type (to represent two dimensional areas), the line-segment type

(to represent borders and rivers), and the point data type (to represent cities), for this purpose. Complex, not necessarily topologically connected, geographical objects are provided as compositions of these basic data types. The literature mentions various data models based on these primitive types [10, 13, 14, 15, 29, 31].

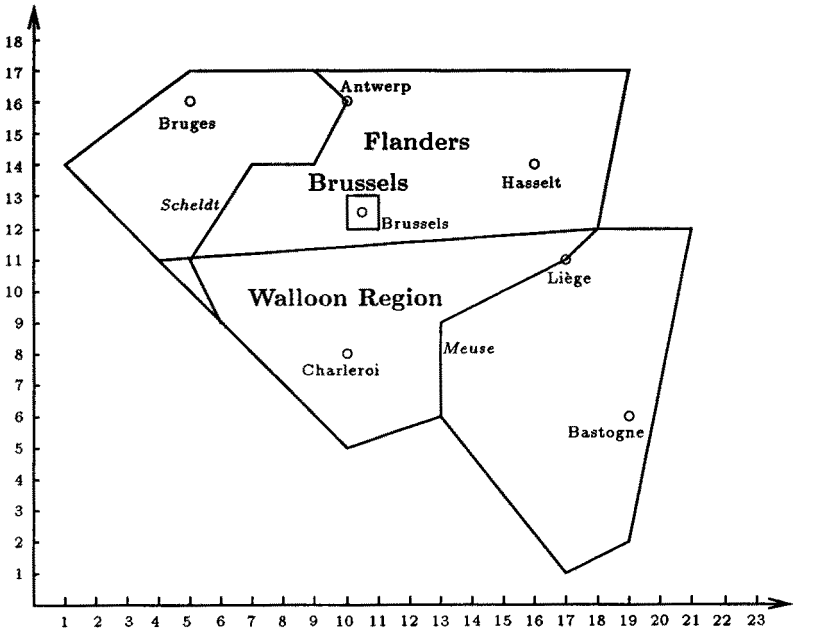


Fig. 1. Spatial information map of Belgium.

The same geographical area can be viewed from various perspectives. We can focus on the road infrastructure, the land use, and weather information, for instance. All these different *thematic layers* of the same area cannot be displayed on the same two-dimensional representation of the area. One particular way to solve this brings all these thematic layers together and considers them as one spatial data object in higher-dimensional space.

In CAD/CAM, the data requirements are somewhat different. Typically, CAD/CAM applications store and manipulate a database containing the representation and features of three-dimensional scenes and solid objects.

Up to now, constructed solid geometry (CSG), together with the boundary representation method, are the most widely propagated representation tools for CAD/CAM objects. In the CSG representation scheme, a geometric object is described as a composition of primitive objects as for instance pyramids and boxes.

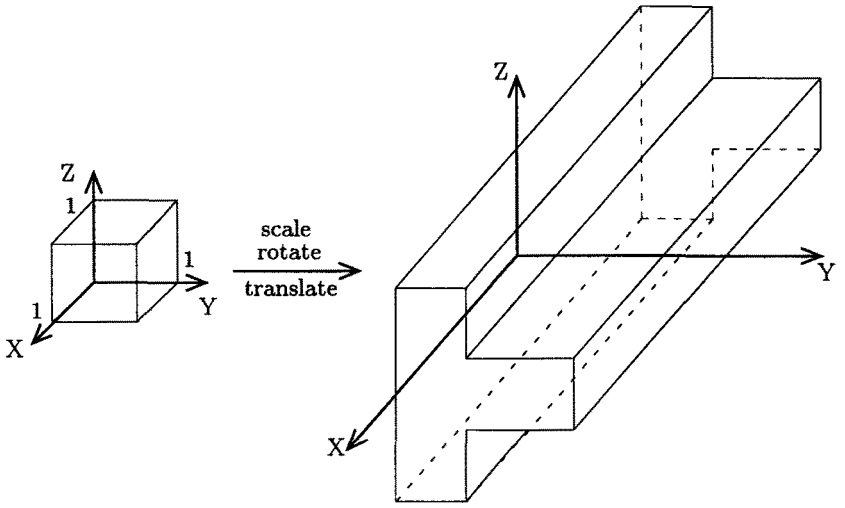


Fig. 2. Composition of two boxes into a T-shape.

The composition is achieved via motial and combinatorial operators. Under motial operators we understand affine transformations as for instance rotation and translation. Examples of combinatorial operators are the set operations union, intersection, and difference. Figure 2 shows an object with a T-shape, created via the CSG methodology.

Although solid objects are always topologically closed, it can be very helpful to use open figures to define them as illustrated by Figure 3. Two boxes are connected using a lump in the one box that precisely fits a notch in the other box. The one box can be represented as the union of the box with a small cube and the other box as the difference of the box with the topologically open interior of the same small cube.

Until now, we were only concerned about the representation of real-world objects. These could all be described using at most three free parameters. The result of linear spatial queries, however, can not always be described with three free variables. Consider, e.g., the query "Compute the position of a couple of shelves in a furnitured room such that desks can still be sit at, and doors and drawers can still be opened." The solution (the coordinates of the mass-centers of the shelves) will define an object in (at least) six-dimensional space. Moreover, that object will not be topologically closed, since open doors and drawers may not touch the shelves. In general, the solution to this kind of linear problems might be an n-dimensional, unbounded and topologically non-closed geometric figure [4].

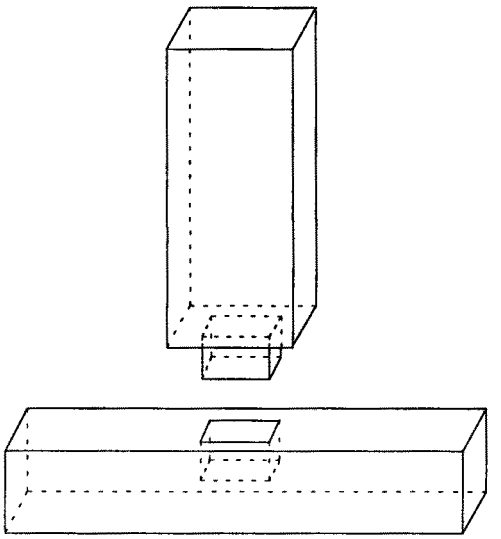


Fig. 3. A connection between two boxes.

2.2 Linear Representation Scheme

In this section, we try to formalize the various data types proposed in the previous section into a mathematical framework. This mathematical framework allows the representation of every *semi-linear set* definable in n-dimensional Euclidean space. Furthermore, the framework is safe in the sense that *only* linear figures can be represented.

As it is more natural to represent a geometrical figure as an enumeration of all its points, the representation we use will be of type "point-set." However, we have to represent our "point-sets" in a finite way to allow physical storage.

More formally, assume a totally ordered infinite set of variables over $\mathbf{R}$ called *real variables*. Define a *linear term* as a linear polynomial with rational coefficients, i.e., of the form $\sum_{i=1}^{k} a_i x_i$, where $x_1, \ldots, x_k$ are real variables and $a_1, \ldots, a_k$ are rational constants. An *atomic linear formula* is a condition of the form $T\ \theta\ a$ where T is a linear term; θ one of the following binary comparison operators $=$, $<$, $>$, $\leq$, $\geq$, and $\neq$; and a a real constant. A *linear formula* is an arbitrary well-formed formula in first-order logic with addition, i.e.,

- atomic linear formulae are linear formulae;
- if φ and ψ are linear formulae, then $\varphi \wedge \psi$; $\varphi \vee \psi$; and $\neg\varphi$ are linear formulae; and
- if x is a real variable and φ is a linear formula in which x is free, then $(\exists x)\varphi$ is a linear formula.

Every linear formula φ with n free variables, $x_1, \ldots, x_n$, defines a point-set

$$\{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\}$$

in n-dimensional Euclidean space $\mathbf{R}^n$. A geometrical figure in $\mathbf{R}^n$ will be called a semi-linear set if there exists a linear formula φ with n free variables such that the point-set defined by φ coincides with the geometrical figure. The representation of a semi-linear set is any linear formula, φ, that defines a point-set equal to the semi-linear set.

It is obvious that this representation scheme is not unique. In fact, any semi-linear set can be represented by an infinite number of linear formulae. On the contrary, the representation scheme is unambiguous. Every linear formula φ defines exactly one semi-linear set.

Assume in the following example that we are working in the three-dimensional Euclidean space $\mathbf{R}^3$.

Example 1. Assume we want to create a prism with height a in the z-direction. The representation of this prism is given by the linear formula

$$\varphi(x, y) \wedge (0 \leq z \leq a),$$

where φ defines the base of the prism in the xy-plane. Any prism with the same shape can now be defined in $\mathbf{R}^3$ as a composition of a translation and a rotation of this prism primitive. In general, any affine transformation can be applied to the prism primitive since affine transformations can be very simply expressed as linear formulae. If we consider, for instance, an affine transformation that does not affect the z-coordinate, the resulting prism may be described in terms of x_{new}, y_{new}, and z_{new} by the linear formula

$$(\exists x)(\exists y)(\varphi(x, y) \wedge (x_{new} = ax + by + t_x) \wedge (y_{new} = cx + dy + t_y)) \wedge (0 \leq z_{new} \leq a)$$

with a, b, c, d, e, t_x, t_y constants defining the affine transformation.

We have explored in this example a very powerful property of the linear representation scheme: the expressibility of affine transformations with linear formulae. This, together with the basic set operations union, intersection, and difference, of which semi-linear sets are closed under [33], allows the definition of linear figures using the "constructed solid geometry" method.

A practical tool to deal with semi-linear sets are *polytopes*. A *polytope* in the Euclidean space (of arbitrary dimension) is defined as the convex hull of a non-empty finite set of points in that space [5, 23, 17]. An *open polytope* is the topological interior of a polytope with respect to the smallest sub-space containing the polytope. It can be proved that bounded semi-linear sets and finite unions of open polytopes are equivalent. [33]

However, since polytopes are necessarily bounded, finite unions of polytopes can only represent bounded linear figures. Therefore, we present another tool characterizing semi-linear point-sets in all their appearances.

Günther [11] defines *polyhedral chains* as a representation scheme for geometric data. A *polyhedral chain* in the Euclidean space (of arbitrary dimension) is defined as a finite sum of *cells* each of which is a finite intersection of half-spaces. It can be shown that semi-linear sets and polyhedral chains are equivalent. [33] This property proves effectively the equivalence of semi-linear sets with mathematically definable linear figures.

The above characterizations allow us to conclude that most spatial data types found in the literature are sub-types of the semi-linear set data type. Güting [13, 14] proposes in his geo-relational algebra the spatial data types *point*, *line*, and *polygon*, which can be seen respectively as zero-, one-, and two-dimensional polytopes.[3] In his spatial data representation model Egenhofer [9] proposes *simplices* as basic objects , which are special kinds of polytopes.

In summary, semi-linear sets constitute a very general and elegant paradigm to represent linear spatial data, which are the kind of spatial data that are most often considered. We believe semi-linear sets have the potential for efficient implementation. Brodsky et al. [4, 3] introduced canonical forms for linear formulae to make efficient implementation of operations on semi-linear sets possible. Lassez et al. [20, 16] have proposed variable elimination algorithms for sets of linear constraints. The alternative characterizations we presented offer the opportunity to use polyhedral chains or polytopes as internal representation for semi-linear sets. Günther [11] has described efficient algorithms to perform set-operations on polyhedral chains. Algorithms to compute efficiently the union or intersection of n-dimensional polytopes are provided by Putnam et al. [28]. Several operations and techniques described in computational geometry, such as plane sweep and divide-and-conquer, can also be used for this purpose [22, 32, 6, 27]. Finally, the notion of semi-linear set is not bounded to any particular dimension.

2.3 The Linear Data Model

The linear spatial database model is based on the relational model, because linear spatial databases require a lot of traditional database capabilities. In particular, if the linear spatial database consists purely of non-spatial flat relations, it degenerates into a traditional database for which the relational model offers a well-accepted representation. Moreover, the relational model, and the relational calculus and algebra as well, have the interesting property of being easily extendible with new data types and operators.

More formally, a *linear spatial database scheme* consists of a finite set of relation names. Each relation name R is of some type $[n, m]$, with n and m integers. A *linear spatial database instance* is a mapping that assigns a linear relation instance to each relation name appearing in the database scheme. A *linear instance* of R, also called a *linear relation*, is a finite set of linear tuples of type $[n, m]$. A *linear tuple* of type $[n, m]$ is straightforwardly defined as a tuple

[3] The polygons considered by Güting are not necessary convex, but can always be decomposed into convex polygons.

of the form

$$(c_1, \ldots, c_n, \varphi(x_1, \ldots, x_m))$$

where $c_1, \ldots, c_n$ are non-spatial values from some domain U and $\varphi(x_1, \ldots, x_m)$ is a linear formula with m free real variables.

The semantics of a linear tuple $t = (c_1, \ldots, c_n, \varphi(x_1, \ldots, x_m))$ of type $[n, m]$ is the possibly infinite subset of $U^n \times \mathbf{R}^m$ defined as the Cartesian product $\{(c_1, \ldots, c_n)\} \times S$, in which $S \subseteq \mathbf{R}^m$ is the semi-linear set $\{(x_1, \ldots, x_m) \mid \varphi(x_1, \ldots, x_m)\}$. This subset of $U^n \times \mathbf{R}^m$ can be interpreted as a possibly infinite $(n + m)$-ary relation, denoted $I(t)$. The semantics of a linear relation, r, denoted $I(r)$, is defined as $I(r) = \bigcup_{t \in r} I(t)$. Finally, the semantics of a linear spatial database, DB, is the set of relations $I(r)$ with r a linear relation of DB.

In the remainder of this section, we give an example spatial database for the map of Belgium depicted in Section 2.1.

2.4 The Linear Spatial Calculus

In this section, we present a calculus-like query language, called FO + linear. The linear calculus, FO + linear, is obtained by adding to the language of linear formulae of Section 2.2 the following:

- a totally ordered infinite set of variables called *non-spatial variables*, disjoint from the set of real variables;
- atomic formulae of the form $v_1 = v_2$, with v_1 and v_2 non-spatial variables;
- atomic formulae of the form $R(v_1, \ldots, v_n; p_1, \ldots, p_m)$, with R a relation name of type $[n, m]$, $v_1, \ldots, v_n$ non-spatial variables, and $p_1, \ldots, p_m$ linear terms; and
- universal and existential quantification of non-spatial variables.

A *query* expressed in FO + linear has the form:

$$\{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\}$$

where $\varphi(x_1, \ldots, x_n)$ is an expression of FO + linear with $x_1, \ldots, x_n$ free variables.

Finally, we shall give some typical example queries, illustrating the expressive power of FO + linear. A precise characterization of the expressive power of FO + linear is still wide open. For a deeper investigation concerning the expressiveness and limitations of FO + linear, we refer to [33].

Example 2. An example of a (very simple) linear spatial query on the database in Example 1 is *"Find all cities that lie on a river and give their names and the names of the rivers they lie on."* This query can be expressed by the following linear calculus expression:

$$\{(c, r) \mid (\exists x)(\exists y)(\text{Cities}(c, x, y) \wedge \text{Rivers}(r, x, y))\}.$$

Cities

Name	*Geometry*
Antwerp	$(x = 10) \wedge (y = 16)$
Bastogne	$(x = 19) \wedge (y = 6)$
Bruges	$(x = 5) \wedge (y = 16)$
Brussels	$(x = 10.5) \wedge (y = 12.5)$
Charleroi	$(x = 10) \wedge (y = 8)$
Hasselt	$(x = 16) \wedge (y = 14)$
Liège	$(x = 17) \wedge (y = 11)$

Rivers

Name	*Geometry*
Meuse	$((y \leq 17) \wedge (5x - y \leq 78) \wedge (y \geq 12)) \vee$
	$((y \leq 12) \wedge (x - y = 6) \wedge (y \geq 11)) \vee$
	$((y \leq 11) \wedge (x - 2y = -5) \wedge (y \geq 9)) \vee$
	$((y \leq 9) \wedge (x = 13) \wedge (y \geq 6))$
Scheldt	$((y \leq 17) \wedge (x + y = 26) \wedge (y \geq 16)) \vee$
	$((y \leq 16) \wedge (2x - y = 4) \wedge (y \geq 14)) \vee$
	$((x \leq 9) \wedge (x \geq 7) \wedge (y = 14)) \vee$
	$((y \leq 14) \wedge (-3x + 2y = 7) \wedge (y \geq 11)) \vee$
	$((y \leq 11) \wedge (2x + y = 21) \wedge (y \geq 9))$

Regions

Name	*Geometry*
Brussels	$(y \leq 13) \wedge (x \leq 11) \wedge (y \geq 12) \wedge (x \geq 10)$
Flanders	$(y \leq 17) \wedge (5x - y \leq 78) \wedge (x - 14y \leq -150) \wedge (x + y \geq 45) \wedge$
	$(3x - 4y \geq -53) \wedge (\neg((y \leq 13) \wedge (x \leq 11) \wedge (y \geq 12) \wedge (x \geq 10)))$
Walloon Region	$((x - 14y \geq -150) \wedge (y \leq 12) \wedge (19x + 7y \leq 375) \wedge (x - 2y \leq 15) \wedge$
	$(5x + 4y \geq 89) \wedge (x \geq 13)) \vee ((-x + 3y \geq 5) \wedge (x + y \geq 45) \wedge$
	$(x - 14y \geq -150) \wedge (x \geq 13))$

Fig. 4. Representation of the spatial database of Belgium shown in Figure 1.

In all the remaining queries, we shall assume the input database consists of one relation S of type $[0, n]$. We shall also use point-variables instead of real variables, e.g., equations such as $(\mathbf{x} < \mathbf{y})$ should be interpreted as $(x_1 < y_1) \wedge \cdots \wedge (x_n < y_n)$. In particular, $(\mathbf{x} \neq \mathbf{0})$ means $(x_1 \neq 0) \wedge \cdots \wedge (x_n \neq 0)$ and not $(x_1 \neq 0) \vee \cdots \vee (x_n \neq 0)$!

Example 3. The following FO + linear-expression decides whether S is bounded:

$$(\exists \mathbf{d})(\forall \mathbf{x})(\forall \mathbf{y})(S(\mathbf{x}) \wedge S(\mathbf{y}) \Rightarrow -\mathbf{d} < \mathbf{y} - \mathbf{x} < \mathbf{d}).$$

Example 4. Several topological properties of a semi-linear set can be computed in FO + linear. For instance, the topological interior of S is computed by the

following FO + linear expression:

$$(\exists \mathbf{d})((\mathbf{d} \neq 0) \wedge (\forall \mathbf{y})(\mathbf{x} - \mathbf{d} < \mathbf{y} < \mathbf{x} + \mathbf{d}) \Rightarrow S(\mathbf{y})).$$

Similarly, the topological closure and topological boundary of S can be computed in FO + linear.

In spite of all this, FO+linear can not be considered as a fully adequate query language. Afrati et al. [2] proved that the query "Does the semi-linear set S lie on a line?" is not expressible in FO + linear. The list of non-expressible FO + linear-queries is further extended in [33] with the predicate $collinear(\mathbf{x}, \mathbf{y}, \mathbf{z})$, which checks if the three points $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are collinear, and the predicate $ch(\mathbf{x_1}, \ldots, \mathbf{x_m}, \mathbf{y})$, which computes all points $\mathbf{y}$ belonging to the convex hull of the set of points $\{\mathbf{x_1}, \ldots, \mathbf{x_m}\}$.

3 The Topological Data Model

In this section, we focus on a spatial database model suited for applications in which knowledge about the relative positions of spatial objects is of importance rather than exact information about the geometric position of the spatial data. We limit the discussion to spatial databases that, conceptually, consist of points, curves between these points and areas formed by these curves in the two-dimensional Euclidean plane. These databases are commonly referred to as spatial databases in the *topological data model*. We will use a railway system as an example application that can be modeled in this way.

We discuss a representation of this type of databases by means of a finite data structure that is primarily geared towards supporting queries that involve only topological properties of the spatial database. This is reflected by the fact that this data structure represents topologically equivalent spatial databases in the same way, but also represents topologically non-equivalent spatial databases in a different way. A representation that satisfies these properties corresponds to an interface which allows the user to concentrate precisely on all the topological properties of the spatial database.

Finally, to conclude this section, we present a language to query spatial databases in the topological data model.

3.1 A Typical Application

Travel information is often provided to the train traveler by means of a spatial database such as the railway map shown in Figure 5. In this map, cities (or their railway stations) are represented by points labeled with the city name and train tracks between the stations are represented by curves connecting these points.

Such spatial database (or map) does not contain exact geographic information. There is no information in this map concerning the exact position of cities or railway stations. It also contains no information to answer a metric query

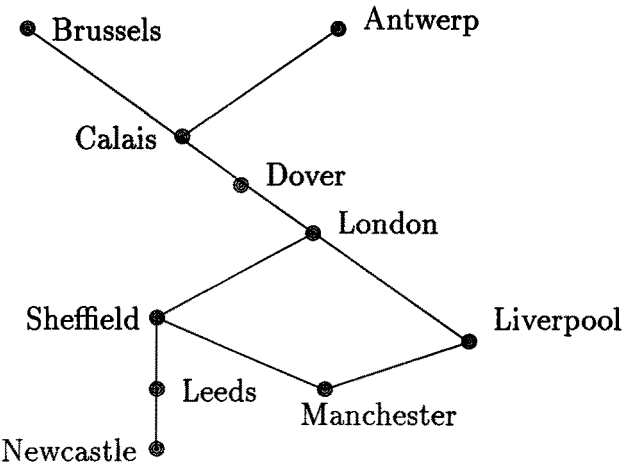


Fig. 5. A railroad map.

such as "What is the distance between London and Calais?" Since the traveler's main concern is not the longitude or latitude of cities he is also not surprised to find London shown right of Brussels in Figure 5 while in reality it is west of Brussels.

On the other hand, with the help of such maps, the traveler can answer his or her typical queries such as "Is there a connection from Brussels to Liverpool with a stop in London?" or "Do I pass Dover before passing Calais when traveling from Antwerp to London?" The class of queries which are of importance to a traveler is formed by those queries involving only properties of the map which are topological in nature. Here, concepts such as adjacency, connectivity, and containment are in the focus. The topological properties of a map are exactly those properties that are shared by any two spatial databases that can be obtained from each other by a topological deformation[4]. As an example of topologically equivalent spatial databases we take the map of Figure 5 and a representation of the same railway system by a conventional map which corresponds more closely to reality. Since in the former map the length of lines is not related to the actual length of the trajectory, length is not a topological property. Since, however, both maps show the same connections, connectedness is a topological property.

In the present section, we elaborate on the idea of topological property in the context of databases consisting of points, curves between these points, and areas formed by these curves [26]. For a survey of other application domains that can be modeled by means of a finite number of points in the plane and curves con-

[4] The notion of topological deformation will be made precise later.

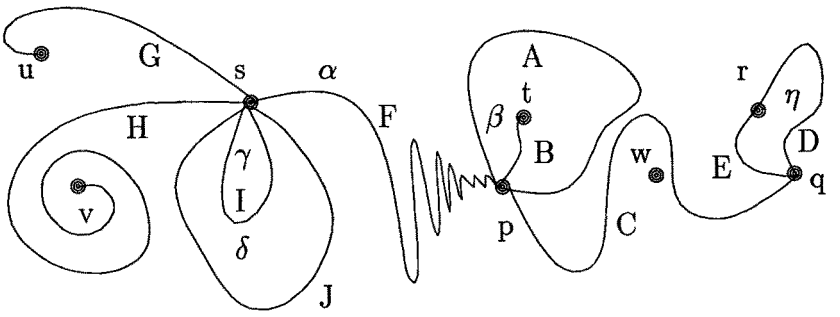


Fig. 6. An example of a spatial database in the topological data model.

necting them, we refer to Chapter 5 of the book by Laurini and Thompson [21] and references therein.

3.2 Spatial Databases in the Topological Data Model

In this section, we exactly define what we mean by spatial data and spatial databases in the topological data model. We work in the Euclidean plane $\mathbf{R}^2$.

Definition 1. 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 $\mathbf{R}^2$. Each curve label is assigned to a distinct non-self-intersecting continuous curve[5] in $\mathbf{R}^2$ 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 remark that this definition allows curves to start and end in the same point, i.e., the database may contain loops. It also may contain multiple curves between two points. It is easily shown that for a database with n curve labels the number of area labels is bounded by $n + 1$.

We apply the following notational convention throughout the remainder of this section: Roman lower-case characters $p, q, \ldots$ denote point labels, Roman capitals $A, B, \ldots$ denote curve labels and Greek characters $\alpha, \beta, \ldots$ are used for area labels.

Figure 6 gives an example of a spatial database. This database has eight points, one of which (w) is isolated. It contains ten curves, three of which $(A, I, \text{and } J)$ are loops. There are five areas, one of which (α) is unbounded.

[5] In topological terms, this is a simple Jordan curve [24].

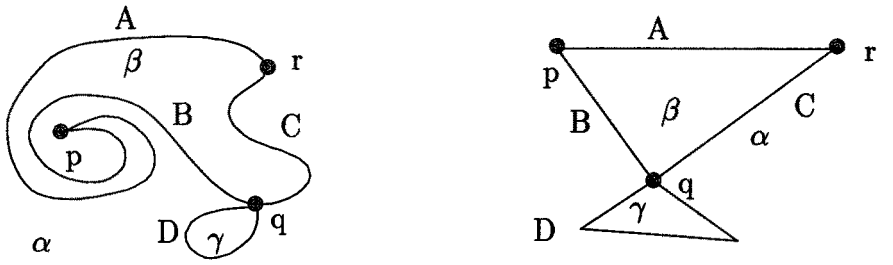


Fig. 7. Two topologically equivalent databases.

3.3 Representations of Spatial Databases in the Topological Data Model

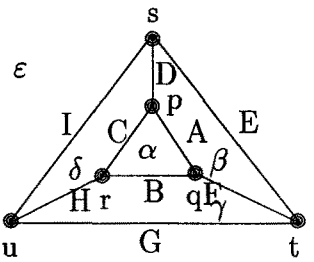
The example application of Section 3.1 shows that in the topological data model the interpretation of spatial data is topological in nature. In other words, spatial databases in the topological data model contain information about topological properties and contain, e.g., no information about metric properties.

When we want to effectively represent spatial databases in the topological data model, and we know that only topological properties are under consideration, it may be desirable to have a representation of a spatial database which is *topologically invariant*, meaning that topologically equivalent databases will be represented in the same way. Ideally, a representation should also be *lossless*, in the sense that two databases that are *not* topologically equivalent will be represented differently.

In order to define these concepts formally, we first have to make the notion of topologically equivalent databases precise. Figure 7 gives an example of two equivalent spatial databases. Intuitively, two spatial databases are topologically equivalent if one can be obtained from the other by a continuous deformation. Mathematically, this continuous deformation is formalized by the notion of *isotopy* [24]. An isotopy h is a continuous series $(h_t \mid 0 \leq t \leq 1)$ of homeomorphisms of the plane. We thus define:

Definition 2. Two spatial databases $\mathcal{D}_1$ and $\mathcal{D}_2$ are called *topologically equivalent* if there exists an isotopy h in $\mathbf{R}^2$ such that $h_0(\mathcal{D}_1) = \mathcal{D}_1$ and $h_1(\mathcal{D}_1) = \mathcal{D}_2$, with the understanding that h respects the labels of points, curves, and areas.

Definition 3. A representation of a spatial database is called *topologically invariant* if any two topologically equivalent spatial databases are represented in the same way. A representation of a spatial database is called *lossless* if any two spatial databases that are not topologically equivalent are distinguished by the representation.



R_1	R_2	R_3	R_4
A p q	A α β	α p A 1	p A α 1
A q p	A β α	α q B 2	p C δ 2
B q r	B α γ	α r C 3	p D β 3
B r q	B γ α	β p D 1	q B α 1
C r p	C α δ	β s E 2	q A β 2
C p r	C δ α	β t F 3	q F γ 3
⋮	⋮	⋮	⋮

Fig. 8. The relations R_1, R_2, R_3 and R_4 illustrated.

Example 5. As an example of a representation of spatial databases in the topological data model we take the representation underlying the cartography system of the US Bureau of the Census [8] (see also [26]). Here a spatial database is represented by means of a classical database consisting of four relations, R_1, R_2, R_3, and R_4, on the labels of points, curves, and areas.

- R_1 gives for every curve its two endpoints;
- R_2 gives for every curve its two adjacent areas;
- R_3 give for each area its border of alternatingly curves and points; and
- R_4 gives for each point its neighborhood of alternatingly curves and areas.

For relation R_3, a clockwise order is agreed upon for outer borders of areas and a counter-clockwise order is used for holes in areas. For relation R_4, a clockwise order is used.

Figure 8 illustrates the relations R_1, R_2, R_3, and R_4 for the depicted spatial database.

The following property shows that this representation can be reduced to the relation R_4 only.

Proposition 4. *R_1, R_2, and R_3 can be deduced from R_4.*

Proof. To deduce R_1 and R_2 we have the following algebra expressions:

$$R_1 = \Pi_{215}(\sigma_{1\neq 5}(\sigma_{2=6}(R_4 \times R_4))) \cup \Pi_{215}(\sigma_{3\neq 7}(\sigma_{1=5}(\sigma_{2=6}(R_4 \times R_4)))),$$

$$R_2 = \Pi_{237}(\sigma_{3\neq 7}(\sigma_{2=6}(R_4 \times R_4))).$$

To deduce R_3 we have $\Pi_{123}(R_3) = \Pi_{312}(R_4)$ and

$$\left.\begin{array}{l}(\alpha pAi) \in R_3 \\ (\alpha qB) \in \Pi_{123}(R_3) \\ (ApQ) \in R_1\end{array}\right\} \Rightarrow (\alpha qBi+1) \in R_3.$$

□

It is more convenient to denote the relation R_4 of Example 5 for each labeled point p as a circular alternating list of curve and area labels rather than as a set of tuples [19]. Actually, this alternating list of labels corresponds to the labels of the curves and areas that an observer, placed in the point p, sees when he makes a full clockwise circular scan of the environment of the point p. A formal definition of this alternating circular list was given in [19] and it was referred to as *the observation of a spatial database from the point p* and denoted as Obs(p). The collection of the observations of a spatial database from each of its points is called *the observation of the spatial database.* The observation of a spatial database satisfies the first requirement of a representation:

Proposition 5. [19] *The observation of a spatial database is an invariant representation of a spatial database.*

Is this representation, on the other hand, lossless? The answer is *no*. Figure 9 contains two spatial databases that can certainly not be obtained from one another by a continuous deformation. So, they are not topologically equivalent but nevertheless represented identically by means of their observation. We have for both the following:

$$\begin{array}{ll}\text{Obs(p)} = (\alpha C\delta D\beta A), & \text{Obs(s)} = (\varepsilon E\beta D\delta I), \\ \text{Obs(q)} = (\alpha A\beta F\gamma B), & \text{Obs(t)} = (\varepsilon G\gamma F\beta E), \\ \text{Obs(r)} = (\alpha B\gamma H\delta C), & \text{Obs(u)} = (\varepsilon I\delta H\gamma G).\end{array}$$

Hence, drastically different spatial databases can be represented in exactly the same way by means of their observations. In [19] it is shown, however, that there is a single cause for this phenomenon: by explicitly marking one of the areas as the unbounded area, losslessness is achieved.

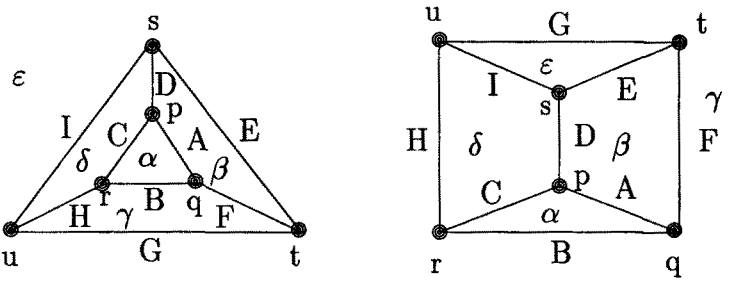


Fig. 9. Two spatial databases that are not topologically equivalent but that have the same observation.

Theorem 6. [19] *A spatial database is losslessly represented by its observation and the the label of the unbounded area.*

Indeed, the two spatial databases of Figure 9 mainly differ in the fact that for the first ε is the label of the unbounded area while it is the label for a bounded one in the second database. Theorem 6 justifies the introduction of a reserved label, α^{∞}, for the unbounded area.

3.4 Querying Spatial Databases in the Topological Data Model

To query spatial databases in the topological data model we introduce the following 3-sorted first-order language $\mathcal{L}_{\mathrm{PCA}}$.

$\mathcal{L}_{\mathrm{PCA}}$ has three sorts of *variables*:

- boldfaced lower-case characters are used for point-variables: $\mathbf{p}, \mathbf{q}, \mathbf{r}, \ldots$;
- boldfaced capitals are used for curve-variables: $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$; and
- boldfaced Greek characters are used for area-variables: $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \ldots$.

The language $\mathcal{L}_{\mathrm{PCA}}$ has one *constant*: α^{∞}, the label of the unbounded area.
A *term* in $\mathcal{L}_{\mathrm{PCA}}$ is

- $\mathbf{p} = \mathbf{q}$ with $\mathbf{p}$ and $\mathbf{q}$ point-variables;
- $\mathbf{A} = \mathbf{B}$ with $\mathbf{A}$ and $\mathbf{B}$ curve-variables;
- $\boldsymbol{\alpha} = \boldsymbol{\beta}$ with $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ area-variables;
- $\boldsymbol{\alpha} = \alpha^{\infty}$ with $\boldsymbol{\alpha}$ an area-variable;
- $\mathbf{A}\boldsymbol{\alpha}\mathbf{B} \subset \mathrm{Obs}(\mathbf{p})$ with $\mathbf{p}$ a point-variable, $\mathbf{A}$ and $\mathbf{B}$ curve-variables and $\boldsymbol{\alpha}$ an area-variable or the area-constant α^{∞}; or
- $\boldsymbol{\alpha} = \mathrm{Obs}(\mathbf{p})$ with $\mathbf{p}$ a point-variable, and $\boldsymbol{\alpha}$ an area-variable or the area-constant α^{∞}.

An *expression* in $\mathcal{L}_{\mathrm{PCA}}$ is

- a term;
- a combination of expressions using $\vee, \wedge, \neg, \rightarrow$; or
- $(\exists \mathbf{p})\varphi$, $(\exists \mathbf{A})\varphi$ and $(\exists \boldsymbol{\alpha})\varphi$ with φ an expression and $\mathbf{p}$ a point-variable, $\mathbf{A}$ a curve variable and $\boldsymbol{\alpha}$ an area-variable.

A *query* expressed in $\mathcal{L}_{\mathrm{PCA}}$ has the form

$$\{(\mathbf{p}_1, \ldots, \mathbf{p}_n, \mathbf{A}_1, \ldots \mathbf{A}_m, \boldsymbol{\alpha}_1, \ldots \boldsymbol{\alpha}_k) \mid \varphi(\mathbf{p}_1, \ldots, \mathbf{p}_n, \mathbf{A}_1, \ldots \mathbf{A}_m, \boldsymbol{\alpha}_1, \ldots \boldsymbol{\alpha}_k)\},$$

where $\varphi(\mathbf{p}_1, \ldots, \mathbf{p}_n, \mathbf{A}_1, \ldots, \mathbf{A}_m, \boldsymbol{\alpha}_1, \ldots \boldsymbol{\alpha}_k)$ is an expression in $\mathcal{L}_{\mathrm{PCA}}$ with free point-variables $\mathbf{p}_1, \ldots, \mathbf{p}_n$, free curve-variables $\mathbf{A}_1, \ldots, \mathbf{A}_m$ and free area-variables $\boldsymbol{\alpha}_1, \ldots \boldsymbol{\alpha}_k$.

$\mathbf{A}\boldsymbol{\alpha}\mathbf{B} \subset \mathrm{Obs}(\mathbf{p})$ means that the observation of the database from $\mathbf{p}$ has the form $(\ldots \mathbf{A}\boldsymbol{\alpha}\mathbf{B} \ldots)$. The term $\boldsymbol{\alpha} = \mathrm{Obs}(\mathbf{p})$ is redundant if a spatial database is not allowed to contain isolated labeled points. The semantics of the expressions in $\mathcal{L}_{\mathrm{PCA}}$ is clear. It should be noticed, however, that the language $\mathcal{L}_{\mathrm{PCA}}$ includes $\mathbf{A}\boldsymbol{\alpha}\mathbf{B} \subset \mathrm{Obs}(\mathbf{p})$ as a term but does not include the construction $\boldsymbol{\alpha}\mathbf{A}\boldsymbol{\beta} \subset \mathrm{Obs}(\mathbf{p})$ as a term. The following property explains why.

Proposition 7. $\boldsymbol{\alpha}\mathbf{A}\boldsymbol{\beta} \subset \mathrm{Obs}(\mathbf{p})$ *if and only if* $(\exists \mathbf{B})\mathbf{B}\boldsymbol{\alpha}\mathbf{A} \subset \mathrm{Obs}(\mathbf{p}) \wedge (\exists \mathbf{C})\mathbf{A}\boldsymbol{\beta}\mathbf{C} \subset \mathrm{Obs}(\mathbf{p}) \wedge (\forall \boldsymbol{\gamma})(\forall \mathbf{D})\mathbf{A}\boldsymbol{\gamma}\mathbf{D} \subset \mathrm{Obs}(\mathbf{p}) \rightarrow (\boldsymbol{\gamma} = \boldsymbol{\beta} \vee \boldsymbol{\gamma} = \boldsymbol{\alpha})$.

Proof. If $\boldsymbol{\alpha}\mathbf{A}\boldsymbol{\beta} \subset \mathrm{Obs}(\mathbf{p})$, then the list $\mathrm{Obs}(\mathbf{p})$ is of the form $\ldots \mathbf{B}\boldsymbol{\alpha}\mathbf{A}\boldsymbol{\beta}\mathbf{C} \ldots$. This proofs the existence of a $\mathbf{B}$ and $\mathbf{C}$ with the desired properties. If we assume $(\exists \boldsymbol{\gamma})(\exists \mathbf{D})\mathbf{A}\boldsymbol{\gamma}\mathbf{D} \subset \mathrm{Obs}(\mathbf{p}) \wedge \neg\boldsymbol{\gamma} = \boldsymbol{\beta} \wedge \neg\boldsymbol{\gamma} = \boldsymbol{\alpha}$, then there are three areas adjacent to $\mathbf{A}$. This is clearly impossible. If $\boldsymbol{\alpha} = \boldsymbol{\beta}$, we have the same conclusion since $(\exists \boldsymbol{\gamma})(\exists \mathbf{D})\mathbf{A}\boldsymbol{\gamma}\mathbf{D} \subset \mathrm{Obs}(\mathbf{p}) \wedge \neg\boldsymbol{\gamma} = \boldsymbol{\alpha}$ is in contradiction with the fact that $\mathbf{A}$ is not a loop in $\mathbf{p}$.

For the other implication, we observe that the observation of the database from $\mathbf{p}$ looks like $\ldots \mathbf{B}\boldsymbol{\alpha}\mathbf{A}\boldsymbol{\beta}\mathbf{C} \ldots$ or like $\ldots \mathbf{B}\boldsymbol{\alpha}\mathbf{A}\boldsymbol{\delta}_1\mathbf{D}_1 \ldots \boldsymbol{\delta}_n\mathbf{A}\boldsymbol{\beta}\mathbf{C} \ldots$. The first case implies $\boldsymbol{\alpha}\mathbf{A}\boldsymbol{\beta} \subset \mathrm{Obs}(\mathbf{p})$. In the second case, $\boldsymbol{\delta}_1 = \boldsymbol{\beta}$ or $\boldsymbol{\delta}_1 = \boldsymbol{\alpha}$. The former of these two again yields the desired result. In the case $\boldsymbol{\delta}_1 = \boldsymbol{\alpha}$, we have $\boldsymbol{\beta} = \boldsymbol{\alpha}$, whence the proposition. □

We note that the condition $(\forall \boldsymbol{\gamma})(\forall \mathbf{D})\mathbf{A}\boldsymbol{\gamma}\mathbf{D} \subset \mathrm{Obs}(\mathbf{p}) \rightarrow (\boldsymbol{\gamma} = \boldsymbol{\beta} \vee \boldsymbol{\gamma} = \boldsymbol{\alpha})$ is necessary in the previous proposition. If we consider a spatial database with one point p with $\mathrm{Obs}(p) = (B\alpha^{\infty}A\gamma A\alpha^{\infty}C\alpha^{\infty})$, we do not have $\alpha^{\infty}A\alpha^{\infty} \subset \mathrm{Obs}(p)$ but still there exists a B and a C such that $B\alpha^{\infty}A \subset \mathrm{Obs}(p)$ and $A\alpha^{\infty}C \subset \mathrm{Obs}(p)$.

On the other hand, we have

Proposition 8. *The constructions* $\mathbf{A}\boldsymbol{\alpha}\mathbf{B} \subset \mathrm{Obs}(\mathbf{p})$ *cannot be expressed in terms of the constructions* $\boldsymbol{\alpha}\mathbf{A}\boldsymbol{\beta} \subset \mathrm{Obs}(\mathbf{p})$.

Proof. For both spatial databases in Figure 10, $\boldsymbol{\alpha}\mathbf{A}\boldsymbol{\alpha} \subset \mathrm{Obs}(\mathbf{p})$, $\boldsymbol{\alpha}\mathbf{B}\boldsymbol{\alpha} \subset \mathrm{Obs}(\mathbf{p})$, and $\boldsymbol{\alpha}\mathbf{C}\boldsymbol{\alpha} \subset \mathrm{Obs}(\mathbf{p})$ hold.

For the first, $\mathbf{A}\boldsymbol{\alpha}\mathbf{B} \subset \mathrm{Obs}(\mathbf{p})$ holds, but for the second $\mathbf{A}\boldsymbol{\alpha}\mathbf{B} \subset \mathrm{Obs}(\mathbf{p})$ does not hold. So, $\mathbf{A}\boldsymbol{\alpha}\mathbf{B} \subset \mathrm{Obs}(\mathbf{p})$ cannot be expressed in terms of the others. □

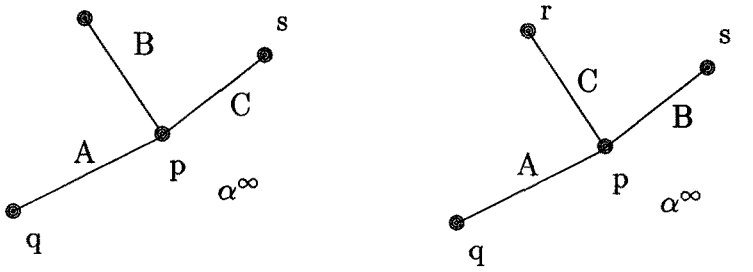


Fig. 10. Two spatial databases with the same area-curve-area information.

As a first example of a query expressed in $\mathcal{L}_{\mathrm{PCA}}$ we take the query "Does the spatial database contain a loop?" This is a Boolean query ($n = 0, m = 0$ and $k = 0$) and it is expressed by the formula

$$(\exists \mathbf{p})(\exists \boldsymbol{\alpha})(\exists \boldsymbol{\beta})(\exists \mathbf{A})\ \neg\boldsymbol{\alpha} = \boldsymbol{\beta} \wedge \boldsymbol{\alpha}\mathbf{A}\boldsymbol{\beta} \subset \mathrm{Obs}(\mathbf{p}) \wedge \boldsymbol{\beta}\mathbf{A}\boldsymbol{\alpha} \subset \mathrm{Obs}(\mathbf{p}),$$

in which we use the abbreviation from Proposition 7.

The relations R_1, R_2, R_3, and R_4 of Section 3.3 can all be expressed in $\mathcal{L}_{\mathrm{PCA}}$:

- $(\mathbf{A}, \mathbf{p}, \mathbf{q}) \in R_1$ if and only if $(\neg \mathbf{p} = \mathbf{q} \wedge (\exists \boldsymbol{\alpha})(\exists \boldsymbol{\beta})\ \boldsymbol{\alpha}\mathbf{A}\boldsymbol{\beta} \subset \mathrm{Obs}(\mathbf{p}) \wedge \boldsymbol{\beta}\mathbf{A}\boldsymbol{\alpha} \subset \mathrm{Obs}(\mathbf{q})) \vee (\mathbf{p} = \mathbf{q} \wedge (\exists \boldsymbol{\alpha})(\exists \boldsymbol{\beta})\ \neg\boldsymbol{\alpha} = \boldsymbol{\beta} \wedge \boldsymbol{\alpha}\mathbf{A}\boldsymbol{\beta} \subset \mathrm{Obs}(\mathbf{p}) \wedge \boldsymbol{\beta}\mathbf{A}\boldsymbol{\alpha} \subset \mathrm{Obs}(\mathbf{p}))$;
- $(\mathbf{A}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \in R_2$ if and only if $(\exists \mathbf{p})\ \boldsymbol{\alpha}\mathbf{A}\boldsymbol{\beta} \subset \mathrm{Obs}(\mathbf{p})$;
- $(\boldsymbol{\alpha}, p, \mathbf{A}, i) \in R_3$ if and only if $(\exists \boldsymbol{\beta})\ \boldsymbol{\beta}\mathbf{A}\boldsymbol{\alpha} \subset \mathrm{Obs}(\mathbf{p})$ and to determine the order of the tuples in R_3: $(\boldsymbol{\alpha}, \mathbf{p}, \mathbf{A}, i) \in R_3$ and $(\boldsymbol{\alpha}, \mathbf{q}, \mathbf{B}, i+1) \in R_3$ if and only if $(\exists \boldsymbol{\beta})\ \boldsymbol{\beta}\mathbf{A}\boldsymbol{\alpha} \subset \mathrm{Obs}(\mathbf{p}) \wedge (\exists \boldsymbol{\beta})\ \boldsymbol{\beta}\mathbf{B}\boldsymbol{\alpha} \subset \mathrm{Obs}(\mathbf{q}) \wedge \mathbf{B}\boldsymbol{\alpha}\mathbf{A} \subset \mathrm{Obs}(\mathbf{q})$;
- $(\mathbf{p}, \mathbf{A}, \boldsymbol{\alpha}, i) \in R_4$ if and only if $(\exists \mathbf{B})\mathbf{A}\boldsymbol{\alpha}\mathbf{B} \subset \mathrm{Obs}(\mathbf{p})$ and to determine the order of the tuples in R_4: $(\mathbf{p}, \mathbf{A}, \boldsymbol{\alpha}, i) \in R_4$ and $(\mathbf{p}, \mathbf{B}, \boldsymbol{\beta}, i+1) \in R_4$ if and only if $\mathbf{A}\boldsymbol{\alpha}\mathbf{B} \subset \mathrm{Obs}(\mathbf{p}) \wedge \boldsymbol{\alpha}\mathbf{B}\boldsymbol{\beta} \subset \mathrm{Obs}(\mathbf{p}) \wedge (\exists \mathbf{C})\ \mathbf{B}\boldsymbol{\beta}\mathbf{C} \subset \mathrm{Obs}(\mathbf{p})$.

An important query in the context of spatial databases in the topological data model is "Is the database connected?" Connectivity for a binary relation is not expressible in the first-order calculus for relational databases (see, e.g., Chapter 17 of [1]). For spatial databases in the topological data model we have

Open Problem: Is connectivity expressible in $\mathcal{L}_{\mathrm{PCA}}$?

Again following the results for the classical case, we see that connectivity *is* expressible in $\mathcal{L}_{\mathrm{PCA}}+$ fixpoint. We define the relation C_i for points $\mathbf{p}$ and $\mathbf{q}$ as follows: $C_i(\mathbf{p}, \mathbf{q})$ is true if and only if there is a path from $\mathbf{p}$ to $\mathbf{q}$ of length at most i in the spatial database. Clearly, C_1 is expressible in $\mathcal{L}_{\mathrm{PCA}}$. Let

$$\varphi(T) = C_1(\mathbf{p}, \mathbf{q}) \vee T(\mathbf{p}, \mathbf{q}) \vee (\exists \mathbf{r})(T(\mathbf{p}, \mathbf{r}) \vee C_1(\mathbf{r}, \mathbf{q})).$$

Then clearly $C_{i+1} = \varphi(C_i)$. For non-inflationary fixpoint semantics, the least fixpoint C of φ can be used to express "Is the database connected?":

$$(\forall \mathbf{p})(\forall \mathbf{q})\ \neg \mathbf{p} = \mathbf{q} \rightarrow C(\mathbf{p}, \mathbf{q}).$$

Acknowledgements

The results in this paper were presented by the three authors at the EDBT Summer School "Advances in Database Technology" in Gubbio, Italy, September 4–8, 1995. The authors are indebted to Marc Gyssens, Jan Van den Bussche, and Dirk Van Gucht for the collaborative research that has lead to several of the results discussed in this paper. The authors also thank Marc Gyssens for carefully proofreading the paper.

References

1. S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*, Addison-Wesley Publishing Company, 1995.
2. F. Afrati, S. Cosmadakis, S. Grumbach, and G. Kuper, "Linear Versus Polynomial Constraints in Database Query Languages," in Proceedings *2nd Int'l Workshop on Principles and Practice of Constraint Programming* (Rosario, WA), A. Borning, ed., *Lecture Notes in Computer Science*, vol. 874, Springer-Verlag, Berlin, 1994, 181–192.
3. A. Brodsky, J. Jaffar, and M.J. Maher, "Toward Practical Constraint Databases," in Proceedings *19th Int'l Conf. on Very Large Databases* (Dublin, Ireland), 1993, 567–580.
4. A. Brodsky and Y. Kornatzky, "The LyriC Language: Querying Constraint Objects," in Proceedings *Post-ILPS'94 Workshop on Constraints and Databases* (Ithaca, NY), 1994.
5. A. Brøndsted, *An Introduction to Convex Polytopes*, *Graduate Texts in Mathematics*, vol. 90, Springer-Verlag, New York, 1983.
6. I. Carlbom, "An Algorithm for Geometric Set Operations Using Cellular Subdivision Techniques," *IEEE Computer Graphics and Applications*, 7:5, 1987, 44–55.
7. A. Chandra and D. Harel, "Computable Queries for Relational Database Systems," *Journal of Computer and System Sciences*, 21:2, 1980, 156–178.
8. J.P. Corbett. Topological Principles of Cartography. Technical Paper No. 48, US Bureau of the Census, Washington, DC, USA: US Government Printing Office, 1979.
9. M.J. Egenhofer, "A Formal Definition of Binary Topological Relationships," in Proceedings *Foundations of Data Organization and Algorithms*, W. Litwin and H.-J. Schek, eds., *Lecture Notes in Computer Science*, vol. 367, Springer-Verlag, Berlin, 1989, 457–472.
10. M.J. Egenhofer, "Why not SQL!", *Int'l J. on Geographical Information Systems*, 6:2, 1992, 71–85.
11. O. Günther, ed., *Efficient Structures for Geometric Data Management*, *Lecture Notes in Computer Science*, vol. 337, Springer-Verlag, Berlin, 1988.
12. O. Günther and A. Buchmann, *Research Issues in Spatial Databases*, in Sigmod Record, vol. 19, 4, 61-68, 1990.

13. R.H. Güting, "Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems," in *Advances in Database Technology—EDBT '88*, Proceedings *Int'l Conf. on Extending Database Technology* (Venice, Italy), J.W. Schmidt, S. Ceri, and M. Missikoff, eds., *Lecture Notes in Computer Science*, vol. 303, Soringer-Verlag, Berlin, 1988, 506–527.
14. R.H. Güting, "Gral: An Extensible Relational Database System for Geometric Applications," in Proceedings *15th Int'l Conf. on Very Large Databases* (Amsterdam, the Netherlands), 1989, 33–34.
15. R.H. Güting, "An Introduction to Spatial Database Systems," *VLDB-Journal*, 3:4, 1994, 357–399.
16. T. Huynh, C. Lassez, and J.-L. Lassez. Fourier Algorithm Revisited. In Proceedings *2nd Int'l Conf. on Algebraic an Logic Programming*, H. Kirchner and W. Wechler, eds. *Lecture Notes in Computer Science*, vol. 463. Springer Verlag, Berlin, 1990, 117–131.
17. P.J. Kelly and M.L. Weiss. *Geometry and Convexity: a Study in Mathematical Methods*, J. Wiley and Sons, New York, 1979.
18. P.C. Kanellakis, G.M. Kuper and P.Z. Revesz, "Constraint Query Languages," *Journal of Computer and System Sciences*, to appear, also in Proceedings *9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Nashville, TN), 1990, 299–313.
19. B. Kuijpers, J. Paredaens, and J. Van den Bussche "Lossless representation of topological spatial data," in Proceedings *4th Symposium on Advances in Spatial Databases*, M. J. Egenhofer and J. R. Herring, eds., *Lecture Notes in Computer Science*, vol. 951. Springer Verlag, Berlin, 1995, 1–13.
20. J.-L. Lassez, "Querying Constraints," in Proceedings *9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Nashville, TN), 1990, 288–298.
21. R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems.* The A.P.I.C. Series, 37, Academic Press, 1992.
22. M. Liebling and A. Prodon, "Algorithmic Geometry," in *Scientific Visualization and Graphics Simulation*, D. Thalmann, ed., J. Wiley and Sons. 14–25.
23. P. McMullen and G.C. Shephard, *Convex Polytopes and the Upper Bound Conjecture*, University Press, Cambridge, 1971.
24. E.E. Moise. *Geometric Topology in Dimensions 2 and 3. Graduate Texts in Mathematics*, vol. 47, Springer-Verlag, 1977.
25. J. Paredaens, J. Van den Bussche, and D. Van Gucht, "Towards a Theory of Spatial Database Queries," in Proceedings *13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Minneapolis, MN), 1994. 279–288.
26. J. Paredaens. Spatial Databases. The Final Frontier. *Database Theory - ICDT '95, Lecture Notes in Computer Science*, vol. 893, 14–32, Springer-Verlag, 1995.
27. F.P. Preparata and D.E. Muller. "Finding the Intersection of n Half-Spaces in Time $O(nlogn)$," *Theoretical Computer Science*, 8, 1979, 45–55.
28. L.K. Putnam and P.A. Subrahmanyam, "Boolean Operations on n-Dimensional Objects," *IEEE Computer Graphics and Applications*, 6:6, 1986, 43–51.
29. N. Roussopoulos, C. Faloutsos, and T. Sellis, "An Efficient Pictorial Database System for PSQL," *IEEE Transactions on Software Engineering*, 14:5, 1988, 639–650.
30. W. Schwabhauser, W. Szmielew, and A. Tarski. *Metamathematische Methoden in der Geometrie*, Springer-Verlag, Berlin, 1983.
31. P. Svensson and Z. Huang, "Geo-Sal: A Query Language for Spatial Data Analysis," in Proceedings *2nd Symposium on Advances in Spatial Databases*, O. Günther

and H.-J. Schek, eds. *Lecture Notes in Computer Science*, vol. 525. Springer-Verlag, Berlin, 1991, 119–140.

32. B. Tilove, "Set Membership Classification: a Unified Approach to Geometric Intersection Problems," *IEEE Transactions on Computers*, C-29:10, 1980, 874–883.
33. L. Vandeurzen, M. Gyssens, and D. Van Gucht, "On the Desirability and Limitations of Linear Spatial Query Languages," in Proceedings *4th Symposium on Advances in Spatial Databases*, M. J. Egenhofer and J. R. Herring, eds., *Lecture Notes in Computer Science*, vol. 951. Springer Verlag, Berlin, 1995, 14-28.