On the Desirability and Limitations of Linear Spatial Database Models

Luc Vandeurzen[1], Marc Gyssens[1], Dirk Van Gucht[2]

[1] Dept. WNI, University of Limburg, B-3590 Diepenbeek, Belgium,
lvdeurze@alpha.luc.ac.be, gyssens@charlie.luc.ac.be.
[2] Computer Science Dept., Indiana Univ., Bloomington, IN 47405-4101, USA,
vgucht@cs.indiana.edu.

Abstract. A general linear spatial database model is presented in which both the representation and the manipulation of non-spatial data is based on first-order logic over the real numbers with addition. We first argue the naturalness of our model and propose it as a general framework to study and compare linear spatial database models. However, we also establish that no reasonable safe extension of our data manipulation language can be complete for the linear spatial queries in that even very simple queries such as deciding colinearity or computing convex hull of a finite set of points cannot be expressed. We show that this fundamental result has serious ramifications for the way in which query languages for linear spatial database models have to be designed.

1 Introduction

There are many database applications that need the ability to store and manipulate geometric data, such as geographic information systems (GIS), geometric modeling systems (CAD), and temporal databases. We refer the reader to the following papers for more background on the work done about spatial and temporal databases[3]. [12, 29]

In a recent paper [18], Güting specified requirements for spatial database systems: a spatial database system must first and foremost be a database system, meaning that it should offer the tools needed to represent, store, and manipulate both conventional and geometric data objects; in addition it should offer spatial data types in its data model and query language; and finally it should support spatial data types in its implementation, e.g., by making available spatial indexing and algorithms for spatial joins.

Spatial database models designed in accordance with the above requirements can roughly be categorized in models based on fixed and variable spatial dimensions. In models based on fixed spatial dimensions (e.g., [2, 16, 17, 35, 33]), the spatial data types are subclasses of all possible point sets of a Euclidean space of some fixed dimension (usually 1, 2, or 3), such as, e.g., points, lines, and polygons.

[3] Since temporal databases can be interpreted as 1-dimensional spatial databases, we shall not give them separate consideration.

Unfortunately, the particular choices of spatial data types and corresponding operators in these models are somewhat "ad hoc" as no singular set of spatial data types and corresponding operators is known to serve well all spatial purposes. Models based on variable spatial dimensions (e.g., [10, 22, 24, 28]) avoid this lack of generality by adopting a more declarative approach. However, some of the latter models may be too general from an implementational perspective.

It is the purpose of this paper to bridge the gap between the two main approaches by presenting a general, variable-dimensional, *linear* spatial database model as a formal framework to study the representation and manipulation of linear spatial data. In Section 2, we introduce our model as a restriction of the very general (non-linear) spatial database model considered by Paredaens et al. [28]. The point sets in the model thus obtained are called semi-linear sets and are characterized as definable in the first-order theory over the real numbers with addition. By providing some alternative characterizations and establishing desirable closure properties semi-linear sets on the one hand, and by proposing a simple and natural declarative, calculus-like query language, called FO + linear, for which an equivalent procedural algebra can be defined, on the other hand, we argue the appropriateness of our model for the purpose it is intended. In Section 3, we study the expressiveness of FO + linear. Although certain complex geometric decision problems and computations can be expressed in an elegant way, we were able to prove that no reasonable safe extension of FO + linear can be complete for the linear spatial queries in that even very simple queries such as deciding colinearity or computing convex hull of a finite set of points cannot be expressed. The viability of alternative strategies to obtain a richer language that circumvent the deep inherent problem identified in this paper are examined. In Section 4, finally, we discuss the ramifications of our main result and compare our approach with other work.

2 A General Linear Spatial Database Model

2.1 Semi-linear Sets

Linear spatial database models and prototypes proposed in the literature typically focus on a finite number of specific spatial data types one might designate as "linear," the particular choice of these primitives usually being driven by the applications that are intended. The choice of these "linear" data types is further motivated by the observation that many geometrical operations on typical linear data, such as lines, and polygons, and their counterparts in three dimensional space, have efficient algorithms. Thus, linear data types are also attractive from an implementational perspective. Variable-dimension models avoid the "ad-hoc" approach of choosing a set of data types and operators satisfying all application needs by offering a general, declarative framework. It goes without saying that such a general framework offers a tool to study spatial databases and their properties in a formal way, as is the case for conventional databases. In our model we try to combine the benefits of these two approaches. It is our purpose to study

linear, spatial databases from a general perspective by offering a constraint-based data model and a calculus-like query language with an equivalent algebra. To do this, we took the most liberal restriction possible of an existing very general and non-linear spatial database model, considered by Paredaens et al. [28].

Paredaens, Van den Bussche, and Van Gucht considered as spatial data all geometrical figures definable in elementary geometry, i.e., first-order logic over the real numbers with addition and multiplication. These figures are called *semi-algebraic sets* in real algebraic geometry. [4] The rationale behind this approach was that the first-order theory of the reals is decidable by means of a very strong form of effective quantifier elimination [11, 3], and that, consequently, many properties of semi-algebraic sets are decidable, too. [20]

A formula in the first-order logic of the reals, a *real formula* for short, is built from *atomic real formulae* using boolean operators and quantification over real variables; *atomic real formulae* are conditions built from *real terms* using one of the six binary comparison relations $=$, $<$, $>$, $\leq$, $\geq$, and $\neq$; and *real terms* are polynomials in real variables with integer coefficients.

A very general and appealing way to obtain linear figures is to consider only those real formulae that are exclusively built from real terms that are *linear* polynomials; these real formulae will be called *linear formulae* and the real terms from which they are built will be called *linear terms*. Clearly, linear formulae can be characterized as first-order formulae over the real numbers with addition only. Without loss of generality, we may assume that *atomic linear formulae* are of the form $\sum_{i=1}^{m} a_i x_i \, \theta \, a$, where $x_1, \ldots, x_m$ are real variables, $a_1, \ldots, a_m$ are integer coefficients, a is an integer, and θ is one of $=$, $>$, $\geq$, $<$, $\leq$, and $\neq$.

Every linear formula $\varphi(x_1, \ldots, x_m)$ with free real variables $x_1, \ldots, x_m$ defines a geometrical figure $\{(x_1, \ldots, x_m) \mid \varphi(x_1, \ldots, x_m)\}$ in m-dimensional Euclidean space $\mathbf{R}^m$ by letting real variables range over the real numbers. Semi-algebraic sets defined in this way are called *semi-linear sets*. Examples of semi-linear sets are given in Example 1.

It is of course important that these semi-linear sets satisfy several desirable closure properties:

Proposition 1. *Semi-linear sets are closed under the set operations union, difference, intersection, and Cartesian product, and under projection.*

The proof of the above proposition is straightforward and is therefore omitted. Notice that the above closure properties can also be regarded as providing interpretations for the various Boolean operators occurring in linear formulae. In particular, existential quantification can be interpreted as a projection. Finally, notice that negation can easily be computed, as the negation of an atomic linear formula is obtained by appropriately changing the comparison relation.

Next, the above closure properties allow us to establish two alternative characterizations of semi-linear sets.

Günther [15] defines *polyhedral chains* as a representation scheme for geometric data. A *polyhedral chain* in a Euclidean space (of arbitrary dimension) is defined as a finite sum of *cells* each of which is a finite intersection of half-spaces.

A polyhedral chain is called *semi-linear* if its cells can be described by equations with rational coefficients.

Proposition2. *Semi-linear sets and semi-linear polyhedral chains represent the same class of figures.*

Proof. Since semi-linear polyhedral chains can be defined in terms of half-spaces that can clearly be described by atomic linear formulae, Proposition 1 yields that they are semi-linear sets.

Conversely, an atomic linear formula represents either a hyperplane or an open or closed half-space or the complement of a hyperplane, which is the union of two open half-spaces. It is therefore easy to see that any semi-linear set defined by a linear formula *without* quantifiers can alternatively be defined as a semi-linear polyhedral chain. This result extends to semi-linear sets defined by *general* linear formulae, as the quantifiers can be eliminated [21, 19] (details omitted).

Another practical tool to deal with semi-linear sets is *polytopes*. A *polytope* in a Euclidean space (of arbitrary dimension) is defined as the convex hull of a non-empty finite set of points in that space. [7, 27, 23]. A polytope is called *semi-linear* if it can be defined in terms of points with rational coefficients. An *open polytope* is the topological interior of a polytope with respect to the smallest sub-space containing the polytope.

Proposition3. *Bounded semi-linear sets and finite unions of open semi-linear polytopes are equivalent.*

Proof. Every open semi-linear polytope can be written as a finite intersection of open half-spaces of which the bounding hyperplanes can be described by equations with rational coefficients. Therefore, it is possible to write the union of open semi-linear polytopes as a semi-linear polyhedral chain. Conversely, a bounded semi-linear set can be written as a semi-linear polyhedral chain. Because of the boundedness of the semi-linear set, all cells of the polyhedral chain are bounded and can therefore be shown to be the union of open semi-linear polytopes (details omitted).

The above characterizations allow us to conclude that most spatial data types found in the literature are sub-types of the semi-linear sets. Güting [16, 17] in his geo-relational algebra proposes the spatial data types *point*, *line*, and *polygon*, which can be seen as 0-, 1-, and 2-dimensional polytopes, respectively.[4] Egenhofer [13] in his spatial data representation model proposes as basic objects *simplices*, 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. As opposed to general semi-algebraic sets which are too

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

complex, we believe semi-linear sets have the potential for efficient implementation. The alternative characterizations we presented offer the opportunity to use polyhedral chains or polytopes as internal representation for semi-linear sets. Günther [15] 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. [31]. Several operations and techniques in computational geometry, such as plane sweep and divide-and-conquer, can be used for this purpose. [26, 36, 8, 30] Brodsky et al. [6] introduced canonical forms for semi-linear sets to make efficient implementation of operations on semi-linear sets possible. Lassez et al. [25, 19] have proposed variable elimination algorithms for sets of linear constraints. Finally, the notion of semi-linear set is not bound to any particular dimension. Even though practical applications are rarely situated in a dimension higher than 4, this generality of semi-linear sets is of relevance, since—as pointed out earlier—semi-linear sets defined by existentially quantified linear formulae can straightforwardly be interpreted as projections of higher-dimensional semi-linear sets defined by unquantified linear formulae.

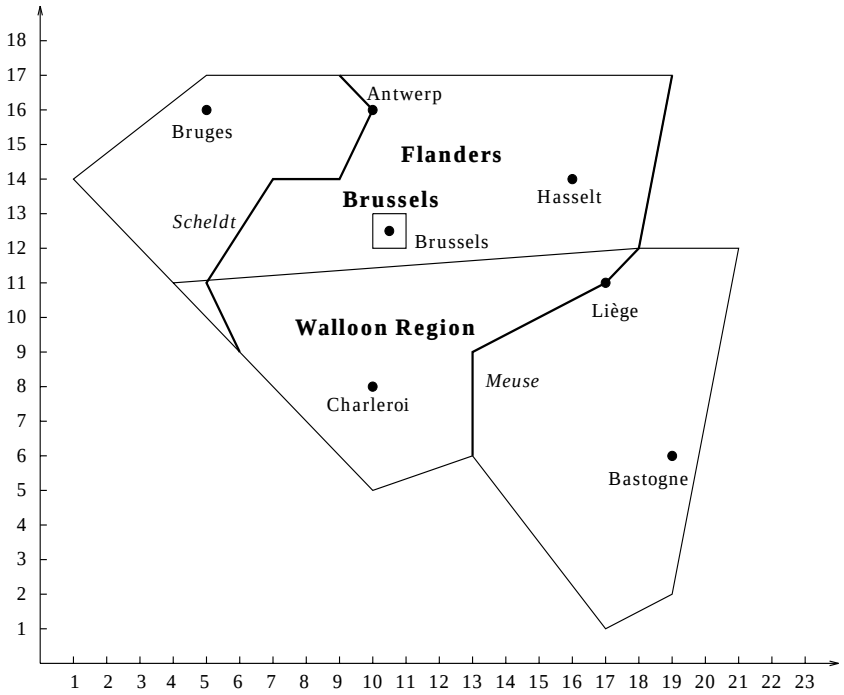
2.2 The Data Representation Model

A linear spatial database *scheme*, $\mathcal{S}$, is a finite set of *relation names*. Each relation name, R, has a type which is a pair of natural numbers, $[n, m]$. Here, n denotes the number of non-spatial columns and m the dimension of the single spatial column of R. Consider a relation type $[n, m]$. A *syntactic tuple* of type $[n, m]$ has the form $(a_1, \ldots, a_n; \varphi(x_1, \ldots, x_m))$, with $a_1, \ldots, a_n$ non-spatial values of some domain, U, and $\varphi(x_1, \ldots, x_m)$ a linear formula with m free variables. As already observed, we may assume without loss of generality that this formula is quantifier-free. A *syntactic relation* of type $[n, m]$ is a finite set of syntactic tuples of type $[n, m]$. A *syntactic instance*, finally, is a mapping assigning to each relation name of a scheme $\mathcal{S}$ a syntactic relation of the same type.

The semantics of a syntactic tuple $t = (a_1, \ldots, a_n; \varphi(x_1, \ldots, x_m))$ of type $[n, m]$ is the possibly infinite subset of $U^n \times \mathbf{R}^m$ denoted as $I(t)$ and defined as the Cartesian product $\{(a_1, \ldots, a_n)\} \times S$, 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, called *semantic relations*, the tuples of which are called *semantic tuples*. The semantics of a syntactic relation, r, is the semantic relation denoted as $I(r)$ and defined as $\bigcup_{t \in r} I(t)$. Finally, the semantics of a syntactic instance, $\mathcal{I}$, over a database scheme $\mathcal{S}$ is the mapping assigning to each relation name R in $\mathcal{S}$ the semantic relation $I(\mathcal{I}(R))$.

Example 1. The example in Figure 1 shows a spatial database representing geographical information about Belgium.

Notice that a syntactic relation has exactly one spatial attribute. Since applications which would require more spatial attributes can be simulated with one spatial attribute using Cartesian product, we chose not to complicate the formalism by relaxing the restriction we imposed.



Regions

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

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	$((y \leq 17) \wedge (x + y = 26) \wedge (y \geq 16)) \vee$ $((y \leq 16) \wedge (2x - y = 4) \wedge (y \geq 14)) \vee$ $((x \leq 9) \wedge (x \geq 7) \wedge (y = 14)) \vee$ $((y \leq 14) \wedge (-3x + 2y = 7) \wedge (y \geq 11)) \vee$ $((y \leq 11) \wedge (2x + y = 21) \wedge (y \geq 9))$

Fig. 1. Example of a spatial database.

2.3 Linear Spatial Queries

In non-spatial database theory, a query is usually defined as a mapping from databases to databases which (i) is computable and (ii) satisfies some regularity condition, usually referred to as genericity. [9]

In spatial models such as ours, the picture is somewhat more complicated, since queries can be viewed both at the syntactic level and the semantic level. The ramifications of this duality were discussed at length by Paredaens, Van den Bussche, and Van Gucht in the context of their general spatial data model [28]. Therefore, we shall only summarize their main conclusions here, in the context of our model:

1. Given an input scheme $\mathcal{S}_{\text{in}}$ and an output scheme $\mathcal{S}_{\text{out}}$, a linear spatial query is a mapping of the linear spatial database instances of $\mathcal{S}_{\text{in}}$ to the linear spatial database instances of $\mathcal{S}_{\text{out}}$, both at the syntactic and the semantic level.
2. At the syntactic level, a linear spatial query must be partially recursive.
3. At the semantic level, a linear spatial query must satisfy certain genericity conditions.

We shall not elaborate on the nature of the above-mentioned genericity conditions as this issue is not within the scope of the present paper. For the sequel, it suffices to realize that a linear spatial query language must be well-defined both at the syntactic and semantic level.

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."* More complicated linear spatial queries are given in Section 3.1.

2.4 The Linear Spatial Calculus and Algebra

In this section, we present two query languages, a calculus and an algebra, and establish their equivalence. As both languages in our opinion were kept as simple as can reasonably expected, we feel that our equivalence result emphasizes the naturalness of both languages.

We first define the *linear calculus*. The *linear calculus* is obtained by adding to the language of linear formulae defined in Section 2.1 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.

Linear calculus formulae can be interpreted as mappings from linear spatial database instances to linear spatial database instances *at the semantic level* in the standard way.[5]

In a straightforward manner, the spatial algebra of Paredaens, Van den Bussche, and Van Gucht [28] can be restricted to a *linear algebra* of which the expressions can be interpreted as computable mappings from linear spatial database instances to linear spatial database instances *at the syntactic level*.

Using the same techniques as Paredaens et al., it is possible to establish the following result:

Proposition 4. *Every linear calculus formula can be effectively converted into a linear algebra expression and vice-versa, in such a way that both express the same mapping from linear spatial database instances to linear spatial database instances, respectively at the semantic and syntactic level.*

The equivalence result in Proposition 4 also establishes that the linear calculus (or algebra) is indeed a spatial query language in the sense of Section 2.3.

Example 3. The query in Example 2 can be expressed by the following linear calculus expression:

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

3 Expressiveness of Linear Spatial Query Languages

In this section, we shall give results concerning both the expressiveness and limitations of the linear spatial calculus of Section 2.4, which will be referred to as FO + linear for brevity. The spatial calculus of Paredaens, Van den Bussche, and Van Gucht [28] designed to manipulate geometric objects definable by general real formulae shall be referred to as FO + poly.

3.1 Expressiveness of FO + linear

Up to now, a precise characterization of the expressive power of FO + linear is still wide open. In this section, we try to give a feeling for the kind of queries that can be solved in FO + linear by presenting some typical examples of topological or geometrical properties computable in FO + linear.

In order to state the solutions to our example queries concisely, we shall use some abbreviations. We shall use vector notion to denote points. In this notation, equations such as $\mathbf{x} - \mathbf{y} < \mathbf{z}$ should be interpreted coordinate-wise. In particular, $\neg(\mathbf{x} = \mathbf{0})$ denotes that $\mathbf{x}$ is not the origin of the coordinate system, whereas $\mathbf{x} \neq \mathbf{0}$ denotes that *none* of the coordinates of $\mathbf{x}$ equals 0! In all the queries below, the input database consists of one relation name S of an arbitrary purely spatial

[5] The linear calculus can also be shown to satisfy a genericity condition, but we shall not digress on this issue here.

type. The restriction on S is justified since in FO + linear manipulation of both conventional and geometric data can be done in a straightforward manner as is shown earlier (Example 3).

Example 4. The following FO + linear expression decides whether S is *discrete*:

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

Since discrete semi-algebraic sets are necessarily finite [4], the same property holds a fortiori also for semi-linear sets. Conversely, a finite semi-linear is necessarily discrete. Hence the above expression can also be used to decide whether S is *finite*. It is however possible to decide finiteness of semi-linear sets without having to rely on the above property of semi-algebraic sets as for an arbitrary set in a Euclidean space finiteness is always equivalent to discreteness *and boundedness*. 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 5. In this example, we show that 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:

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

Similarly, the topological closure of S is computed by the following FO + linear expression:

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

Hence, also the topological boundary of S can be computed as the difference of the topological closure and the topological interior. We note that Egenhofer in his paper [13] showed that with these topological operations, a whole class of topological properties can be expressed in $\mathbf{R}^2$.

Due to space limitations, we conclude this section with some useful queries that can be expressed in FO + linear, without giving the formulae.

- Is a semi-linear set of $\mathbf{R}^n$ n-dimensional (i.e., not embeddable in a space of lower dimension)?
- Compute the regularization[6] of a semi-linear set. Because of this query, the regularization of the set operations union, intersection and difference can be computed. This is an important result since in practice it turns out that the regularized set operations are more important than the standard set operations.
- Translate or scale a semi-linear set. Reflection according to some axis or the origin is also computable.

[6] Intuitively, a regular set has no dangling or isolated boundary points.

3.2 Limitations of FO + linear

In this section, we demonstrate that there are fundamental, *inherent* limitations to safe calculus-like languages for linear spatial databases, which in particular apply to FO+linear. By *safe*, we mean that the language can *only* express linear queries.

Definition 5. Let $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ be m-dimensional vectors of real variables. The $3m$-ary *colinearity* predicate $\mathrm{line}_m(\mathbf{x}, \mathbf{y}, \mathbf{z})$ evaluates to *true* if in m-dimensional Euclidean space the points with coordinates $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are colinear.

Theorem 6. *The language* FO+linear+colinearity *is equivalent to the language* FO + poly.

Proof. Obviously, the colinearity predicate $\mathrm{line}_m(\mathbf{x}, \mathbf{y}, \mathbf{z})$ can be expressed in FO + poly by the following formula:

$$(\mathbf{x} = \mathbf{y}) \vee (\exists \lambda)(\mathbf{z} = \lambda \mathbf{x} + (1 - \lambda)\mathbf{y}) \quad .$$

Conversely, consider the ternary multiplication predicate $\mathrm{product}(x, y, z)$ which is true if $z = xy$. Let $\mathbf{x}$ and $\mathbf{z}$ be m-ary vectors of real variables of which the first components are x and z, respectively, and let $\mathbf{e}_2$ and $\mathbf{y}$ be m-ary vectors of real variables of which the second components are 1 and y, respectively. All other components of $\mathbf{e}_2$, $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are 0. In m-dimensional Euclidean space, this predicate can be expressed by the following formula in FO + linear + colinearity:

$$(\forall \mathbf{u}) \neg (\mathrm{line}_m(\mathbf{x}, \mathbf{e}_2, \mathbf{u}) \wedge \mathrm{line}_m(\mathbf{z}, \mathbf{y}, \mathbf{u})) \quad .$$

The above formula expresses the geometric construction of the product of two real numbers (see, e.g., [34], p. 144), whence its correctness. (The expression is not correct in case of $y = 1$, $y = 0$ or $x = 0$. However, these individual cases can be treated separately.) Clearly, any atomic real formula can be expressed in FO + linear extended with the above multiplication predicate. Therefore, FO + linear + colinearity and FO + poly are equivalent.

Corollary 7. *The convex hull of a set of n points in m-dimensional Euclidean space cannot be expressed as a $m(n+1)$-ary predicate in any safe extension of* FO + linear *if $m \geq 2$ and $n \geq 2$.*

Proof. The proof follows immediately from the above theorem and the observation that three points in m-dimensional Euclidean space are colinear if and only if one of them is on the line segment defined by the other two, which is the convex hull of that pair of points.

Notice that the convex hull of any m-dimensional (semi-algebraic) set S can be expressed in FO + poly as

$$\{(\mathbf{z}) \mid (\exists \mathbf{x}_i)(\exists \lambda_i)(\bigwedge_{i=0}^{m} S(\mathbf{x}_i) \wedge \bigwedge_{i=0}^{m} (\lambda_i \geq 0) \wedge (\sum_{i=0}^{m} \lambda_i = 1) \wedge (\mathbf{z} = \sum_{i=0}^{m} \lambda_i \mathbf{x}_i))\} \quad .$$

It is interesting to note that in FO + linear convexity is decidable:

Proposition 8. [32] *It is decidable in* FO + linear *whether a semi-linear set is convex.*

Proof. The following FO + linear expression decides whether S is *convex*:

$$(\forall \mathbf{x})(\forall \mathbf{y})((S(\mathbf{x}) \wedge S(\mathbf{y})) \Rightarrow (\exists \mathbf{z})(S(\mathbf{z}) \wedge (2\mathbf{z} = \mathbf{x} + \mathbf{y}))) \ .$$

Similarly, one may ask if in FO + linear it is decidable whether a semi-linear set is a line, even though colinearity is not computable.

3.3 Extensions of FO + linear

The question arises whether "reasonable," "non-trivial" safe proper extensions of FO + linear exist at all. In this section, we review some mechanisms for extension proposed by other authors and discuss to which extent they might be useful for our purposes.

In recent papers, Afrati, Cosmadakis, Grumbach, and Kuper [10, 1] proposed a calculus in which they extended a language similar to FO+linear with variables that range over lines. Unfortunately, they were able to show that their language is also equivalent to FO + poly. Thus, extending FO + linear with line variables does not lead to a safe proper extension of FO + linear.

In a series of papers, J.-L. Lassez et al. (e.g., [25]) proposed so-called *parameterized queries*. A *parameterized query* is of the form

$$\{(\alpha_1, \ldots, \alpha_n, \beta) \mid (\forall y_1) \ldots, (\forall y_n)(S(y_1, \ldots, y_n) \Rightarrow \\ ((\alpha_1 y_1 + \cdots + \alpha_n y_n \leq \beta) \wedge \varphi(\alpha_1, \ldots, \alpha_n, \beta))\}$$

where S is a set of linear constraints on $y_1, \ldots, y_n$ and φ is a set of linear constraints on the parameters $\alpha_1, \ldots, \alpha_n$ and β. Observe that because of the term $\alpha_1 y_1 + \cdots + \alpha_n y_n$ this formula violates the syntax of FO + linear. However, it follows from a result by T. Huynh, C. Lassez, and J.-L. Lassez [19] that parameterized queries are safe linear spatial queries.

The precise relationship between FO+linear and the language of parametrized queries is as of yet still unclear. On the one hand, some FO + linear queries may not be expressible by parametrized queries, but on the other hand, not all parameterized may be expressible by FO + linear queries because they violate FO + linear syntax. A mechanism such as used in parametrized queries remains a viable candidate for extending FO + linear, provided sufficient syntactic restrictions are built in to prevent that, e.g., colinearity can be expressed.

We want to point out that languages such as FO + linear rely heavily on the use of subqueries as a tool to solve more complicated queries. A possible approach towards extending FO + linear might therefore be adding some non-expressible linear queries which may not be applied arbitrarily to the results of subqueries. This is currently under investigation.

4 Discussion

In this paper, we have proposed a general linear spatial database model that tries to combine the benefits of both fixed dimensional linear spatial database models and general variable dimensional databases. We used this model as a framework to study the manipulation and representation properties of formal and still implementable linear spatial database models in general.

The proposed model uses semi-linear sets as spatial data type. We showed by establishing the equivalence of semi-linear sets with the data types used in fixed dimensional linear spatial database models that semi-linear sets have the right properties to be used as a practical spatial data type. Otherwise, semi-linear sets form a specialization of semi-algebraic sets, which are the data type used in the variable-dimension database model considered by Paredaens et al. [28]. Also the query language, FO + linear, is a specialization of the query language defined in [28] and is equivalent with a procedural algebra language. Both our spatial data and query language (FO + linear) have linear constraints as their fundamentals, and can therefore serve as a formal framework to study the intended properties.

Although a lot of interesting practical queries can be expressed in FO+linear, we found that FO + linear nevertheless has serious shortcomings with respect to expressive power, and can therefore not be considered fully adequate as a *querying tool* for linear spatial databases. The central issue in this regard is that certain natural linear queries (such as deciding colinearity or computing convex hull of a finite set of points) cannot be expressed. However, the problem encountered is not merely a deficiency of our particular model, but is of a deep fundamental nature, since we have shown that extending FO + linear to accommodate these queries leads to languages which are no longer safe in the sense that also non-linear data can be derived.

We have shown by considering some other proposals for linear spatial database query languages that there is no obvious way to circumvent this problem. In this connection, we want to mention two more approaches to linear spatial databases.

Brodsky and Kornatzky [6] propose a complex-object object-oriented spatial database models in which the data in the objects are represented as linear constraints and are therefore equivalent to semi-linear sets. Therefore, they face the same expressibility problems as we do. The issue of complex-object types is orthogonal to the more fundamental issue of reasoning about data types specific to spatial and temporal data.

Kanellakis and Goldin [22] describe a general framework for pure spatial relations with as query language the union of some existing query language and a decidable logical theory. Unfortunately, they only work out the dense order constraint case by giving an appropriate algebra. They only suggest the practical importance of linear constraints, but do not say anything about their expressiveness.

Our paper has therefore elicited a fundamental problem in the design of logic-based query languages for linear spatial databases.

In order to overcome this problem it may perhaps be necessary to first restrict FO + linear before extending it. As hinted at towards the end of the previous section, the main culprit of the problem we identified seems to be the absence of any limits to the use of (existential) quantification—geometrically corresponding to projection—in FO+linear. In retrospect, an algebraic approach similar to that of Güting [16] may be most promising and therefore deserves further study in this new light.

References

1. 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, pp. 181–192.
2. W.G. Aref and H. Samet, "Extending a Database with Spatial Operations," in Proceedings *2nd Symposium on Advances in Spatial Databases*, O. Günther, H.-J. Schek, eds., *Lecture Notes in Computer Science*, vol. 525, Springer-Verlag, Berlin, 1991, pp. 299–319.
3. D.S. Arnon, "Geometric Reasoning with Logic and Algebra," *Artificial Intelligence*, 37, 1988, pp. 37–60.
4. J. Bochnak, M. Coste, and M.F. Roy, *Géométrie algébrique réelle*, in *Ergebnisse der Mathematik und ihrer Grenzgebiete*, 3. Folge, Band 12, Springer-Verlag, Berlin, 1987.
5. 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, pp. 567–580.
6. A. Brodsky and Y. Kornatzky, "The LyriC Language: Querying Constraint Objects," in Proceedings *Post-ILPS'94 Workshop on Constraints and Databases* (Ithaca, NY), 1994.
7. A. Brøndsted, *An Introduction to Convex Polytopes*, in *Graduate Texts in Mathematics*, vol. 90, Springer-Verlag, New York, 1983.
8. I. Carlbom, "An Algorithm for Geometric Set Operations Using Cellular Subdivision Techniques," *IEEE Computer Graphics and Applications*, 7:5, 1987, pp. 44–55.
9. A. Chandra and D. Harel, "Computable Queries for Relational Database Systems," *Journal of Computer and System Sciences*, 21:2, 1980, pp. 156–178.
10. S.S. Cosmadakis and G.M. Kuper, "Expressiveness of First-Order Constraint Languages," *Technical Report*, ECRC–94–13, European Computer-Industry Research Centre, Munich, 1994.
11. G.E. Collins, "Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition," in Proceedings *2nd GI Conf. on Automata Theory and Formal Languages* (Kaiserslautern, Germany), H. Brakhage, ed., *Lecture Notes in Computer Science*, vol. 33, 1975, pp. 134–183.
12. J. Nievergelt and M. Freeston, eds., Special issue on spatial data, *Computer Journal*, 37:1, 1994.
13. 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, pp. 457–472.

14. M.J. Egenhofer, "Why not SQL!", *Int'l J. on Geographical Information Systems*, 6:2, 1992, pp. 71–85.
15. O. Günther, ed., *Efficient Structures for Geometric Data Management*, in *Lecture Notes in Computer Science*, vol. 337, Springer-Verlag, Berlin, 1988.
16. 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, pp. 506–527.
17. 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, pp. 33–34.
18. R.H. Güting, "An Introduction to Spatial Database Systems," *VLDB-Journal*, 3:4, 1994, pp. 357–399.
19. 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*, volume 463. Springer Verlag, Berlin, 1990, pp. 117–131.
20. J. Heintz, T. Recio, and M.F. Roy. "Algorithms in Real Algebraic Geometry and Applications to Computational Geometry," in *Discrete and Computational Geometry*, W. Steiger, J. Goodman, and R. Pollack, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 6, AMS-ACM, 1991, pp. 137–163.
21. J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publ. Co.,, Reading, MA, 1979, pp. 355–357.
22. P.C. Kanellakis and D.Q. Goldin, "Constraint Programming and Database Query Languages," in Proceedings *2nd Conf. on Theoretical Aspects of Computer Software*, M. Hagiya and J.C. Mitchell, eds., *Lecture Notes in Computer Science*, vol. 789, Springer-Verlag, Berlin, 1994.
23. P.J. Kelly and M.L. Weiss. *Geometry and Convexity: a Study in Mathematical Methods*, J. Wiley and Sons, New York, 1979.
24. 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, pp. 299–313.
25. J.-L. Lassez, "Querying Constraints," in Proceedings *9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Nashville, TN), 1990, pp. 288–298.
26. M. Liebling and A. Prodon, "Algorithmic Geometry," in *Scientific Visualization and Graphics Simulation*, D. Thalmann, ed., J. Wiley and Sons. pp. 14–25.
27. P. McMullen and G.C. Shephard, *Convex Polytopes and the Upper Bound Conjecture*, University Press, Cambridge, 1971.
28. 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. pp. 279–288.
29. N. Pissinou, R. Snodgrass, R. Elmasri, I. Mumick, T. Özsu, B. Pernici, A. Segef. B. Theodoulidis, and U. Dayal, "Towards an Infrastructure for Temporal Databases," *SIGMOD Records*, 23:1, 1994, pp. 35–51.

30. F.P. Preparata and D.E. Muller. "Finding the Intersection of n Half-Spaces in Time $O(nlogn)$," *Theoretical Computer Science*, 8, 1979, pp. 45–55.
31. L.K. Putnam and P.A. Subrahmanyan, "Boolean Operations on n-Dimensional Objects," *IEEE Computer Graphics and Applications*, 6:6, 1986, pp. 43–51.
32. E. Robertson, *personal communications*, 1994.
33. N. Roussopoulos, C. Faloutsos, and T. Sellis, "An Efficient Pictorial Database System for PSQL," *IEEE Transactions on Software Engineering*, 14:5, 1988, pp. 639–650.
34. W. Schwabhauser, W. Szmielew, and A. Tarski. *Metamathematische Methoden in der Geometrie*, Springer-Verlag, Berlin, 1983.
35. 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, pp. 119–140.
36. B. Tilove, "Set Membership Classification: a Unified Approach to Geometric Intersection Problems," *IEEE Transactions on Computers*, C-29:10, 1980, pp. 874–883.

This article was processed using the LaTeX macro package with LLNCS style