Data Models and Query Languages for Spatial Databases

Jan Paredaens and Bart Kuijpers
University of Antwerp (UIA)*

Abstract

The main purpose of this paper is to investigate the characteristics that distinguish spatial databases systems from traditional ones. Hereto, we give an overview of some well-known data models and query languages of spatial database systems. We also investigate the concept of genericity, as introduced by Chandra and Harel for classical databases [6], for spatial databases. Paredaens, Van den Bussche and Van Gucht [34] have shown that the concept of genericity breaks up in a hierarchy of genericity classes. In this respect, we classify data models and query languages according to the type of generic operations they are designed to support [33].

1 Introduction

During the last decade the number of applications in which database systems are used to represent spatial information in, mainly, the two-dimensional plane or the three-dimensional space has steadily increased. Applications that rely on spatial databases can be found in CAD-CAM, VLSI-design, robotics, historical databases, geographical information systems, architectural sciences, visual perception and autonomous navigation, tracking, environmental protection and medical imaging [4, 18, 1, 16, 40].

Güting describes in [22] a spatial database as a database system that offers spatial data types in its data model and query language and supports such data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial join.

A spatial database system should

*Contact address: Dept. Math. & Computer Sci., University of Antwerp (UIA), Universiteitsplein 1, B-2610 Antwerp, Belgium. E-mail: {pareda, kuijpers}@uia.ua.ac.be.

- contain an elegant framework to combine geometric and thematic (i.e. classical attribute-value pairs) information;
- be as general as possible and not designed for one particular area of applications;
- have a formally defined semantics that is closed under set theoretic, geometric and topological operations and that is defined in terms of finite representations;
- be independent of a particular database management system (DBMS) but co-operative with any DBMS;
- use efficient implementation techniques, especially for the operations on n-dimensional objects;
- have an up-to-date visual user interface and a gateway to multimedia.

The first spatial database systems that were built do not support all these criteria. Their main issue was to extend existing database management systems by introducing rather trivial spatial data types and extending SQL in an ad hoc way. There was, and still is, a lack to understand the more fundamental issues that are involved in geometric data types. In traditional database systems there is a clear understanding of which part of the information retrieval is handled by the DBMS, and which part is handled by the application software. A projection, for instance, is part of the query language, the calculation of a standard deviation, is not. How do we make such a distinction in the case of a spatial database? Which is the range of data models that can be acceptable candidates for a spatial DBMS and which are the typical features and properties of spatial data manipulation languages?

We believe that, after a few years of experience, it is now time to investigate deeper those characteristics that distinguish the spatial database systems from the traditional ones. As the authors say in [32]:

> "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".

Looking to the literature, we see that only a few researchers have paid attention to the basic issues of spatial data. One of the reasons for this lack of interest could be the vast know-how that is needed in three totally different areas of science:

- databases and information systems [28];
- geography and cartography [7];
- abstract and computational geometry [36]; and
- topology [2].

Contribution of each of them is inevitable in a solid theory of spatial information systems. They are the basis of the so-called *geomatic data models* (see also [33]).

In Section 2, we focus on typical geomatic operations and we classify different kinds of spatial queries. For this purpose we take a closer look at the well-known concept of genericity, introduced by Chandra and Harel for classical databases [6].

Genericity distinguishes between operations that are independent from the representation of the information and operations whose result is intrinsically influenced by the representation of the information. Paredaens, Van den Bussche and Van Gucht [34] have shown that for spatial databases this concept depends on the particular geometric properties that are considered to be of importance. Spatial database applications can be classified according to the geometric concepts that are involved in the interpretation of the spatial information. Metric queries, for instance, that deal with distances form a broader class than the one consisting out of queries in which only topological properties are involved. We take a closer look at a taxonomy of genericity-classes that was introduced in [34].

In Section 3, we give an overview of a number of geomatic data models and discuss some of their properties. All these models are intensional, i.e., they give a finite representation of the mostly infinite and even non-enumerable set of points of the spatial objects that are described by the database. In the *polynomial model* [27, 34], spatial databases are described by means of polynomial equalities and inequalities. In this model, the relational calculus, extended with comparisons between polynomials, is used as a query language. We will show that this natural query language does not give rise to sound and complete languages for any of the genericity-classes discussed in Section 2. There are, however, point-based languages known

that do [25]. In the *topological data model* only some kind of topological (and as a consequence, partial) information is handled without dealing with the exact position and form of the spatial objects. In this context, we look at the relation between topology-genericity and a type of genericity introduced by Egenhofer [11, 12]. In the *raster model*, an object is given by a finite number of its points. These points are equally distributed following an easy geometric pattern, which is normally a square. In the *spaghetti model*, an object is intentionally deduced from its contour, which is a polyline. The *Peano model* also uses a finite number of object-points, but here these points are distributed non-uniformly, according to the form of the object. This distribution method is based on the well-known Peano curve. Although most discussions and illustrations in Sections 2 and 3 will be limited to two-dimensional space, generalizations to higher dimensions are in most cases straightforward.

2 Data Models, Query Languages and Genericity

In this section, we present a general model for spatial data and spatial queries. We use this model to discuss the notion of genericity. For spatial queries, this notion was first studied by Paredaens, Van den Bussche and Van Gucht [34]. Throughout this section, we follow the definitions of [34].

We assume that we have a set of *relation names*, where each relation name R has a *type* $\tau(R) = [n, m]$ which is a pair of natural numbers. A relation R of type $[n, m]$ has the form

$$(TA_1, \ldots, TA_n; SA),$$

where TA_i are called the *thematic attributes* of R and where SA is called the *spatial* or *geometric attribute* of R. This definition of a relation can easily be adapted to allow more spatial attributes. This generalization is not essential however. An *instance* of a relation of type $[n, m]$ on the extensional level is a (possibly infinite) subset of $\mathbf{U}^n \times \mathbf{R}^m$, where $\mathbf{U}$ is a countably infinite domain of thematic atomic values and $\mathbf{R}$ is the set of the real numbers. The number n determines the dimension of the thematic part of the relation. The number m, on the other hand, determines the dimension of the space in which the geometrical part is embedded. If $n = 0$ we call the relation *purely spatial*. If, on the other hand, $m = 0$ we have a classical relation and we call it a *flat* relation.

We call a finite set of relation names a *database scheme*. An extensional instance of a database scheme is a set of extensional instances of its relations. For a database scheme $\mathcal{S}$ we denote the set of its extensional instances by $e(\mathcal{S})$. We say that a relation of type $[n, m]$ is of *dimension* k if m is a multiple of k. We say that a database scheme is of *dimension* k if all its relations are. By this definition, relations and databases clearly can be of more than one dimension. The choice may depend on the spatial database application at hand.

Without bothering about which finite representation is used to represent infinite instances of relations, we can now define what we mean by a query and by a generic query.

Definition 1 For two database schemes $\mathcal{S}_{\text{in}}$ and $\mathcal{S}_{\text{out}}$, we define an *(extensional) query* of signature $\mathcal{S}_{\text{in}} \to \mathcal{S}_{\text{out}}$ to be a partial function from $e(\mathcal{S}_{\text{in}})$ to $e(\mathcal{S}_{\text{out}})$ which is generic on the thematic part, i.e., which is invariant under every permutation of the set $\mathbf{U}$. ■

We remark that the condition on the thematic part of the relations is the classical genericity condition of Chandra and Harel [6]. They investigate in the context of the relational model of databases which queries are "reasonable". They characterize this class of queries by means of the concept of genericity which since then has taken a central position in the theory of computable queries in databases. Indeed, a query in the relational model is called computable if and only if it is a Turing-computable function on a representation of the database, that is also generic. By a generic function Q we mean here a function whose result is invariant on any permutation ϕ of the universal domain of the database. This means that the value of a generic query is independent of the internal representation of the data. Formally, on the other hand, this means that if D and D' are two relational databases such that $D' = \phi(D)$, for some isomorphism ϕ, then $Q(D') = \phi(Q(D))$. It is well-known that all the operations in the relational algebra are generic (except for the selections of the form $\sigma_{A=c}$ and $\sigma_{A>B}$). On the other hand, the algebra is not complete for the generic functions, since the transitive closure, for instance, is a generic function that cannot be expressed by the algebra.

A weaker kind of genericity, the C-genericity, where C is a subset of the universal domain, has been introduced: a function Q is called C-generic if its result is invariant on any permutation ϕ for which $\phi(c) = c$ for all $c \in C$. Obviously $\sigma_{A=c}$ is $\{c\}$-generic.

In the case of databases that are not finite but recursive the concept of genericity has been studied in [26].

For spatial databases, however, the definition of genericity depends on the particular kind of geometry in which the spatial information is to be interpreted [34]. This interpretation is determined by the spatial database application and the type of queries that are considered to be of importance in the application. The concept of a "reasonable query" depends on the geometrical properties that are used in the application. In some applications, like temporal databases dealing with points on the real line, relative positions among the different spatial figures in the database are essential. For these applications it is appropriate to consider only translations of the space as isomorphisms and not for instance reflections. Geographical applications, on the other hand, deal with areas and complete geometrical information. For these applications isometries (distance-preserving transformations) are suited as isomorphisms. These examples show that it is useful to define genericity of spatial database queries as a function of some group of geometric transformations.

We will now formalize the definition of genericity relative to the general model for spatial data and spatial queries that was given above.

Definition 2 [34] Let $\mathcal{S}_{\text{in}}$ and $\mathcal{S}_{\text{out}}$ be schemes of dimension k. And let G be a group of transformations of $\mathbf{R}^k$. A query Q of signature $\mathcal{S}_{\text{in}} \to \mathcal{S}_{\text{out}}$ is *G-generic* if for every transformation $g \in G$ and any two instances I_1 and I_2 of $\mathcal{S}_{\text{in}}$ the fact $g(I_1) = I_2$ implies $g(Q(I_1)) = Q(I_2)$. ■

In the remainder of this section we illustrate the notion of geomatic genericity for a number of naturally occurring transformation groups [34] with an example of a database in $\mathbf{R}^2$ that contains information about persons and the places they live. By place we mean the geomatic information of their home. This relation has the name Lives. On the extensional level, a tuple in Lives has the form $(n; x, y)$, where n is the name of a person, and (x, y) are the co-ordinates of the place he lives. The database also contains geomatic information about a region in the relation Reg of the form (x, y). We have $\tau(\mathsf{Lives}) = [1, 2]$ and $\tau(\mathsf{Reg}) = [0, 2]$. In the top left corner of Figure 1 an instance of the $\{\mathsf{Lives}, \mathsf{Region}\}$ scheme is given. The remaining five figures show the answers to some of the following seven queries.

A first simple kind of query is

Q_1: Give the home of the persons that live in Reg.

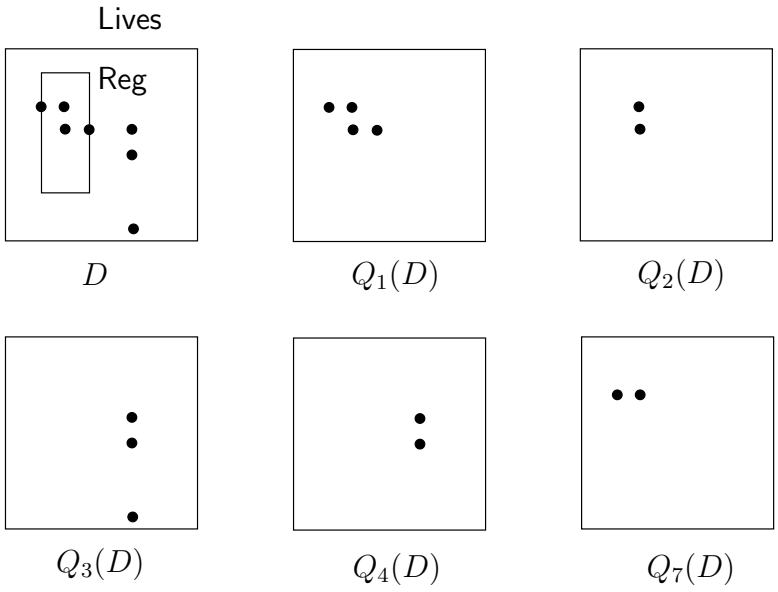


Figure 1: An instance of the scheme {Lives, Region} is given at the top left followed by some answers to queries

This is indeed a query since it clearly commutes with any permutation of the domain of thematic atomic values. The result of this query is purely spatial of type [0,2]. But, more importantly in this context, this query also commutes with every permutation of the points of $\mathbf{R}^2$. Indeed, if we denote by $\mathbf{P}$ the set of all permutations of $\mathbf{R}^2$ and if $\phi \in \mathbf{P}$, then we have that

$$Q_1 \cdot \phi = \phi \cdot Q_1.$$

The result of the query Q_1 is independent of any geometric, geomatic or topological property of the region. We mean that, if we first select the wanted locations and then apply some permutation ϕ, we get the same result as first doing the permutation ϕ and then ask the question. We say that Q_1 is *permutation-generic* or $\mathbf{P}$*-generic*. This type of genericity is identical to the genericity in the relational model, as explained above.

Consider now a second question

Q_2: Give the home of the persons that live in the interior of Reg.

By interior we mean here the topological interior. It is clear that this query does not commute with every permutation. Indeed, take only one person living in $(0,0)$, the region being the closed disk with center $(0,0)$ and radius 1 and consider the permutation $\phi \in \mathbf{P}$ that interchanges $(0,0)$ and $(2,2)$ and fixes the rest. Obviously $(2,2)$ is not in $(Q_2 \cdot \phi)(D)$, while it is in $(\phi \cdot Q_2)(D)$, where D is the database.

However, Q_2 commutes with every homeomorphism (i.e. continuous permutation). We will denote the group of all homeomorphisms by $\mathbf{H}$. Therefore we call the query Q_2 *topology-generic* or $\mathbf{H}$-*generic* for short.

An affinity is a permutation that preserves collinearity. The group $\mathbf{A}$ of affinities gives rise to the *affinity-generic* or $\mathbf{A}$-*generic* queries, being those queries that commute with the affinities. The query Q_3 is obviously an affinity-generic query:

> Q_3: Give all the triples of homes of those persons that live outside Reg and live on one line.

The query Q_3 does not commute with an arbitrary homeomorphism since collinearity is not preserved by all homeomorphisms.

We can prove that Q_4,

> Q_4: Give the homes of those persons that live outside Reg but closest to it.

is not affinity-generic, but it commutes with all elements of $\mathbf{S}$, the set of similarities. A similarity is a permutation that preserves the angles. Such queries are called the *similarity-generic* or just $\mathbf{S}$-*generic* queries.

The next question

> Q_5: Give the pairs of homes of those persons that live exactly 10 from each other.

This query is obviously not similarity-generic but it commutes with every isometry. An isometry is a permutation that preserves the distances. We write $\mathbf{I}$ for the group of all isometries. Therefore we call this query *isometry-generic* or $\mathbf{I}$-*generic*.

The next question

> Q_6: Give the triples (p_1, p_2, p_3) of locations of homes such that the smallest angle between the vectors $p_1\vec{p}_2$ and $p_2\vec{p}_3$ is clockwise.

Q_6 is a *direct-isometry-generic* or $\mathbf{D}$-*generic* query with a flat result. These queries commute with the direct-isometries (elements of $\mathbf{D}$), those permutations that preserve the distance and the clockwise-orientation.

Finally, we have the *translation-generic* or $\mathbf{T}$-*generic* queries. They commute with the elements of the group $\mathbf{T}$ of translations.

> Q_7: Give the homes of those persons that have no person living north of them

illustrates the $\mathbf{T}$-generic queries.

We could even further limit the number of permutations and consider for instance the group consisting of one single element: the identity. Clearly, for this group every query is generic.

To summarize the taxonomy of genericity-classes given, in [34], we give the chain (strict inclusions) of seven transformation groups to which they correspond:

$$\{1\} \subset \mathbf{T} \subset \mathbf{D} \subset \mathbf{I} \subset \mathbf{S} \subset \mathbf{A} \subset \mathbf{H} \subset \mathbf{P}.$$

This taxonomy may be considered natural but it certainly is not comprehensive for what concerns interesting transformation groups. Other interesting groups that could be considered in this context are:

- those permutations that fix the first quadrant;
- those permutations that fix the x-axis and the y-axis;
- those permutations that fix the points with equal thematic information;
- the reflection on a given axis;
- the horizontal translations;
- the homothecies $(x, y) \mapsto (ax, ay)$ with $(0, 0)$ as center;

The following lemma is easy to prove and often useful.

Lemma 1 *G is a subgroup of G' if and only if all G'-generic queries are G-generic.* ■

The genericity classes above form a hierarchy in the sense of the following property. The example queries Q_i $(i = 1, \ldots, 7)$ prove the strictness.

Proposition 1 *[34]* **P**-*genericity* $\Rightarrow$ **H**-*genericity* $\Rightarrow$ **A**-*genericity* $\Rightarrow$ **S**-*genericity* $\Rightarrow$ **I**-*genericity* $\Rightarrow$ **D**-*genericity* $\Rightarrow$ **T**-*genericity* $\Rightarrow$ $\{1\}$-*genericity. These implications are strict.*

The great challenge is to find for each group G, a sound and complete language for the class of G-generic queries. The challenge becomes still harder if we add the geomatic model as an additional parameter to the problem. We will meet this challenge partially in the next section.

3 Geomatic Data Models

Four main characteristics distinguish geomatic data models from the classic ones:

- geomatic data models are used to represent information about the n-dimensional real space $\mathbf{R}^n$. The latter is an infinite, even a non-enumerable, set of points. So, in general, the information we want to represent is infinite in nature. This prevents us to use extensional data models. Different intentional techniques are used in geomatic data models for representing this infinite information. The data model that will be used in a particular geomatic database depends on the operations that have to be defined and on the efficiency needs of the implementation;

- the intentional aspect of geomatic data models has a particular influence on the operations, those that are defined within the model as well as those that are user-defined [20]: the data model has to be closed for all the operations. Since geomatic applications ask typically for a rich set of operations, the above property can be hard to be fulfilled;

- the information that is represented in geomatic applications has usually not the elegant geometric properties of a man-created structure, but is mostly the visualization of a symmetry-less phenomenon proceeded from nature. This induces that the intentional information is mostly vast and that we need particular algorithms for implementing the data structure. These algorithms are based on algebraic, geometrical and topological properties;

- As discussed in Section 2, the notion of genericity, seems to break up in a hierarchy for geomatic data models.

In this section we give an overview of some well-known geomatic data models: the *polynomial model*, the *topological data model*, the *raster model*, the *spaghetti model* and the *Peano model*.

3.1 The Polynomial Model and Complete Languages

3.1.1 The Polynomial Model

A natural approach to spatial data is to consider as a geometrical figure any figure that is definable in elementary geometry, i.e., first-order logic over the real numbers. This is the approach of the so-called *polynomial model* [27, 34] in which exactly this class of figures, referred to as *semi-algebraic sets* [3] in real algebraic geometry, is considered. In the polynomial model the information is stored in relations, each of which contains a finite number of tuples. As mentioned before, the spatial properties of an n-dimensional spatial object are described by a semi-algebraic set of the form

$$\{(x_1, \ldots, x_n) \mid x_1, \ldots, x_n \in \mathbf{R} \wedge \Phi(x_1, \ldots, x_n)\}$$

where $\Phi(x_1, \ldots, x_n)$ is a *semi-algebraic formula*. This class of formulas, also referred to as $\mathsf{FO} + \mathsf{poly}$ is formally defined by

Definition 3 The *alphabet* of $\mathsf{FO} + \mathsf{poly}$ consists of an infinite, enumerable, set of variables $x_1, x_2, \ldots$, which we will call *real variables*, the symbols $\neg, \vee, \wedge, (,), \exists, \forall$, a set of constant symbols, the function symbols $+, \cdot$, and the basic predicate symbols $=$, $<$, $\leq$, $>$ and $\geq$.

- *Real terms* are polynomials in real variables with rational coefficients;
- A *real formula* is an arbitrary well-formed first-order formula built from
  - atomic formulas, are of the form $P \,\Theta\, Q$, where P and Q are real terms and Θ is one of $=$, $<$, $\leq$, $>$ or $\geq$;
  - boolean operators; and
  - quantifications $(\exists x)$ or $(\forall x)$ of real variables.

The number of free variables of a formula is called its *arity*. ■

We note that there are only two such sets for spatial dimension 0: $\{(\,)\}$, and $\emptyset$ (or *True* and *False*).

Remark that the coefficients of the polynomials are rational numbers in order to be representable.

$$\{(x_1, x_2) \mid (81 < (x_1 - 6)^2 + (x_2 - 5)^2) \wedge ((x_1 - 6)^2 + (x_2 - 5)^2 < 121)\}$$

and

$$\{(x_1, x_2) \mid \exists x_3 \exists x_4((x_3 - 6)^2 + (x_4 - 5)^2) = 100 \wedge ((x_3 - x_1)^2 + (x_4 - x_2)^2 < 1)\}$$

are two semi-algebraic sets that both represent Figure 2. This example shows that one figure may have more than one representation in this model. However, it is, in general, decidable whether two semi-algebraic sets are equivalent, in the sense that they represent the same geometrical object [34]. This is a consequence of a well-known result by Tarski [41] that says that every semi-algebraic formula is equivalent to a semi-algebraic formula without quantifiers. This effective quantifier elimination makes many properties of semi-algebraic decidable.

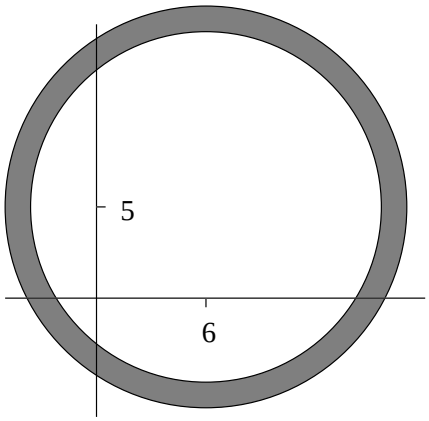


Figure 2: An example of a semi-algebraic set in $\mathbf{R}^2$

Consider now the spatial relation

$$R = (TA_1, \ldots, TA_k; SA)$$

of type $[k, n]$. An *intensional tuple* of this relation has the form $(a_1, \ldots, a_k; \Phi)$, with the Φ being a semi-algebraic formula and the a_i's being elements of $\mathbf{U}$, the countably infinite domain of atomic values.

The extension of this tuple is the set

$$\{(a_1, \ldots, a_k; x_1, \ldots, x_n) \mid \Phi(x_1, \ldots, x_n)\}.$$

In this context, we also use the terms *intensional relation*, *extensional tuple* and *extensional relation* with the obvious meanings. For a database scheme $\mathcal{S}$ we denote the set of its intensional relation instances $i(\mathcal{S})$. We now define what a query is in the context of the polynomial model.

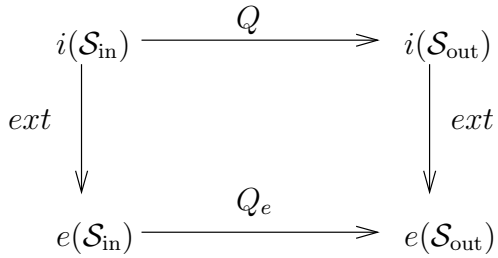
Definition 4 In the polynomial model we call for two database schemes $\mathcal{S}_{\text{in}}$ and $\mathcal{S}_{\text{out}}$

- an *extensional query* of type $\mathcal{S}_{\text{in}} \to \mathcal{S}_{\text{out}}$ any extensional query which can be expressed by

  $$\{(x_1, \ldots, x_n, y_1, \ldots, y_{k'}) \mid \Psi(x_1, \ldots, x_n, y_1, \ldots, y_{k'})\}$$

  where $\Psi(x_1, \ldots, x_n, y_1, \ldots, y_{k'})$ is a formula that contains boolean operators, existential and universal quantifiers and whose terms have the form $(x'_1, \ldots, x'_n, y'_1, \ldots, y'_k) \in R$ where R is a relation name from $\mathcal{S}_{\text{in}}$, $y'_i = y'_j$, $y'_i = c$ or are comparisons, using $<, \leq, >, \geq, =, \neq$, between polynomials whose coefficients are rational numbers. The variables $x_1, \ldots, x_n$ are the free thematic variables of Ψ and $y_1, \ldots, y_{k'}$ are the free spatial variables of Ψ. A formula such as Ψ is called a *formula of the polynomial calculus*;

- an *intensional query* of type $\mathcal{S}_{\text{in}} \to \mathcal{S}_{\text{out}}$ any partial recursive function $Q : i(\mathcal{S}_{\text{in}}) \to i(\mathcal{S}_{\text{out}})$ for which there is an extensional query Q_e of the same type such that the following diagram is commutative for the function *ext* that maps intensional relations to the corresponding extensional ones.



To illustrate this, we turn to the seven queries Q_i $(i = 1, \ldots, 7)$ of Section 2. They can all be expressed in the polynomial model. We give some examples: Q_1 is expressible as

$$\{(x, y) \mid (\exists n)((n; x, y) \in \mathsf{Lives} \wedge (x, y) \in \mathsf{Reg})\}.$$

Q_5 is expressible as

$$\begin{aligned}\{(x, y, x', y') \mid \quad & (\exists n)(\exists n')((n; x, y) \in \mathsf{Lives} \\ & \wedge (n'; x', y') \in \mathsf{Lives} \wedge (x - x')^2 + (y - y')^2 = 100)\}.\end{aligned}$$

Q_7 is of result type $[0, 2]$ and is expressible as

$$\begin{aligned}\{(x, y) \mid \quad & (\exists n)((n; x, y) \in \mathsf{Lives} \\ & \wedge (\forall n')(\forall x')(\forall y')(n'; x', y') \in \mathsf{Lives} \rightarrow y' \leq y)\}.\end{aligned}$$

3.1.2 Sound and Complete Languages

We now turn to genericity of queries expressible in $\mathsf{FO} + \mathsf{poly}$. The examples Q_i $(i = 1, \ldots, 7)$ show that all the genericity-classes of the taxonomy in Section 2 can be handled by $\mathsf{FO} + \mathsf{poly}$.

In [34] the following negative result was stated and proved:

> *for the non-trivial permutation groups G of the taxonomy of Section 2 G-genericity of an* $\mathsf{FO} + \mathsf{poly}$ *query is undecidable.*

Because of this undecidability result it is important to find languages which capture exactly the G-generic $\mathsf{FO} + \mathsf{poly}$ queries for the most relevant groups G.

Gyssens, Van den Bussche and Van Gucht [25] have solved this problem for the groups G in the chain

$$\mathbf{T} \subset \mathbf{D} \subset \mathbf{I} \subset \mathbf{S} \subset \mathbf{A}.$$

All the transformations in these groups are also expressible by means of polynomial equalities and inequalities. In fact, they give for each of these groups G, first-order point languages $\mathsf{FO}(\Pi_G)$, parameterized by sets Π_G of point predicates that are complete for the $\mathsf{FO} + \mathsf{poly}$ queries that are G-generic.

For example, they show that $\Pi_{\mathbf{A}}$ can be taken to be the set with the single point predicate $\mathbf{between}(p, q, r)$ (meaning that p lies on the line segment between q and r). If the predicates $\mathbf{equidistance}(p, q, r, s)$ (meaning that

the distance between p and q equals the distance between r and s) and **unitdistance**(p, q) (meaning that the distance between p and q equals 1) are added to this set, $\Pi_{\mathbf{I}}$ is obtained.

For the group of homeomorphisms $\mathbf{H}$, we refer to the next section.

3.2 Topology-Genericity and the Topological Data Model

Unlike most of the other transformation groups discussed before, the group of topological transformations cannot be parameterized by a finite number of real numbers. A translation of the plane $\mathbf{R}^2$, e.g., is completely known when its image on only one point is known. It is therefore completely determined by a couple (a, b) of real numbers (the image of the point $(0, 0)$ under this translation). This is not the case for homeomorphisms. Indeed, it can be easily shown that for any set of n points there exist two homeomorphisms that have the same image on this set but a different image on some point not belonging to this set.

In this subsection we will show that by limiting the spatial databases, on the other hand, we can achieve an "effective representation" in a topological sense. We present a data structure that gives an invariant and lossless representation used to represent spatial databases in $\mathbf{R}^2$ that are in the *topological data model*. Further on, we investigate the connection between topology-genericity and a form of genericity introduced by Egenhofer [11, 12].

3.2.1 The Topological Data Model

In this section, we are interested in queries that only involve properties of the database that are topological in nature. In this class of queries, concepts such as adjacency, connectivity, and containment are important. Queries like "Is there a highway connecting Brussels to Paris?" or "Give all countries in Europe adjacent to the Atlantic" are typical in this respect. Characteristic of topological properties is that they do not distinguish between two databases that can be obtained from each other by a topological deformation. We will call such databases *topologically equivalent*. [1]

[1] Here we divert from the usual definition of topological equivalence in terms of homeomorphisms and we call two databases $\mathcal{D}_1$ and $\mathcal{D}_2$ topologically equivalent if and only if there exists an *isotopy* $h = (h_t \mid 0 \leq t \leq 1)$ such that $h_0(\mathcal{D}_1) = \mathcal{D}_1$ and $h_1(\mathcal{D}_1) = \mathcal{D}_2$.

In applications in which only topological properties are under consideration, it may be desirable to be able to work with a representation of the database which is *topologically invariant*, meaning that two topologically equivalent databases will be represented identically. Ideally, as we will illustrate later, a representation should also be *lossless*, in the sense that two databases that are *not* topologically equivalent will be represented differently.

We elaborate on the idea of topological property in the context of a limited class of spatial databases consisting of points in the plane $\mathbf{R}^2$, lines between these points, and areas formed by these lines. This model is commonly referred to as the *topological data model* [21, 37]. An example application is a subway or railroad map in which only relative positions of spatial objects such as stations and tracks are depicted without, for instance, taking the actual length of the trajectory into account. A survey of application domains that can be modeled in this manner was given by Laurini and Thompson [37].

We first formalize what a spatial database is in this context.

Definition 5 A *database in the topological data model* consists of a finite set of labeled points, a finite set of labeled lines and a finite set of labeled areas. Each point name is assigned to a distinct point in the plane $\mathbf{R}^2$. Each line name is assigned to a distinct injective and continuous curve (a simple Jordan curve) [31] in the plane that starts and ends in a labeled point and does not contain any other labeled points except these. Distinct curves only meet in labeled points. Each area label is assigned to a distinct area formed by the labeled lines. ■

Figure 3 gives an example of such a database.

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

The Census Bureau of the United States introduced this data model in 1979 [8] to model topological information on what they called zero-cells (points), one-cells (lines) and two-cells (areas) [37]. Here, the information in the two-dimensional plane is described by a number of cells, each having an identifier. Furthermore, the following "classical" relations R_1, R_2, R_3 and R_4 are given:

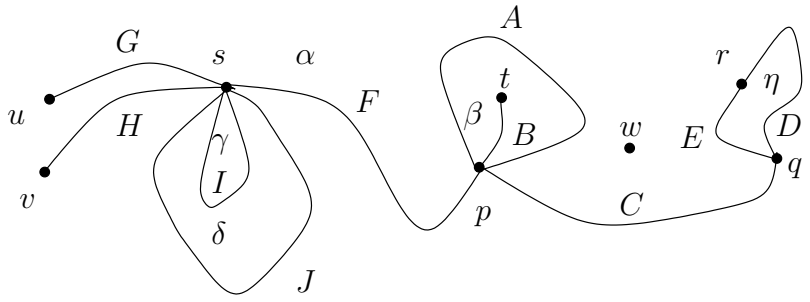


Figure 3: An example of a database in the topological data model

- R_1: every one-cell has two zero-cells (indicating that every line has exactly two endpoints);
- R_2: every one-cell has two two-cells (indicating that every line is the border between exactly two areas);
- R_3: every two-cell is surrounded by a (ordered) cycle of one-cells and zero-cells (indicating the border of an area);
- R_4: every zero-cell is surrounded by a (ordered) cycle of one-cells and two-cells (indicating the neighborhood of a point).

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. To settle the planarity of the model there is the additional condition that all intersections of one-cells are zero-cells and and all intersections of two-cells are one-cells.

Figure 4 illustrates the relations R_1, R_2, R_3 and R_4 (to save space) only partially for a triangular shaped database.

R_1		
A	p	q
A	q	p
B	q	r
B	r	q
C	r	p
C	p	r
	$\ldots$	

R_2		
A	α	β
A	β	α
B	α	γ
B	γ	α
C	α	δ
C	δ	α
	$\ldots$	

R_3			
α	p	A	1
α	q	B	2
α	r	C	3
β	p	D	1
β	s	E	2
β	t	F	3
	$\ldots$		

R_4			
p	A	α	1
p	C	δ	2
p	D	β	3
q	B	α	1
q	A	β	2
q	F	γ	3
	$\ldots$		

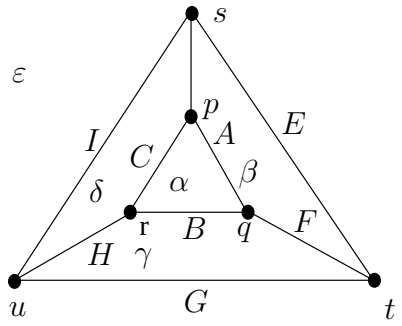


Figure 4: The relations R_1, R_2, R_3 and R_4 illustrated

Neither the exact position of the cells, nor their length or surface is not given by this representation. Only the interrelation of the position of the lines, the areas and the points, i.e. its topological properties are determined. If we are only interested in this kind of information, this model does not contain any redundancy.

Clearly, a topological deformation of the database of Figure 4 is also represented by the same relations. Hence, this representation of a database is *invariant*. Figure 5 gives a database in the topological data model that gives rise to the same instances for R_1, R_2, R_3 and R_4 as the one of Figure 4. The database depicted in Figure 4 is topologically seen, however, different from the one depicted in Figure 5. This not shows that this representation is *not lossless*.

Kuijpers, Paredaens and Van den Bussche give in [29] a representation of databases in the topological data model that is both invariant and lossless. The notion of an observation of a database from one of its labeled points is at the basis of this representation. For each labeled point in a spatial

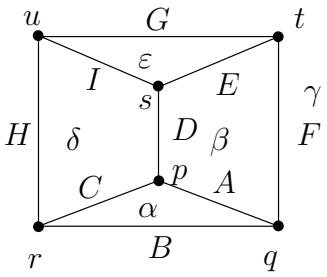


Figure 5: A database with the same values for R_1, R_2, R_3 and R_4 as the one of Figure 4.

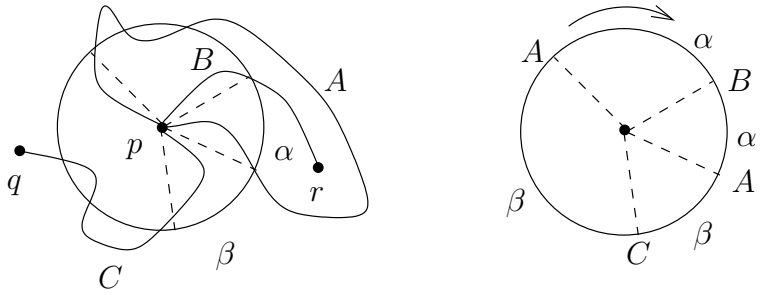


Figure 6: An observation of a database from one of its points

database, we make a circular alternating list of area names and line names corresponding respectively to the areas and lines that an observer, placed in the named point, sees when he makes one clockwise full turn and scans the environment of the point. This is illustrated in Fig. 6. There, the alternating list for the point with name p is $(\alpha\ B\ \alpha\ A\ \beta\ C\ \beta\ A)$. A point which is isolated from the remainder of the database gives rise to an observation consisting of one single area label. We denote the observation from p by $\mathrm{Obs}(p)$.

In [29] it is shown that the concept of observation is well-defined. Clearly, also a representation of a database by means of observations is invariant. The examples of Figure 5 again show that observations alone are not enough to achieve losslessness. In [29]

Definition 6 For a given database $\mathcal{D}$, we call the data structure $(\mathcal{P}, \mathcal{L}, \mathcal{A}, \alpha^\infty, \text{Obs}())$ the *PLA-structure of* $\mathcal{D}$ if $\mathcal{P}$ is the set of point labels of $\mathcal{D}$, $\mathcal{L}$ is the set of line labels of $\mathcal{D}$, $\mathcal{A}$ is the set of area labels of $\mathcal{D}$, α^∞ is the name of the unbounded area of $\mathcal{D}$, and Obs() is a function that associates with each element p of $\mathcal{P}$, the observation of $\mathcal{D}$ from p.

Clearly, the information in the relations R_1, R_2, R_3, R_4 can be reconstructed from the observations. Furthermore, it can be shown that additional information about the identity of the unbounded area suffices to get a lossless representation:

Theorem 1 *[29] The PLA-structure is an invariant and lossless representation of a database.*

This result is also relevant from a user interface point of view: a topologically invariant, lossless representation of the database corresponds to an interface which allows the user to concentrate only on the topological aspects of the spatial data, and on all of them, if he so desires. A query involving distances (for instance query Q_5 of Section 2) is not supported by the PLA-structure.

This result implies that a query language which operates on PLA-structures gives a sound and complete language for topology-genericity. [2] For instance, we can easily define a three-sorted (with point, line and area variables) first-order calculus on PLA-structures with some constructions on observation lists. Such a language was described and studied in [30]. If we restrict the spatial databases to be databases in the topological data model we can in this way, at least in part, fill the gap of the last subsection.

3.2.2 Topology-Genericity and Related Genericity-Classes

Here, we discuss topology-generic queries in $\mathbf{R}^2$ with its natural topology [2]. We limit our discussion to the case that the database only contains one relation, which has one thematic attribute that contains the object identifiers and one two-dimensional spatial attribute. In this case a database can be viewed as a collection of two-dimensional objects.

Topology-generic queries are those that only use topological properties. Here are some examples:

- give the connected objects of the relation;

[2] To be precise: isotopy-genericity.

- give the objects that overlap with some other objects;
- give the topological boundary of each object;
- give the objects whose interior includes some other objects;
- give the objects that have some holes;
- give the pairs of objects that have the same number of holes;
- give the objects that have more than five holes.

Queries that use the shape of an object, its surface, a distance, the gravity center, etc., are not topology-generic:

- give all the triangles in the relation;
- give the points of the objects in the relation that are closer than 10 to (0,0);
- translate the the objects of the relation 5 in the direction of the x-axis;
- rotate the objects of the relation 90 degrees around $(0, 0)$;
- give the gravity center of each object.

Max Egenhofer has studied the topological issues that are related with geomatic data types very extensively [14, 10, 11, 12, 15, 13].

In [11, 12] the *9-intersection model* is given for topological relations in $\mathbf{R}^2$. This model is based on the overlapping properties of the interiors (A°, B°), the complements $(\overline{A}, \overline{B})$ and the boundaries $(\partial A, \partial B)$ of two two-dimensional objects A and B. Although this model only contains limited topological information it has shown to be very useful and efficient in practice.

There are $3 \times 3 = 9$ combinations for the intersections of $A^\circ, \overline{A}, \partial A$ and $B^\circ, \overline{B}, \partial B$. They are represented in the following matrix.

$$R(A, B) = \begin{pmatrix} \partial A \cap \partial B & \partial A \cap B^\circ & \partial A \cap \overline{B} \\ A^\circ \cap \partial B & A \cap B^\circ & A^\circ \cap \overline{B} \\ \overline{A} \cap \partial B & \overline{A} \cap B^\circ & \overline{A} \cap \overline{B} \end{pmatrix}$$

Each of these intersections can be empty ($\emptyset$) or not empty ($\neg\emptyset$). Hence there are $2^9 = 512$ different possible values of such matrices, i.e., 512 possible different topological relationships between two objects. Exactly one of these

topological relationships holds between any two objects in $\mathbf{R}^2$. However only 8 of these relations can be realized for two-dimensional objects. These eight possibilities are illustrated in Figure 7.

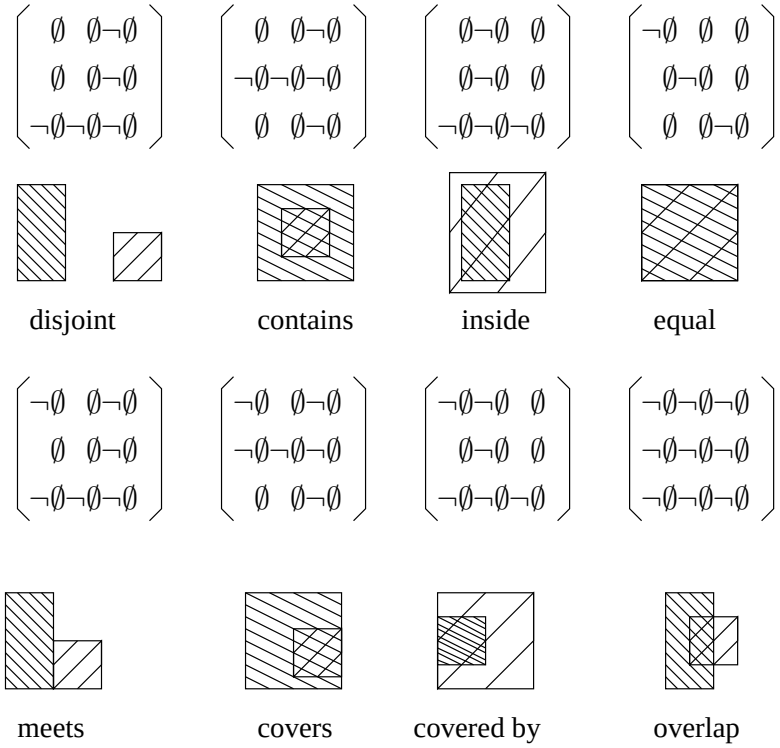


Figure 7: The 8 possible relations between two spatial objects in $\mathbf{R}^2$

Clearly, when we consider three objects, we need a $3 \times 3 \times 3$-matrix. For n objects, we need an n-dimensional $3 \times 3 \times \cdots \times 3$-matrix.

The matrix of Egenhofer allows us to define concepts that are analogical to topological equivalence and topologic genericity.

Definition 7 Let R and R' be two relations.
• We call now the contents of the relations R and R' *Egenhofer equivalent* (notation $R \sim_E R'$) if they contain the same object identifiers and if their topological relationship is given by the same matrix.

• Let us call a query Q *Egenhofer-generic* if and only if $R \sim_E R'$ implies $Q(R) \sim_E Q(R')$. ■

For instance, the relation that contains the tuples $(A; \{(x, y) \mid y > 0\})$ and $(B; \{(x, y) \mid y < 0\})$, the relation that contains the tuples $(C; \{(x, y) \mid y > 0\})$ and $(D; \{(x, y) \mid y \leq 0\})$ and the relation that contains the tuples $(E; \{(x, y) \mid y \geq 0\})$ and $(F; \{(x, y) \mid y \leq 0\})$ are not Egenhofer-equivalent, even if $A = C = E$ and $B = D = F$, since $\overline{A} \cap \partial B = \partial A \cap \overline{B} = \overline{C} \cap \partial D = 0$ and $\partial C \cap \overline{D} = \overline{E} \cap \partial F = \partial E \cap \overline{F} = \emptyset$.

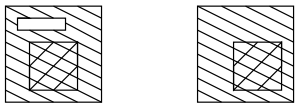
As an example of an Egenhofer-generic query we give: "Give the interior of the objects in the relation."

The Egenhofer matrix only contains information about a limited number of topological properties of a spatial object: its interior, complement and boundary. In practice, this information will often be sufficient to describe the relations between a number of spatial objects. In theory, of course, such limited information does not capture all the topological relations between objects. Therefore it is interesting to study what the relations are between Egenhofer-equivalence (resp. -genericity) and topological equivalence (resp. -genericity).

The following results show that Egenhofer-equivalence is a weaker property than topological equivalence.

Proposition 2 *Topological equivalence implies Egenhofer-equivalence. The comverse does not hold.*

Proof. For two topologically equivalent relations, there exists a homeomorphism $h : \mathbf{R}^2 \rightarrow \mathbf{R}^2$ that maps one onto the other. Since for any spatial object A, we have $\overline{h(A)} = h(\overline{A})$, $h(A)^\circ = h(A^\circ)$ and $\partial(h(A)) = h(\partial A)$, they are also Egenhofer-equivalent. The following two relations have the same Egenhofer matrix, but are not equivalent in a topological sense.



R_1 R_2 (*)

■

The following property shows that topology-genericity and Ehenhofer-genericity are not related.

Proposition 3 *Topology-genericity does not imply Egenhofer-genericity. Egenhofer-genericity does not imply topology-genericity.*

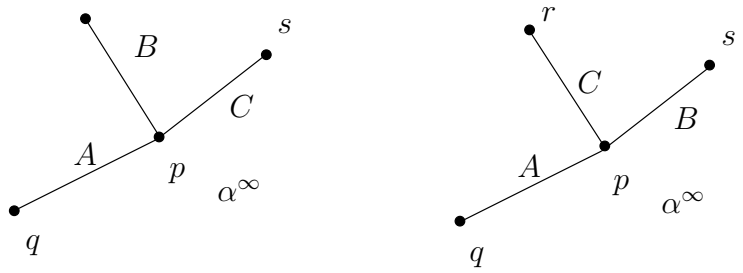
Proof. The query "Give the objects in the relation that have exactly one hole" is topology-generic since the number of holes of an object in the plane is a topological property of the object. Applied to the relations R_1 and R_2 of $(*)$ in the previous proof, this query gives one object in the first case and the empty relation in the second. Nevertheless, R_1 and R_2 are Egenhofer-equivalent. So, the query is not Egenhofer-generic. The other case follows from the observation that relations with only one spatial object with the same identifier are always Egenhofer-equivalent. ■

To end this section we look at the relationship between Egenhofer-equivalence and PLA-equivalence for databases in the topological data model. By the latter notion we mean equivalence in terms of PLA-structure. Here again we observe that Egenhofer-equivalence is stricktly weaker.

Proposition 4 *Two databases in the topological data model that have the same PLA-structure are Egenhofer-equivalent. The converse does not hold.*

Proof. From the previous subsection we now that two databases with the same PLA-structure are isotopic and thus topologically equivalent. By Proposition 2 they are also equivalent in a topological sense.

Consider the two databases R_1 and R_2 depicted in the following figure. They have the same Egenhofer matrix but a different PLA-structure. So, the Egenhofer matrix does not contain enough information to reconstruct the order of the lines A, B and C.



■

3.3 Some Other Models

Here, we briefly describe three models that were developed towards practice. The focus here is more on practical issues such as implementation efficiency

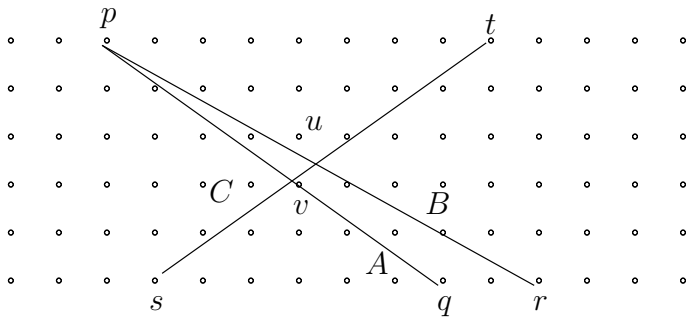


Figure 8: A database in the raster model

and less on theoretical concerns such as genericity. None of these models is geared towards a particular kind of genericity. In the *raster model* an object is given by a finite number of its points. The geomatic information is intentionally represented. In the *spaghetti model*, information in n-dimensional space is intentionally deduced from its contour, which is a m-dimensional hyperspace ($m < n$). The *Peano model* tries to intentionally represent every point of an object. It combines the use of space-filling curves and quadtree into a framework that supports an efficient implementation of a number of natural operations.

3.3.1 The Raster Model

In the *raster model* [24, 23, 39] the geomatic information is intentionally represented by a finite number of raster points with the semantics that the infinite number of points in the environment of a raster point p have the same properties as p. The raster points are uniformly distributed over the object that is considered. Although this is a quite natural definition, some problems can arise since the environment of a raster point is not always homogeneous as to the relevant properties. In Figure 8 for instance most of the points in the environment of raster point u are not on the line B, (an infinite number of) others are. This fundamental problem gives rise to a number of anomalies in the natural extension of classical operations to the raster model. We conclude with an overview of other possible models.

We can illustrate this in Figure 8. In the raster model a line is rep-

resented by two of its raster points. Theoretically this gives a problem to represent those lines that do not contain any raster point, but in most applications these lines can be approximated sufficiently by representable lines. In Figure 8, the line A is represented by q and p, B by r and p and C by s and t. Clearly the two lines B and C intersect although their real intersection point is not a raster point. The model could define the model intersection point to be the nearest raster point to the real intersection point, here this is v, but then we deduce in the model that v is on line B. Since v is also on line A it is the raster intersection point of A and B, but p is also the raster intersection point of A and B, since it is a raster point as well as the real intersection point of A and B. This kind of problems is handled in [24, 23].

In the context of these approximation problems it is easy to show that the raster model does not support translation-generic queries. We will illustrate this with Figure 8 in which we assume that the raster points correspond to points $(a, b) \in \mathbf{R}^2$ with integer co-ordinates a and b. The query Q that gives the intersection point of the lines B and C does not commute with the translation τ determined by $\tau(0, 0) = (0, 0.3)$. The translation of the result (v) is again v. The query on the translated databases returns u on the other hand. It is easily verified, on the other hand, that the raster model does support queries that are *integer-translation-generic* (or maybe more appropriately *raster-generic*), i.e., that commute with any translation τ for which $\tau(0, 0) \in \mathbf{Z}^2$.

3.3.2 The Spaghetti Model

In the *spaghetti model*, or *vectorization model* [37], the information in an n-dimensional space is represented only using m-dimensional hyperspaces, with $m < n$. This means that in a three-dimensional space we only consider polyhedra, the boundaries of which contain planar facets, segments and points.

In the two-dimensional case we only consider polygons, the boundaries of which contain segments and points. More concretely we use here:

1. points;
2. graphs, whose data structure is a finite set of pairs of points;
3. polylines, whose data structure is a finite sequence of points;
4. polygons that are represented by non-selfintersecting closed polylines;

5. complex polygons, that can contain holes, which are again complex polygons (up to a finite level);

6. objects are sets of polygons, points or graphs.

Hence we have

$$
\begin{array}{lcl}
\text{POINT} & = & (x : \text{REAL}, y : \text{REAL}, a : \text{DOM}) \\
\text{GRAPH} & = & (g : \{(p_1 : \text{POINT}, p_2 : \text{POINT})\}, a : \text{DOM}) \\
\text{P\_LINE} & = & (p : \text{POINT}*, a : \text{DOM}) \\
\text{POLYGON} & = & \text{P\_LINE} \\
\text{COMPLEX\_} & & \\
\text{POLYGON} & = & \text{POLYGON} \mid (p : \text{POLYGON}, \\
 & & \text{holes} : \{\text{COMPLEX\_POLYGON}\}, a : \text{DOM}) \\
\text{OBJECT} & = & \{(id : \text{KEY}, p : \text{POINT}, a : \text{DOM})\} \mid \\
 & & \{(id : \text{KEY}, g : \text{GRAPH}, a : \text{DOM})\} \mid \\
 & & \{(id : \text{KEY}, c : \text{COMPLEX\_POLYGON}, a : \text{DOM})\}\}.
\end{array}
$$

"a : DOM" denotes one or more thematic attributes. Figure 9 gives an example in a GIS-application showing one object that contains some complex polygons (representing pieces of land and houses) one object that contains a graph (representing streets).

The reason why this model is so popular is the existence of very efficient algorithms for detecting properties in this model [39]. We are thinking of verification algorithms to detect whether two polygons overlap, whether a point lies in a polygon or on a segment, whether two segments intersect, whether a polyline selfintersects, whether a polygon is contained in another one, etc. Furthermore, the model is simple to use and offers in most applications a sufficient approximation to reality. Also the technique of "recursive holes" has some unexpected applications, such as the representation of a third dimension in a two-dimensional space (like isobars and isotherms).

The spaghetti model is a widely discussed model [5, 17] and numerous query languages [9, 38, 21, 13] and algebras [19, 20] can be found in the literature that are based on a spaghetti-like model.

3.3.3 The Peano Model

In contrast to the spaghetti model, some models try intentionally to represent every point of an object, in the same sense as the raster model. Such models are sometimes called the "pizza" models. The *Peano model* is such

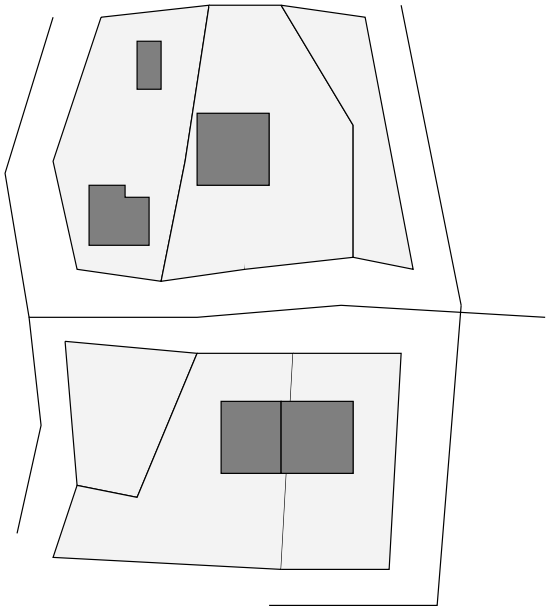


Figure 9: An example of a database in the spaghetti model

a model. It is an elegant marriage of two different well-known techniques: space-filling curves and quad-trees.

A space-filling curve is an infinite sequence of curves, whose limit fills a given square. Two of the most well-known examples are the Peano curve [35] and the Hilbert curve. The Peano curves is illustrated in Figure 10.

The quad-tree [39] is a generalization of the binary tree in which every node is a leave or has four children. Each node represents a quadrant of its parent. The quad-tree is an appropriate implementation technique for the two-dimensional information of a square. It is illustrated in Figure 11.

$[i, j]$ indicates the subsquare with edge-length j, whose left bottom unit subsquare is labeled by i. Remark that the unit subsquares are labeled according to the second Peano curve of Figure 10. There is a very handsome technique to calculate the label of a unit square:

$$\text{label}(u) = \text{interleave}(u_x, u_y)$$

with

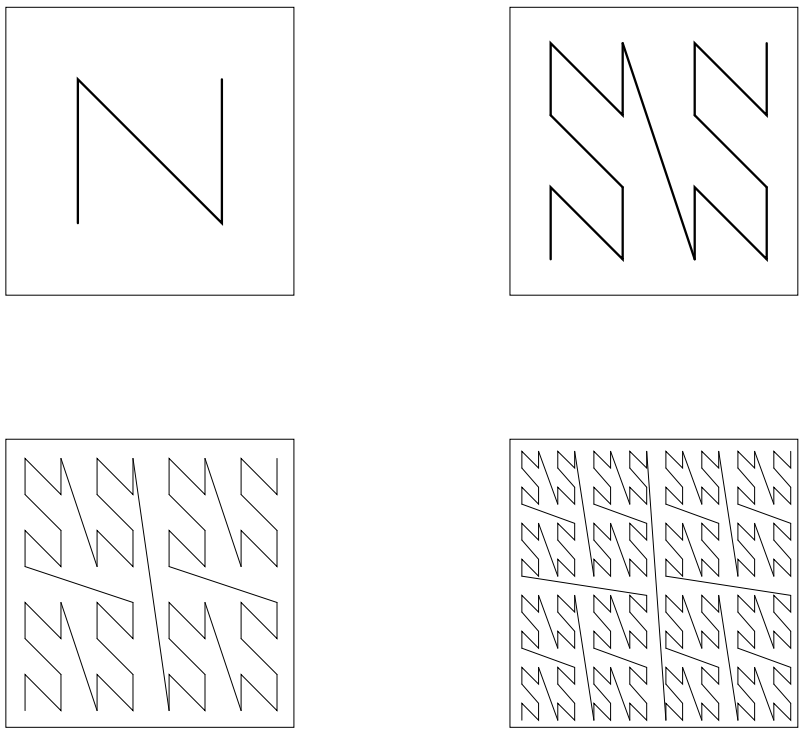


Figure 10: Peano curves

- label(u) the binary representation of the label of the subsquare u;
- u_x and u_y the binary coordinates of u; and
- interleave a function that shuffles its two binary arguments.

The Peano model [37] is particularly interesting to represent surfaces or volumes. As in the raster model the information is represented by a finite number of points, but these points are not uniformly distributed as is the case in the raster model. Actually it is a special kind of the relational model where each relation contains tuples that represent subsquares. Each Peano relation has the form

$$\mathrm{PR}(\mathrm{OID}, \mathrm{P}, \mathrm{S}, \mathrm{A}_1, \ldots, \mathrm{A}_\mathrm{n}),$$

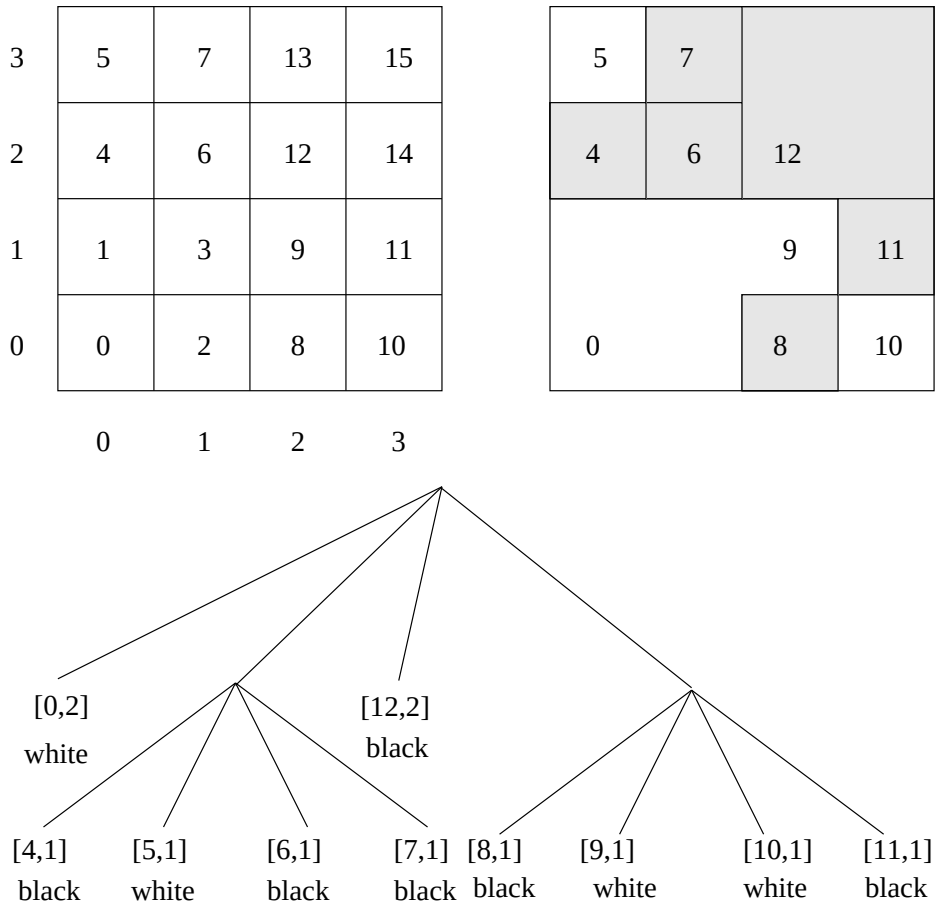


Figure 11: The quadtree for a black and white colored square

where

- PR is the name of the Peano relation;
- OID is the object identifier of which the subsquare is a part;
- P is the Peano key, that is the label of the bottom-left unit subsquare of the square that is represented;
- S is the edge-length of the subsquare;

- $A_1, \ldots, A_n$ are attributes.

Clearly, we will always endeavor a maximal quadrant compaction.

On this model, the Peano algebra was defined with 12 operators on relations:

- union, intersection, difference;
- translation of an object, rotation of a object over $k \cdot 90$ degrees;
- scaling of an object with a factor $2 \cdot i$;
- symmetry according to an axis;
- extracting, which is deleting all objects outside a given window;
- duplication of an object;
- changing the unit, which looses some information;
- classical projection and join.

The Peano model is an elegant model for a direct representation of areas and volumes that is not based on contours. Because of its relationship with Peano curves it is efficiently implementable. Concerning genericity, it is clear that as for the raster model, raster-genericity is supported.

4 Conclusion

From the example models of the previous section it is clear that they can be divided into two categories. On the one hand there are the polynomial model, the topological data model and the model by Egenhofer which are primarily geared to handle theoretical issues. On the other hand there are the raster model, the spaghetti model and the Peano model which were developed for efficient implementation. The former models are not really concerned with issues of efficiency and implementability. The latter models were not developed with theoretical issues (such as genericity) in mind.

We think that a solid theory of spatial information systems will be found at the converging point where practical implementation concerns and theoretical foundations meet.

Acknowledgments
The authors are very indebted to Jan Van den Bussche and Dirk Van Gucht for their fruitful cooperation and to the referees for their constructive remarks.

References

[1] D. Abel and B.C. Ooi, editors. *Proceedings of the 3rd International Symposium on Spatial Databases*, volume 692 of *Lecture Notes in Computer Science*, Berlin, 1993. Springer-Verlag.

[2] P. Alexandroff. *Elementary Concepts of Topology.* Dover Pub. Inc., New York, 1961.

[3] J. Bochnak, M. Coste, and M.-F. Roy. *Géométrie algébrique réelle.* Springer-Verlag, 1987.

[4] A. Buchmann, editor. *Proceedings of the 1st International Symposium on Spatial Databases*, volume 409 of *Lecture Notes in Computer Science*, Berlin, 1989. Springer-Verlag.

[5] W. Burton. Representation of many-sided polygons and polygonal lines for rapid processing. *Comm. of the ACM*, 20, 3:166–171, 1977.

[6] A. Chandra and D. Harel. Computable queries for relational database systems. *Journal of Computer and System Sciences*, 21(2):156–178, 1980.

[7] P. Cole and C. King. *Quantitative Geography.* John Wiley, London, 1968.

[8] J.P. Corbett. Topological principles in cartography. Technical Report 48, US Bureau of the Census, Washington DC, 1979.

[9] E. Chan E. and R. Zhu. QL/G - A query language for geometric data bases. Technical Report CS-94-25, Univ. of Waterloo, 1994.

[10] M. Egenhofer. A formal definition of binary topological relationships. In *Foundations of Data Organization and Algorithms, 3rd International Conference, FODO 1989*, volume 367 of *Lecture Notes in Computer Science*, pages 457–472, 1989.

[11] M. Egenhofer. Reasoning about binary topological relations. In Buchmann [4], pages 143–160.

[12] M. Egenhofer. Topological relations between regions in $\mathbf{R^2}$ and $\mathbf{Z^2}$. In *Advances in Spatial Databases, Third International Symposium, SSD'93*, volume 692 of *Lecture Notes in Computer Science*, pages 316–336, Berlin, 1993. Springer-Verlag.

[13] M. Egenhofer. Spatial SQL: a query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, 1994.

[14] M. Egenhofer and A. Frank. Towards a spatial query language: User interface considerations. In F. Bancilhon and D.J. DeWitt, editors, *Proceedings of the Fourteenth International Conference on Very Large Data Bases*, pages 124–133. Morgan Kaufmann, 1988.

[15] M. Egenhofer and R. Franzosa. On the equivalence of topological relations. *Int. J. Geographical Information Systems*, pages 523–542, 1994.

[16] M.J. Egenhofer and J.R. Herring, editors. *Proceedings of the 4th International Symposium on Spatial Databases*, volume 951 of *Lecture Notes in Computer Science*, Berlin, 1995. Springer-Verlag.

[17] O. Günther. *Efficient structures for geometric data management*. Number 337 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1988.

[18] O. Günther and H.-J. Schek, editors. *Proceedings of the 2nd International Symposium on Spatial Databases*, volume 525 of *Lecture Notes in Computer Science*, Berlin, 1991. Springer-Verlag.

[19] R. Güting. Geo-relational algebra: A model and query language for geometric database systems. In *Proceedings of Extending Data Base Technology*, volume Springer-Verlag, pages 506–527, 1988.

[20] R. Güting. Gral: An extensible relational database system for geometric applications. In *Proceedings of 15th VLDB*, pages 33–44, 1989.

[21] R. Güting. GraphDB: A data model and query language for graphs in databases. Technical Report Informatik Berichte 155-2/1994, FernUniversität, Hagen, 1994.

[22] R. Güting. An introduction to spatial database systems. *The VLDB Journal*, 3(4), 1994.

[23] R. Güting and Schneider. Realms: A foundation for spatial data types in database systems. In Abel and Ooi [1], pages 14–35.

[24] R. Güting and M. Schneider. Realm-based spatial data types: The ROSE algebra. Technical Report TR 141-3-93, Fern Universität, Hagen, 1993.

[25] M. Gyssens, J. Van den Bussche, and D. Van Gucht. Complete geometrical query languages. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 62–67, New York, 1997. ACM.

[26] T. Hirst and D. Harel. Completeness results for recursive data bases. In *Proc. 12th Symp. on Princ. of Database Systems*, pages 244–252, 1993.

[27] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51:26–52, 1995.

[28] A. Kemper and M. Wallrath. An analysis of geometric modeling in database systems. *Computing Surveys*, 19(1):47–91, 1987.

[29] B. Kuijpers, J. Paredaens, and J. Van den Bussche. Lossless representation of topological spatial data. In Egenhofer and Herring [16].

[30] B. Kuijpers, J. Paredaens, and L. Vandeurzen. Semantics in spatial databases. In L. Libkin and B. Thalheim, editors, *Semantics in Databases*, Lecture Notes in Computer Science. Springer-Verlag, 1997.

[31] E.E. Moise. *Geometric Topology in Dimensions 2 and 3*, volume 47 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1997.

[32] O.Günther and A. Buchmann. Research issues in spatial databases. *Sigmod Record*, 19(4):61–68, 1990.

[33] J. Paredaens. Spatial databases, the final frontier. In G. Gottlob and M.Y. Vardi, editors, *Proceedings of the 5th International Conference on Database Theory - ICDT'95*, volume 893 of *Lecture Notes in Computer Science*, pages 14–32. Springer-Verlag, 1995.

[34] J. Paredaens, J. Van den Bussche, and D. Van Gucht. Towards a theory of spatial database queries. In *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 279–288, New York, 1994. ACM Press.

[35] G. Peano. Sur une courbe qui remplit toute une aire plane. *Mathematische Annalen*, 36(a):157–160, 1890.

[36] F. Preparata and M. Shamos. *Computational Geometry.* Springer-Verlag, 1985.

[37] D. Thompson R. Laurini. *Fundamentals of Spatial Information Systems.* Number 37 in APIC Series. Academic Press, 1992.

[38] PH. Rigaux and M. Scholl. Multiple representation modelling and querying. In J. Nievergelt, H.J. Schek Th. Roo and, and P. Widmayer, editors, *International Workshop on Advanced Research in Geographic Information Systems (IGIS)*, number 884 in Lecture Notes in Computer Science, pages 59–69, Berlin, 1994. Springer-Verlag.

[39] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

[40] M. Scholl and A. Voisard, editors. *Proceedings of the 5th International Symposium on Spatial Databases*, volume 1262 of *Lecture Notes in Computer Science*, Berlin, 1997. Springer-Verlag.

[41] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California Press, 1951.