On the Expressiveness of Linear-Constraint Query Languages for Spatial Databases*

Luc Vandeurzen†
University of Limburg

Marc Gyssens*
University of Limburg

Dirk Van Gucht‡
Indiana University

Abstract

The linear database model, in which semi-linear sets are the only geometric objects, has been identified as suitable for spatial database applications from both modeling expressiveness as query efficiency considerations. For querying linear databases, the language FO+linear has been proposed. In this paper, we examine the expressiveness of this language. First, we present a list of general queries expressible in FO + linear. In particular, we mention the dimension query, which in turn allows us to express several other interesting linear queries. Next, we show the non-expressibility of a whole class of linear queries that are related to sets not definable in FO + linear, a result which demonstrates the need for more expressive linear query languages. In this paper, we show how FO + linear can be extended within FO + poly in a safe way. Whether any of the proposed extensions is complete for the linear queries definable in FO + poly remains open. We do show, however, that it is undecidable whether an expression in FO + poly induces a linear query.

*Preliminary versions of parts of this paper were presented at the 4th International Symposium on Large Spatial Databases—SSD '95 (Portland, Maine, USA, August 1995) and the 2nd International Conference on Principles and Practice of Constraint Programming–CP 96 (Cambridge, Massachusetts, USA, August 1996).

†Dept. WNI, University of Limburg, B-3590 Diepenbeek, Belgium.
Part of the work of the second author was conducted at and supported by the University of Antwerp.
E-mail: lvdeurze@luc.ac.be, gyssens@charlie.luc.ac.be.
WWW: http://www.luc.ac.be/Research/TheoComp.

‡Computer Science Dept., Indiana University, Bloomington, IN 47405-4101, USA.
E-mail: vgucht@cs.indiana.edu.

1 Introduction

A growing number of database applications require the ability to store and manipulate besides alpha-numerical data (e.g., strings, numbers, and dates) also geometric data. Typical examples of such applications are geographic information systems (GIS), geometric modeling systems (CAD), and temporal databases (see, e.g., [12, 36, 34] for an overview). The traditional relational database model cannot provide a natural representation of geometric data and an easy way to express geometric computation in the query language [16, 21, 31]. For that reason, there is an ongoing search for appropriate database models that can handle both alpha-numerical and geometric data. These database models are collectively known as spatial database models.

Existing spatial database models can be divided roughly into two categories: data-type-based models [3, 37, 17, 22, 23, 25, 24, 26] and constraint-based models [29, 27].

Data-type-based models extend the relational database model with a fixed set of spatial data types, typically points, lines, and polygons. As a consequence, geometrical figures are not treated as point-sets, but as finite compositions of points, lines, and polygons. Since the number of spatial data types is fixed, these models are restricted to geometric data in an Euclidean space of some fixed, generally low, dimension. Query languages for data-type-based languages are essentially relational algebra extended with a fixed set of geometrical operators. In the implementations of these models, the data structures to represent the different data types are selected in such a way that the various geometrical operators can be computed as efficient as possible using techniques from computational geometry. While this approach guarantees of course very good performance for several applications, the major drawback of data-type-based models is that there is no single set of data types and geometrical operators known to serve well all purposes.

The constraint-based approach was first proposed in the seminal work of Kuper, Kanellakis, and Revesz [29]. Constraint-based models allow users to define relational databases which may, besides alpha-numerical values, contain constraints formulated as first-order logic formulae of a certain type (e.g., polynomial constraints, linear constraints, or dense order constraints). Such formulae are finite representations of geometrical figures consisting of all points (in an appropriate Euclidean space) satisfying the formulae. In contrast to the data-type-based approach, the constraint-based approach does not necessitate to put an a-priori bound on the dimension of the Euclidean space considered. A natural query language to accompany these database models is the relational calculus extended with the same class of constraints as used to represent the spatial data. The validity of this approach follows from the fact that, for several classes of constraints, first-order logic restricted to these constraints is decidable. This property holds, for instance, for polynomial constraints (whence also for linear constraints), by the quantifier-elimination theorem of Tarski [38], which states that a formula can be replaced by an equivalent, effectively computable quantifier-free formula. In constraint-based models, both the representation and manipulation of the spatial data is inherently declarative. From a theoretical point of view, constraint-based models are preferable over data-type-based models, since the former allow to study spatial databases and their properties in a less ad-hoc and more uniform way than the latter.

Although data models introduced within the framework of polynomial and linear constraints are at the moment proposed as good candidates to deal with spatial data, there are fundamental questions yet unsolved.

Various researchers have studied the expressive power of the query languages based on linear and polynomial constraints [30, 19, 1, 35, 2, 40, 4, 41], but insufficient insight has been obtained in the nature of the queries expressible in these languages. Only recently, results on the non-expressibility of the parity query and the connectivity query within polynomial and linear constraint query languages have appeared in the literature . [19, 30, 4] Furthermore, people started to investigate spatial aggregate functions (e.g., area and volume) and found out that these aggregate functions were undefinable in constraint query languages and that adding these functions to the query language created serious closure[1] problems. [10]

From the implementational point of view, all attention is focused on linear-constraint databases, as general polynomial-constraint databases are not considered feasible. For linear-constraint databases, various implementation projects have recently been started with approaches ranging from restricting the linear constraints which can be used [7, 6, 8] to working with finite precision arithmetic [18].

For the reasons given above, we focus in this paper on the linear-constraint databases, and, more in particular, on the expressiveness of the corresponding query languages. Our contribution is threefold:

1. We identify a collection of general queries expressible in FO + linear, first-order logic extended with linear constraints. In particular, we show that the dimension query is expressible in FO + linear, which in turn yields the expressibility of several other important queries.

2. We present a general theorem stating that the non-expressibility in FO + linear of certain sets of points can be lifted to the non-expressibility of closely related linear queries. As a consequence, we can show that several important linear queries, indispensable in most spatial database applications, cannot be computed in FO + linear.

3. To remedy the shortcomings of FO + linear, we present a technique to extend FO+linear with geometrical operators. The new query languages thus obtained can be seen as a bridge between constraint-based query languages and data-type-based query languages.

The paper is organized as follows. In Section 2, we review the polynomial and the linear database model, and define polynomial- and linear-constraint queries. In Section 3, we propose a list of practical, general-purpose queries which are expressible in FO + linear. In particular, we demonstrate the expressibility of the dimension query, which yields the expressibility of several other important linear queries, as is richly illustrated by examples. In Section 4, we argue that FO + linear is nevertheless *not* sufficiently expressive to be regarded as a general-purpose query language for linear databases. Thereto, we establish a theorem that lifts the non-definability of

[1] A constraint query language is called closed if a query of this language applied on a constraint database always results in a constraint database.

certain point sets to the non-expressibility of closely related linear queries. We give several examples of important linear queries which can be proven to be inexpressible in FO + linear using the above-mentioned theorem. To overcome these problems, we provide a method in Section 5 to extend the query language FO + linear with linear geometric operators in a sound way. In Section 6, finally, we conclude this paper by stating some problems that remain open.

2 Constraint-based database models

In this section, we provide the necessary background of the polynomial and linear database models. We explain the notion query in the context of these databases models. We define two natural query languages, called FO + poly and FO + linear, for the polynomial and the linear database model, respectively. Since the linear database model is a sub-model of the polynomial database model, we start with the latter.

2.1 The polynomial database model

First, we define a *real formula* as a well-formed first-order logic formula built from polynomial equations and inequalities, i.e.,

- if p is a polynomial with real coefficients[2] over the variables $x_1, \ldots, x_n$ over the real numbers, then $p(x_1, \ldots, x_n)\ \theta\ 0$ is a real formula, with $\theta \in \{=, <, >, \leq, \geq, \neq\}$;
- if φ and ψ are real formulae, then $\varphi \wedge \psi$, $\varphi \vee \psi$; and $\neg\varphi$ are real formulae; and
- if x is a real variable and φ is a real formula in which x occurs free, then $(\exists x)\varphi(x)$ is a real formula.

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

$$\{(u_1, \ldots, u_n) \in \mathbf{R}^n \mid \varphi(u_1, \ldots, u_n)\}$$

in n-dimensional Euclidean space $\mathbf{R}^n$ in the standard manner. Point sets defined by real formulae are called *semi-algebraic sets*.

For convenience, we shall frequently use vector notation in real formulae. Atoms involving vector notation must be interpreted coordinate-wise. Thus, $\neg(\vec{x} = \vec{0})$ indicates that $\vec{x}$ is not the origin of the coordinate system, whereas $\vec{x} \neq \vec{0}$ denotes that *none* of the coordinates of $\vec{x}$ equals 0. As usual, $\varphi \Rightarrow \psi$ and $\varphi \Leftrightarrow \psi$ will be used as abbreviations for $\neg\varphi \vee \psi$ and $(\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$, respectively.

In the polynomial database model, the only geometric data under consideration are semi-algebraic sets. By definition semi-algebraic sets are finitely representable by means of real formulae. It must be noted that several real formulae can represent the same semi-algebraic set, as illustrated by the following example.

[2] In order to obtain formulae that are finitely representable, only real coefficients that are finitely representable (e.g., integers) may be allowed. We shall not elaborate on this issue here, however, because it is outside the scope of this paper.

Example 2.1 The following two real formulae define the same area in the plane:

- $(\exists x_3)(\exists x_4)(x_3^2 + x_4^2 = 100 \wedge (x_3 - x_1)^2 + (x_4 - x_2)^2 < 1)$; and
- $x_1^2 + x_2^2 > 81 \wedge x_1^2 + x_2^2 < 121$. □

By the quantifier elimination theorem of Tarski [38], it is always possible to represent a semi-algebraic set by a quantifier-free formula. The same theorem also guarantees decidability of the equivalence of two real formulae.

In essence, the polynomial database model is an extension of the relational data model, where a relation, besides columns that store values of some particular non-spatial data type, can have one extra spatial column of type semi-algebraic set. In contrast with the traditional data columns, there is a sharp distinction between what is stored in a spatial column (finitely representable real formulae) and the meaning of the stored data (possibly infinite point-sets, which may even be unbounded). In the next two paragraphs, we give the formal definitions.

A *polynomial database scheme*, $\mathcal{S}$, is a finite set of *relation names*. We associate with each relation name, R, a type which is a pair of natural numbers, $[m, n]$, where m denotes the number of non-spatial columns and n the dimension of the single spatial column of R. A database scheme has type $[m_1, n_1, \ldots, m_k, n_k]$ if the scheme consists of relation names, $R_1, \ldots, R_k$, respectively of type $[m_1, n_1], \ldots, [m_k, n_k]$. A *syntactic relation* of type $[m, n]$ is a finite set of tuples of the form $(v_1, \ldots, v_m; \varphi(x_1, \ldots, x_n))$, with $v_1, \ldots, v_m$ non-spatial values of some domain, D, and $\varphi(x_1, \ldots, x_n)$ a real formula with n free variables. As argued before, we may assume without loss of generality that this formula is quantifier-free. A *syntactic database instance* is a mapping, $\mathcal{I}$, assigning to each relation name, R, of a scheme, $\mathcal{S}$, a syntactic relation $\mathcal{I}(R)$ of the same type.

Given a syntactic relation, r, the semantic relation $I(r)$ is defined as

$$\bigcup_{t \in r} \Big(\{(t.v_1, \ldots, t.v_m)\} \times \{(u_1, \ldots, u_n) \in \mathbf{R}^n \mid t.\varphi(u_1, \ldots, u_n)\} \Big).$$

This subset of $D^m \times \mathbf{R}^n$ can be interpreted as a possibly infinite $(m+n)$-ary relation, called *semantic relation*, the tuples of which are called *semantic tuples*. The semantics of a syntactic database instance, $\mathcal{I}$, over a database scheme, $\mathcal{S}$, is the mapping, I, assigning to each relation name, R, in $\mathcal{S}$ the semantic relation $I(\mathcal{I}(R))$.

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 the polynomial database model, 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 [33]. Therefore, we shall only summarize their main conclusions here:

1. Given an input scheme $\mathcal{S}_{\text{in}}$ and an output scheme $\mathcal{S}_{\text{out}}$, a query is a mapping of the polynomial spatial database instances of $\mathcal{S}_{\text{in}}$ to the polynomial spatial database instances of $\mathcal{S}_{\text{out}}$, both at the syntactic and the semantic level.

2. At the syntactic level, a query must be partially recursive.

3. At the semantic level, a 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.

We associate with every query a type

$$[m_1, n_1, \ldots, m_k, n_k] \to [m, n]$$

with $[m_1, n_1, \ldots, m_k, n_k]$ the type of the input database scheme and $[m, n]$ the type of the output relation.

The most natural query language accompanying the polynomial data model is obtained by adding to the language of the real formulae the following:

1. a totally ordered infinite set of variables called *non-spatial variables*, disjoint from the set of real variables;

2. atomic formulae of the form $v_1 = v_2$, with v_1 and v_2 non-spatial variables;

3. atomic formulae of the form $R(v_1, \ldots, v_n; x_1, \ldots, x_m)$, with R a relation name of type $[n, m]$, $v_1, \ldots, v_n$ non-spatial variables, and $x_1, \ldots, x_m$ real variables; and

4. universal and existential quantification of non-spatial variables.

In the literature, this query language is commonly known as FO + poly.

Example 2.2 Assume a relation *Lives* of type $[1, 2]$ that contains tuples of persons with their home coordinates. A (simple) query on this relation is *Give the pairs of all people that live exactly at a distance of 10 from each other*. This query can be expressed as

$$\{(p_1, p_2) \mid (\exists x_1)(\exists x_2)(\exists y_1)(\exists y_2)(Lives(p_1, x_1, y_1) \wedge Lives(p_2, x_2, y_2) \wedge (x_1 - x_2)^2 + (y_1 - y_2)^2 = 100\}.$$

in FO + poly. □

Due to the existence of quantifier elimination algorithms for real formula, every FO + poly-query is effectively computable, and yields a polynomial database as result. [29]

2.2 The linear database model

We next introduce the linear database model which is a restriction of the polynomial database model in which only linear polynomial constraints are allowed. Real formulae only containing linear polynomial equations or inequalities, i.e., $\sum_{i=1}^{n} a_i x_i + a \;\theta\; 0$, with $\theta \in \{=, <, >, \leq, \geq, \neq\}$, $x_1, \ldots, x_n$ real variables, and $a_1, \ldots, a_n, a$ real coefficients[3] are called *linear formulae*. Point sets defined by linear formulae are called *semi-linear sets*.

[3]See footnote 2.

In [20], Günther introduces *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. As is well-known, half-spaces can be described in terms of linear inequalities. Furthermore, the Boolean operators occurring in linear formulae can be regarded as the set operations union, complement and intersection, and existential quantification can be interpreted as a geometrical projection. Therefore, it is easy to see that semi-linear sets and polyhedral chains define the same class of geometrical figures. Bounded semi-linear sets can be characterized as finite unions of open polytopes[4]. From this perspective, semi-linear sets cover all popular two- and three-dimensional spatial data types in existing models.

The linear data model is defined in the same way as the polynomial data model above using linear formulae instead of real formulae.

Example 2.3 The example in Figure 1 shows a linear database representing geographical information about Belgium. □

As polynomial queries, linear queries are defined as mappings between linear databases that are well-defined both at the syntactic and the semantic level. A very appealing linear query language for the linear spatial data model, called FO + linear, is obtained by restricting the real formulae in FO + poly to linear formulae.

Example 2.4 An example of a (very simple) linear spatial query on the database in Example 2.3 is *Find all cities to the north of Brussels that lie on a river and give their names and the names of the rivers they lie on.* This query can be expressed as

$$\{(c,r) \mid (\exists x)(\exists y)(\exists b_x)(\exists b_y)(\text{Cities}(c,x,y) \wedge \text{Rivers}(r,x,y)) \wedge \text{Cities}(\mathit{Brussels}, b_x, b_y) \wedge x \geq b_x\}.$$

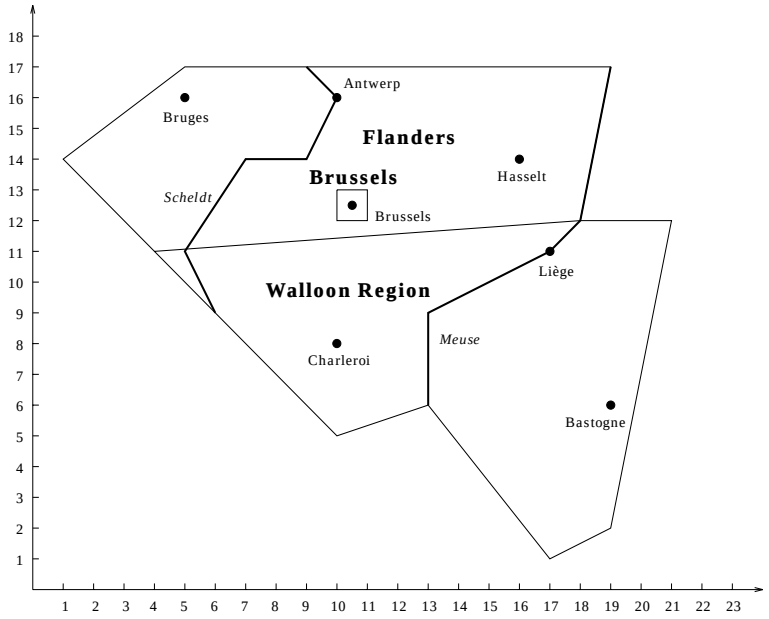
in FO + linear. □

3 Expressiveness of FO + linear

In this section, we present a list of fundamental queries of topological or geometrical nature expressible in FO + linear. To simplify our discussions, we assume that, in all the queries below, the input database consists of one relation name S of an arbitrary purely spatial type.

We start by observing that set operations such as union, intersection, difference, complement, and projection can be expressed rather straightforwardly in FO + linear. In general, any fixed affine transformation of semi-linear sets can be expressed in FO + linear.

[4] A polytope in a Euclidean space is defined as the convex hull of a non-empty set of points in that space. An open polytope is the topological interior of a polytope with respect to the smallest sub-space containing the polytope. In two-dimensional space, for instance, the open polytopes are points, open line segments, and open convex regions.



Regions

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

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

Figure 1: Example of a (linear) spatial database.

Example 3.1 The FO + linear-expression

$$(\exists \vec{y_1})(\exists \vec{y_2})(S(\vec{y_1}) \wedge S(\vec{y_2}) \wedge \vec{x} = \vec{p} + \vec{y_1} - \vec{y_2})$$

computes the line through a point $\vec{p}$ parallel to the line assumed to be stored in S. □

Example 3.2 The FO + linear expression

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

decides whether S is *discrete*. □

Since discrete semi-algebraic sets are necessarily finite [5], the same property holds a fortiori for semi-linear sets. Conversely, a finite semi-linear set is necessarily discrete. Hence the expression in Example 3.2 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. An arbitrary set in a Euclidean space is finite if it is both discrete and *and bounded*.

Example 3.3 The FO + linear expression

$$(\exists \vec{\varepsilon})(\forall \vec{x})(\forall \vec{y})(S(\vec{x}) \wedge S(\vec{y}) \Rightarrow -\vec{\varepsilon} < \vec{y} - \vec{x} < \vec{\varepsilon})$$

decides whether S is *bounded*. □

The expressive power of FO + linear unfolds completely, however, when topological properties of geometrical objects are considered. The definitions of topological interior, boundary, and closure can indeed be translated almost straightforwardly into linear calculus expressions, as shown in the following example.

Example 3.4 The FO + linear expression

$$(\exists \vec{\varepsilon})(\vec{\varepsilon} \neq \vec{0} \wedge (\forall \vec{y})(\vec{x} - \vec{\varepsilon} < \vec{y} < \vec{x} + \vec{\varepsilon} \Rightarrow S(\vec{y})))$$

computes the topological interior of S. Similarly, the FO + linear expression

$$(\forall \vec{\varepsilon})(\vec{\varepsilon} \neq \vec{0} \Rightarrow (\exists \vec{y})(S(\vec{y}) \wedge \vec{x} - \vec{\varepsilon} < \vec{y} < \vec{x} + \vec{\varepsilon}))$$

computes the topological closure of S. The topological boundary of S can be computed as the difference of the topological closure and the topological interior. □

We note that Egenhofer et al. showed in a series of papers [13, 14, 15] that a whole class of topological relationships in the two-dimensional plane, such as *disjoint*, *in*, *contained*, *overlap*, *touch*, *equal*, and *covered*, can be defined in terms of intersections between the boundary, interior, and complement of the geometrical objects. Furthermore, the regularization of a semi-linear set, defined as the closure of its interior[5], can be computed in FO + linear, which is of importance, since the regularized set operators turn out to be indispensable in most spatial database applications [28, 21].

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

The remainder of this section is concerned with another property of geometrical objects often used in spatial database applications, namely *dimension*. For instance, in [11], the dimension is used to further refine the class of topological relationships defined by Egenhofer et al. We now show that it can be decided in FO + linear whether a given semi-linear set has a given number as its dimension. Since there are only finitely many values to consider for the dimension of a semi-linear set, it follows that the dimension can actually be *computed* in FO + linear.

Definition 3.5 The *dimension* of a semi-linear set S of $\mathbf{R}^n$ is the maximum value of d for which there exists an d-dimensional open cube fully contained in S. The dimension of the empty set is defined as -1.

Theorem 3.6 *The predicate* $\dim_n(S, d)$*, in which* S *is a semi-linear set of* $\mathbf{R}^n$ *and* d *is a number, and which evaluates to* true *if the dimension of* S *equals* d*, can be defined in* FO + linear*.*

The correctness of Theorem 3.6 follows from five lemmas we present next. We use the notation $\pi_i(S)$, with S a semi linear set of $\mathbf{R}^n$, to denote the semi-linear set

$$\{(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \mid (\exists x_i) S(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)\}$$

of $\mathbf{R}^{n-1}$. Thus, $\pi_i(S)$ is the orthogonal projection of S onto the i-th coordinate hyperplane of $\mathbf{R}^n$.

Obviously, the following is true:

Lemma 3.7 *The dimension of* $\pi_i(S)$*, with* S *a* d*-dimensional semi-linear set of* $\mathbf{R}^n$*, is at most* d*, for* $1 \leq i \leq n$*.* □

We now show that if $d < n$, at least one projection of S preserves the dimension.

Lemma 3.8 *If* S *is a* d*-dimensional semi-linear set of* $\mathbf{R}^n$*, and* $d < n$*, then there exists* i*,* $1 \leq i \leq n$*, such that the dimension of* $\pi_i(S)$ equals d*.*

Proof. Since S has dimension d, there exists a d-dimensional open cube C fully contained in S. Let $\vec{p}, \vec{r}_1, \ldots, \vec{r}_d$ be points in C such that the vectors[6] $\overrightarrow{pr_1}, \ldots, \overrightarrow{pr_d}$ are independent. Since $d < n$, there exists i, $1 \leq i \leq n$, such that $\vec{e}_i$ is not a linear combination of $\overrightarrow{pr_1}, \ldots, \overrightarrow{pr_d}$. Consider $\pi_i(S)$. Clearly, $\pi_i(C)$ is convex and open within $\mathbf{R}^n - 1$, because C is convex and open in $\mathbf{R}^n$. Let $\vec{q}, \vec{s}_1, \ldots, \vec{s}_d$ be the orthogonal projections on the ith coordinate hyperplane of $\vec{p}, \vec{r}_1, \ldots, \vec{r}_d$, respectively. We next show that $\overrightarrow{qs_1}, \ldots, \overrightarrow{qs_d}$ are linearly independent. Thereto, let $\lambda_1, \ldots, \lambda_d$ be real numbers and assume that $\lambda_1 \overrightarrow{qs_1} + \cdots + \lambda_d \overrightarrow{qs_d} = \vec{0}$. Let $\vec{u}$ be the unique point of $\mathbf{R}^n$ for which $\overrightarrow{pu} = \lambda_1 \overrightarrow{pr_1} + \cdots + \lambda_d \overrightarrow{pr_d}$. By the linearity of projection, $\pi_i(\overrightarrow{pu}) = \vec{0}$, whence $\overrightarrow{pu}$ is a multiple of $\vec{e}_i$, By choice of i, this multiple cannot be non-zero. Hence $\overrightarrow{pu} = \vec{0}$. From the linear independence of $\overrightarrow{pr_1}, \ldots, \overrightarrow{pr_d}$, it then follows that $\lambda_1 = \cdots = \lambda_d = 0$. Thus $\overrightarrow{qs_1}, \ldots, \overrightarrow{qs_d}$ are linearly independent. Clearly, an

[6] For $\vec{a}$ and $\vec{b}$ in $\mathbf{R}^n$, $\overrightarrow{ab}$ is defined as $\vec{b} - \vec{a}$.

open convex set of $\mathbf{R}^n$ containing $d+1$ points $q, s_1, \ldots, s_d$ such that $qs_1, \ldots, qs_d$ are linearly independent contains a d-dimensional open cube. Since $\pi_i(S)$ cannot contain an open cube of a strictly larger dimension, we have effectively shown that $\pi_i(S)$ is d-dimensional. □

Lemma 3.9 *The predicate* empty(S)*, with S is a semi-linear set of $\mathbf{R}^n$, and which evaluates to* true *if S is the empty relation, is definable in* FO + linear*.*

Proof. The FO + linear formula $\neg(\exists \vec{x})S(\vec{x})$ defines the predicate empty(S). □

Lemma 3.10 *The predicate* maxdim(S)*, with S is a semi-linear set of $\mathbf{R}^n$, and which evaluates to* true *if the dimension of S equals n, is definable in* FO + linear*.*

Proof. The FO + linear formula

$$(\exists \vec{x})(\exists \vec{\varepsilon})(\vec{\varepsilon} \neq \vec{0} \wedge (\forall \vec{y})(\vec{x} - \vec{\varepsilon} < \vec{y} < \vec{x} + \vec{\varepsilon} \Rightarrow S(\vec{y})))$$

expresses that S contains an open cube of maximal dimension. □

Lemma 3.11 *The predicate* $\max(d, d_1, \ldots, d_n)$*, $n \geq 2$, which evaluates to* true *if d is the maximum value of $d_1, \ldots, d_n$, is definable in* FO + linear*.*

Proof. The FO + linear-formula

$$(d = d_1 \vee \ldots \vee d = d_n) \wedge d \geq d_1 \wedge \ldots \wedge d \geq d_n$$

defines the predicate $\max(d, d_1, \ldots, d_n)$. □

We are now ready to give the proof of Theorem 3.6.

Proof of Theorem 3.6. The FO+linear formulae defining the predicates $\dim_n(S, d)$ are obtained inductively. By Lemmas 3.7, 3.9 and 3.10, the FO + linear formula

$$(d = -1 \wedge \text{empty}(S)) \vee (d = 1 \wedge \text{maxdim}(S)) \vee (d = 0 \wedge \neg\text{empty}(S) \wedge \neg\text{maxdim}(S))$$

clearly defines $\dim_1(S, d)$ in $\mathbf{R}$. Assume now that, in $\mathbf{R}^k$, $\dim_k(S, d)$ has been defined in FO + linear for $k < n$. Then, by the induction hypotheses and Lemmas 3.7, 3.8, 3.9, 3.10, and 3.11, the FO + linear formula

$$\begin{aligned}(d = n \wedge \text{maxdim}(S)) \vee (\neg\text{maxdim}(S) \wedge \\ \dim_{n-1}(\pi_1(S), d_1) \wedge \ldots \wedge \dim_{n-1}(\pi_n(S), d_n) \wedge \max(d, d_1, \ldots, d_n))\end{aligned}$$

defines $\dim_n(S, d)$ in $\mathbf{R}^n$. □

Many interesting queries can be expressed in a natural way using the dimension predicate, and are therefore also expressible in FO + linear, as is illustrated by the following list of sample queries.

- The Boolean query which decides whether a semi-linear set S is a line or a line segment, is expressible in FO + linear by the expression

$$\dim_n(S,1) \wedge (\forall \vec{x})(\forall \vec{y})(S(\vec{x}) \wedge S(\vec{y}) \Rightarrow S((\vec{x}+\vec{y})/2)).$$

- The Boolean query which decides whether a semi-linear set S consists only of lines and non-degenerated line segments is expressible in FO + linear by the expression

$$\dim(S,1) \wedge \neg(\exists \vec{x})(\exists \vec{\varepsilon})(S(\vec{x}) \wedge (\forall \vec{y})(\neg(\vec{y}=\vec{x}) \wedge \vec{x}-\vec{\varepsilon} < \vec{y} < \vec{x}+\vec{\varepsilon} \Rightarrow \neg S(\vec{y}))).$$

- The query which yields the k-dimensional component[7] of a semi-linear figure S is expressible in FO + linear in a straightforward manner.

- The Boolean query which decides whether the semi-linear set S represents a k-dimensional, convex figure is expressible in FO + linear by the expression

$$\dim(S,k) \wedge (\forall \vec{x})(\forall \vec{y})(S(\vec{x}) \wedge S(\vec{y}) \Rightarrow (\exists \vec{z})(S(\vec{z}) \wedge 2\vec{z} = \vec{x}+\vec{y})).$$

We are still far away from a precise insight into the nature of the queries expressible in FO + linear, however.

4 Limitations of FO + linear

Section 3 may have convinced the reader that FO + linear is a rich query language, suitable to accompany the linear database model. In this Section, we intend to moderate this positive perception of the query language FO + linear.

First, we must point out that Afrati et al. [1] have shown that FO + linear is *not* complete for the linear queries definable in FO + poly. More concretely, Afrati et al. proved the following result:

Proposition 4.1 [1] *The Boolean query on semi-linear sets S of $\mathbf{R}$ which decides whether there exist u and v in S with $u^2+v^2=1$, is not definable in* FO + linear.

Even though the query in Proposition 4.1 involves a non-linear computation in order to evaluate it, it is a linear query because it is a Boolean query, and therefore Proposition 4.1 suffices to establish the incompleteness of FO + linear for the linear queries definable in FO + poly. We shall denote the class of linear queries definable in FO + poly by FO + poly^{lin}.

Nevertheless, Proposition 4.1, by the somewhat artificial character of the query exhibited, does not provide us insight in the adequacy of FO + linear as linear query language.

Here, we develop a tool to show the non-expressibility in FO + linear of a whole range of linear queries in FO + poly^{lin}, by linking their non-expressibility in FO + linear to the non-definability by linear formula of certain related semi-algebraic sets. Definition 4.2 makes this link precise.

[7] The k-dimensional component of a semi-linear set S is the set of all points $\vec{p}$ of S for which there exists a neighborhood V in $\mathbf{R}^n$ such that, for each neighborhood $W \subseteq V$ of $\vec{p}$ in $\mathbf{R}^n$, $S \cap W$ has dimension k.

Definition 4.2 Let P be a semi-algebraic subset of $(\mathbf{R}^n)^m$, $m, n \geq 1$. Let k be such that $0 \leq k \leq m$. Furthermore assume that P and k are such that, for each sequence $\vec{u}_1, \ldots, \vec{u}_k$ in $\mathbf{R}^n$, and for all sequences $i_1, \ldots, i_k$ such that $\{\vec{u}_{i_1}, \ldots, \vec{u}_{i_k}\} = \{\vec{u}_1, \ldots, \vec{u}_k\}$, the following permutation invariance property holds for all $\vec{u}_{k+1}, \ldots, \vec{u}_m$ in $\mathbf{R}^n$:

$$(\vec{u}_1, \ldots, \vec{u}_k, \vec{u}_{k+1}, \ldots, \vec{u}_m) \in P \Leftrightarrow (\vec{u}_{i_1}, \ldots, \vec{u}_{i_k}, \vec{u}_{k+1}, \ldots, \vec{u}_m) \in P.$$

The query $Q_{P,k}$ of signature $[0, n] \to [0, n(m-k)]$ is now defined as follows. If S consists of at most k points of $\mathbf{R}^n$, say $S = \{\vec{u}_1, \ldots, \vec{u}_k\}$ ($\vec{u}_1, \ldots, \vec{u}_k$ not necessarily all distinct), then

$$Q_{P,k}(S) = \{(\vec{u}_{k+1}, \ldots, \vec{u}_m) \mid (\vec{u}_1, \ldots, \vec{u}_k, \vec{u}_{k+1}, \ldots, \vec{u}_m) \in P\};$$

otherwise $Q_{P,k}(S)$ is empty.

Observe that the invariance property assumed for P and k guarantees that $Q_{P,k}$ is a well-defined query expressible in FO + poly.

Example 4.3 We give some examples of sets P and corresponding queries $Q_{P,k}$ which will be used further on in this section.

1. Let P_1 be the set

   $$\{(\vec{u}_1, \ldots, \vec{u}_m) \in (\mathbf{R}^n)^m \mid \vec{u}_1, \ldots, \vec{u}_m \text{ are colinear}\},$$

   for appropriately chosen values of n and m. The set P_1 is obviously semi-algebraic; e.g., for $m = 3$, it is expressed by the real formula

   $$\vec{x}_2 = \vec{x}_3 \vee (\exists\lambda_1)(\exists\lambda_2)(\lambda_1 + \lambda_2 = 1 \wedge \vec{x}_1 = \lambda_1\vec{x}_2 + \lambda_2\vec{x}_3).$$

   Moreover, it satisfies Definition 4.2 for $k = m$. The associated query $Q_{P_1,m}$ can be interpreted as the Boolean query which decides whether a semi-linear set S consists of at most m colinear points.

2. Let P_2 be the set

   $$\{(\vec{u}_1, \ldots, \vec{u}_m) \in (\mathbf{R}^n)^m \mid \vec{u}_m \text{ is on the convex hull}^8 \text{ of } \{\vec{u}_1, \ldots, \vec{u}_{m-1}\}\},$$

   for appropriately chosen values of n and m. The set P_2 is semi-algebraic. To see this, first note that the real formula

   $$(\exists\lambda_1)\ldots(\exists\lambda_{m-1})\Big(\sum_{i=1}^{m-1}\lambda_i = 1 \wedge \lambda_1 \geq 0 \wedge \ldots \wedge \lambda_{m-1} \geq 0 \wedge \vec{x}_m = \sum_{i=1}^{m-1}\lambda_i\vec{x}_i\Big)$$

[8] The convex closure of a subset S of $\mathbf{R}^n$ is the smallest convex subset of $\mathbf{R}^n$ containing S. The convex hull of S is the boundary of the convex closure of S with respect to the topology of its affine support. The affine support of a subset S of $\mathbf{R}^n$ is defined as the smallest affine subspace of $\mathbf{R}^n$ containing S.

can be used to compute the convex *closure* of $m-1$ points. The convex hull, now, is the boundary of the convex closure with respect to the topology of its affine support. The boundary of a convex closed semi-algebraic set S with respect to the topology of its affine support can be computed using the following FO + linear expression:

$$S(\vec{x}) \wedge (\forall \vec{\varepsilon})(\vec{\varepsilon} \neq \vec{0} \Rightarrow (\exists \vec{y})(S(\vec{y}) \wedge \vec{x} - \vec{\varepsilon} < \vec{y} < \vec{x} + \vec{\varepsilon} \wedge (\exists \vec{z})(\vec{z} = 2\vec{x} - \vec{y} \wedge \neg S(\vec{z})))).$$

Moreover, the set P_2 satisfies Definition 4.2 for $k = m - 1$. The associated query $Q_{P_2,m-1}$ of type $[0,n] \to [0,n]$ can be interpreted as the linear query that associates with each semi-linear set S consisting of at most $m-1$ points the convex hull of S (which is also semi-linear), and with every other semi-linear set S the empty set.

3. Let P_3 be the set

$$\{(\vec{u}_1, \ldots, \vec{u}_m) \in (\mathbf{R}^n)^m \mid \vec{u}_m \text{ is on the affine support}^9 \text{ of } \{\vec{u}_1, \ldots, \vec{u}_{m-1}\}\},$$

for appropriately chosen values of n and m. The set P_3 is semi-algebraic, because it is expressed by the real formula

$$(\exists \lambda_1) \ldots (\exists \lambda_{m-1}) \Big(\sum_{i=1}^{m-1} \lambda_i = 1 \wedge \vec{x}_m = \sum_{i=1}^{m-1} \lambda_i \vec{x}_i \Big).$$

Moreover, it satisfies Definition 4.2 for $k = m-1$. The associated query $Q_{P_3,m-1}$ of type $[0,n] \to [0,n]$ can be interpreted as the linear query that associates with each semi-linear set S consisting of at most $m-1$ points the affine hull of S (which is also semi-linear), and with every other semi-linear set S the empty set.

4. Let P_4 be the set

$$\{(\vec{u}_1, \ldots, \vec{u}_m) \in (\mathbf{R}^n)^m \mid \vec{u}_m \text{ is in the (open) Voronoi cell of } \vec{u}_{m-1} \text{ with respect to } \vec{u}_1, \ldots, \vec{u}_{m-2}\}.$$

The point $\vec{u}_m$ belongs to the (open) Voronoi cell of $\vec{u}_{m-1}$ with respect to $\vec{u}_1, \ldots, \vec{u}_{m-2}$ if the condition

$$\bigwedge_{i=1}^{m-2} (u_{m1} - u_{m-1_1})^2 + \ldots + (u_{mn} - u_{m-1_n})^2 < (u_{m1} - u_{i1})^2 + \ldots + (u_{mn} - u_{in})^2$$

is satisfied. Hence, P_4 is semi-algebraic. Moreover, it satisfies Definition 4.2 for $k = m-2$. The associated query $Q_{P_4,m-2}$ of type $[0,n] \to [0,2n]$ can be interpreted as the linear query that associates with each semi-linear set S consisting of at most $m-2$ points pairs of points such that the latter belongs to the Voronoi cell of the former with respect to the points of S (this set of pairs is semi-linear), and with every other semi-linear set S the empty set.

[9] See footnote 8.

We now establish that the query $Q_{P,k}$ is not expressible in FO + linear as soon as the set P is not definable by linear formulae.

Theorem 4.4 *Let P be a semi-algebraic subset of $(\mathbf{R}^n)^m$, $m, n \geq 1$, let k be such that $0 \leq k \leq m$, and let P and k satisfy the conditions of Definition 4.2. If P is not definable by linear formulae, then the following holds:*

1. *The query $Q_{P,k}$ is not expressible in* FO + linear.
2. *If Q is a linear query of type $[0, n] \to [0, n(m-k)]$ such that, for every semi-linear set S of $\mathbf{R}^n$, $Q(S) = Q_{P,k}(S)$ if $Q_{P,k}(S)$ is not empty, then Q is not expressible in* FO + linear.

Proof.

1. Assume, to the contrary, that the query $Q_{P,k}$ is expressible in FO + linear. Then there exists an FO + linear formula $\varphi_{P,k}(R; \vec{x}_{k+1}, \ldots, \vec{x}_m)$, with R an appropriate predicate name, such that, for each semi-linear set S of $\mathbf{R}^n$, $Q_{P,k}(S) = \{(\vec{u}_{k+1}, \ldots, \vec{u}_m) \mid \varphi_{P,k}(S; \vec{u}_{k+1}, \ldots, \vec{u}_m)\}$. We now argue that the predicate name R must effectively occur in $\varphi_{P,k}$. If this were not the case, then the query associated with $\varphi_{P,k}$ would be independent of the input, i.e., a constant function. This constant function must return the empty set, since $Q_{P,k}$ by definition returns the empty set on all inputs containing more than k points. However, $Q_{P,k}$ cannot return the empty set on *every* input unless P is the empty set, which is obviously definable by a linear formula, contrary the hypothesis of the theorem. Thus R must occur in $\varphi_{P,k}$.

   Given the formula $\varphi_{P,k}$, we construct a new formula $\hat{\varphi}_{P,k}$, as follows. Let $\vec{x}_1, \ldots, \vec{x}_k$ be variables that do not occur in $\varphi_{P,k}$. Now replace every literal of the form
   $$R(\vec{z})$$
   in $\varphi_{P,k}$ by the formula
   $$\vec{z} = \vec{x}_1 \vee \cdots \vee \vec{z} = \vec{x}_k.$$
   Observe that the formula $\hat{\varphi}_{P,k}$ is a linear formula with free variables $\vec{x}_1, \ldots, \vec{x}_m$. Our claim is that the formula $\hat{\varphi}_{P,k}$ defines the set P, a contradiction with the hypothesis of the theorem. To substantiate our claim, we consider an m-tuple $(\vec{u}_1, \ldots, \vec{u}_m) \in (\mathbf{R}^n)^m$. From the definition of $Q_{P,k}$ and $\varphi_{P,k}$, we have
   $$(\vec{u}_1, \ldots, \vec{u}_m) \in P \Leftrightarrow (\vec{u}_{k+1}, \ldots, \vec{u}_m) \in Q_{P,k}(\{\vec{u}_1, \ldots, \vec{u}_k\}),$$
   whence
   $$(\vec{u}_1, \ldots, \vec{u}_m) \in P \Leftrightarrow \varphi_{P,k}(\{\vec{u}_1, \ldots, \vec{u}_k\}; \vec{u}_{k+1}, \ldots, \vec{u}_m).$$
   It follows from the construction of $\hat{\varphi}_{P,k}$ from $\varphi_{P,k}$ that
   $$(\vec{u}_1, \ldots, \vec{u}_m) \in P \Leftrightarrow \hat{\varphi}_{P,k}(\vec{u}_1, \ldots, \vec{u}_m).$$

2. Assume that Q is expressible in FO + linear. Then there exists a formula

$$\varphi_Q(R; \vec{x}_{k+1}, \ldots, \vec{x}_m)$$

that defines Q, where R stands for the input predicate. Given φ_Q, we can construct the formula $\hat{\varphi}_Q$:

$$\hat{\varphi}_Q(R; \vec{x}_{k+1}, \ldots, \vec{x}_m) \Leftrightarrow (|R| \leq k \wedge \varphi_Q(R; \vec{x}_{k+1}, \ldots, \vec{x}_m)) \vee (|R| > k \wedge \mathit{false}).$$

It is obvious that this expression for $\hat{\varphi}_Q$ can be translated into proper FO+linear syntax. It now follows from the properties of Q that the FO + linear-formula $\hat{\varphi}_Q$ expresses the query $Q_{P,k}$, which is impossible by the first part of the theorem. □

To allow ourselves to apply Theorem 4.4, we first establish that the semi-algebraic sets in Example 4.3 are not definable by linear formulae for most values of m and n.

Proposition 4.5 *The sets P_1, P_2, P_3, P_4 are not definable by linear formulae if $n \geq 2$ and $m \geq 3$.*

Proof.

1. We first show that P_1 is not definable by a linear formula. Assume to the contrary that P_1 is definable by a linear formula for some $n \geq 2$ and some $m \geq 3$. Then, clear, P_1 is also definable by a linear formula for $n = 2$ and $m = 3$. Let $\mathit{colinear}(x_1, x_2, y_1, y_2, z_1, z_2)$ denote this formula. We now show that there exists a linear formula $\mathit{product}(x, y, z)$, with x, y, z real variables, equivalent to the real formula $z = xy$, an obvious contradiction. From the geometric construction of the product shown in Figure 2, it follows that

$$\begin{aligned}(x = 0 \wedge z = 0) \vee (y = 0 \wedge z = 0) \vee (y = 1 \wedge z = x) \vee \\ \neg(\exists v)(\exists w)(\mathit{colinear}(x, 0, 0, 1, v, w) \wedge \mathit{colinear}(z, 0, 0, y, v, w))\end{aligned}$$

is the desired linear formula.

2. The semi-algebraic set P_2 is not definable by a linear formula since P_1 is not: indeed, 3 points are colinear if and only one of them is on the convex hull of the other two.

3. The semi-algebraic set P_3 is not definable by a linear formula since P_1 is not: indeed, 3 points are colinear if and only one of them is on the affine support of the other two.

4. The semi-algebraic set P_4 is not definable by a linear formula since P_1 is not: indeed, 3 points are colinear if and only if the complement of the union of the (open) Voronoi cells of each of the points with respect to the other two consists of two parallel hyperplanes. □

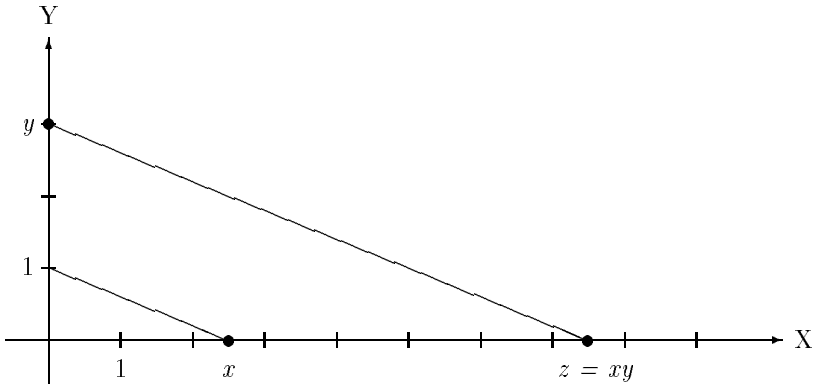


Figure 2: Geometric construction of the product $z = xy$.

Theorem 4.4 and Proposition 4.5 yield the following corollary, the proof of which is immediate from the former:

Theorem 4.6 *1. The Boolean query of type* $[0,n] \to [0,0]$ *deciding whether a semi-linear subset of* $\mathbf{R}^n$ *is contained in a line is not expressible in* FO + linear.

2. The linear query of type $[0,n] \to [0,n]$ *computing the convex hull of a semi-linear subset of* $\mathbf{R}^n$ *is not expressible in* FO + linear.

3. The linear query of type $[0,n] \to [0,n]$ *computing the affine support of a semi-linear subset of* $\mathbf{R}^n$ *is not expressible in* FO + linear.

4. The linear query of type $[0,n] \to [0,2n]$ *computing all pairs of points of* $\mathbf{R}^n$ *such that that the latter is in the Voronoi cell of the former with respect to a semi-linear subset of* $\mathbf{R}^n$ *is not expressible in* FO + linear.

Obviously, Theorem 4.4 can be used to show the non-expressibility of many more linear queries. For instance, it can be used to prove Proposition 4.1 as well as the non-expressibility in FO + linear of several other Boolean queries. Just as Boolean queries restricted to semi-linear sets are necessarily linear, (real-valued) aggregate queries, such as volume or surface, restricted to semi-linear sets are necessarily linear.[10] We conclude this section by exhibiting an aggregate query not expressible in FO + linear.

Example 4.7 Let S be a semi-linear subset of $\mathbf{R}^n$. We define the *diameter* of S denoted $\emptyset(S)$, as the maximal (Euclidean) distance between two points of S. Obviously, the diameter of a semi-linear set can be seen as a aggregate query of type $[0,n] \to [0,1]$. Let

$$P_5 = \{(\vec{u}_1, \vec{u}_2, (d, \underbrace{0, \ldots, 0}_{n-1})) \mid \emptyset(\{\vec{u}_1, \vec{u}_2\}) = d\}.$$

[10] This statement must be moderated to the extent that the real number returned by the aggregate query must be finitely representable for it to fully hold. In the light of earlier remarks, we shall not be concerned with this restriction in this paper.

It is easily seen that P_5 is a semi-algebraic set that satisfies the conditions of Definition 4.2 for $k = 2$. Obviously, P_5 is not definable by a linear formula, because an appropriate intersection of P_5 with affine spaces yields a circle. Thus $Q_{P,2}$ is not expressible in FO + linear, whence the linear query of type $[0, n] \to [0, n]$ computing the singleton

$$(\emptyset(S), \underbrace{0, \ldots, 0}_{n-1})$$

upon a semi-linear subset S of $\mathbf{R}^n$ as input. From this result, it is easily seen that the diameter query of type $[0, n] \to [0, 1]$ is not expressible either. □

5 Extending FO + linear

Although a wide range of useful, complex linear queries is expressible in FO + linear, as shown in Section 3, there are several other, practically relevant linear queries not expressible in FO + linear, as shown in Section 4. Therefore, it is important to search for linear query languages that capture these queries. Without such languages, we would indeed be hard-pressed to substantiate the claim that the linear model is to be adopted as the fundamental model for applications involving linear geometric objects.

A first approach towards the problem raised above is searching for a language that is sound and complete for the FO + poly^{lin} queries, i.e., that can express precisely the linear queries expressible in FO + poly. The most straightforward way to obtain such a query language is to discover an algorithm to decide whether a FO + poly formula induces a linear query. Unfortunately, such an algorithm does not exist, as shown by the following theorem.

Theorem 5.1 *It is undecidable whether a* FO + poly *formula induces a linear query.*

Proof. The proof of Theorem 5.1 is a variation of a proof by Paredaens et al. concerning undecidability of genericity in FO + poly (Theorem 1, pp. 285 of [33]). The $\forall^*$-fragment of number theory is undecidable since Hilbert's 10th problem can be reduced to it. Now, encode a natural number n by the one-dimensional semi-algebraic set $enc(n) := \{0, \ldots, n\}$, and encode a vector of natural numbers $(n_1, \ldots, n_k)$ by $enc(n_1) \cup (enc(n_2) + n_1 + 2) \cup \ldots \cup (enc(n_k) + n_1 + 2 + \cdots + n_{k-1} + 2)$. The corresponding decoding is first-order expressible. We then reduce a $\forall^*$-sentence $(\forall \vec{x})\varphi(\vec{x})$ of number theory to the following query of signature $[0, 1] \to [0, 2]$:

if R encodes a vector $\vec{x}$ **then if** $\varphi(\vec{x})$ **then** $\emptyset$ **else** $\{(u, v) \mid u^2 + v^2 = 1\}$ **else** $\emptyset$.

This query is definable in FO + poly and induces a linear query if and only if the $\forall^*$-sentence is valid. □

Theorem 5.1 shows that a top-down approach to discover a useful linear subquery language is difficult. Observe that Theorem 5.1 does not rule out that one can isolate a subset of the FO + poly formulae which expresses precisely the FO + poly^{lin} queries, in the same way that the undecidability of domain independence in the relational calculus is not in contradiction with the existence of a sub-language of

the relational calculus which precisely expresses the domain-independent relational calculus queries. [39]

In this section, we therefore take a bottom-up approach to discover restrictions of FO + poly^{lin} that are strictly more expressive than FO + linear. The basic idea is to extend FO + linear with certain linear operators, such as the colinearity or the convex-hull query, or any of the other linear queries listed in Theorem 4.6.

However, we can*not* achieve our goal by adding the corresponding predicates to FO + linear. Indeed, from the proof of Proposition 4.5, it follows that, e.g., adding a predicate $colinear(\vec{x}, \vec{y}, \vec{z})$, which evaluates to *true* if its arguments are colinear points, would yield a language equivalent to FO+poly, as the product of real numbers would become definable. Obviously, we need a less liberal syntax to ensure that the extensions of FO + linear envisaged remain sound with respect to the FO + poly^{lin} queries.

We now proceed with showing how FO + linear can effectively be extended with linear operators in a sound way. The subtle point in the definition of our extensions is that we disallow free *real* variables in set terms.

An *operator* is defined to be an FO + poly^{lin} query. The signature of an operator is the signature of that query.

Let $\mathcal{O}$ be a set of operator names O typed with a signature, each of which represents an operator op(O) of the same signature.[11]

The query language FO + linear + $\mathcal{O}$ is then defined as an extension of FO + linear, as follows. First, we extend the terms of FO + linear with *set terms*:

- If φ is an FO + linear + $\mathcal{O}$ formula with n free real variables $x_1, \ldots, x_n$ and m free value variables $v_1, \ldots, v_m$, and if $k \leq m$, then

$$\{(v_1, \ldots, v_k, x_1, \ldots, x_n) \mid \varphi(v_1, \ldots, v_m, x_1, \ldots, x_n)\}$$

  is a *set term* of type $[k, n]$. Observe that of the value variables, $v_{k+1}, \ldots, v_m$ occur *free*, while *all* real variables, $x_1, \ldots, x_n$, occur *bounded* in the set term.[12]

- If O is an operator name in $\mathcal{O}$ of type $[m_1, n_1, \ldots, m_k, n_k] \rightarrow [m, n]$, and $S_1, \ldots, S_k$ are set terms of types $[m_1, n_1], \ldots, [m_k, n_k]$, respectively, then

$$O(S_1, \ldots, S_k)$$

  is a *set term* of type $[m, n]$ with as free variables those in the union of all free variables in S_1 through S_k (which are all value variables).

Finally, we extend the atomic formulae of FO + linear:

- Let S be a set term of type $[m, n]$. Then $S(v_1, \ldots, v_m, x_1, \ldots, x_n)$, with $v_1, \ldots, v_m$ value variables and $x_1, \ldots, x_n$ real variables, is an *atomic formula* with as free variables $v_1, \ldots, v_m$, $x_1, \ldots, x_n$, and the free (value) variables of S.

[11] To be practically relevant, the set $\mathcal{O}$ must be finitely representable.

[12] Observe that this definition allows us to interpret a predicate name R of type $[k, n]$ as a set term of type $[k, n]$.

Semantically, when actual values are substituted for the free variables, a set term of type $[m, n]$ represents a subset of $D^m \times \mathbf{R}^n$. Now consider an atomic formula of the form $S(v_1, \ldots, v_m, x_1, \ldots, x_n)$. When actual values are substituted for the free variables, this atomic formula evaluates to *true* if the evaluation of $(v_1, \ldots, v_m, x_1, \ldots, x_n)$ belongs to the set represented by the set term S. The full semantics of FO+linear+$\mathcal{O}$ is now straightforward to define.

The following soundness property is now easily shown by structural induction:

Theorem 5.2 *The query language* FO + linear + $\mathcal{O}$ *only expresses* FO + poly^{lin}*-definable queries.*

The syntactic restriction that set terms do not contain free real variables is essential for Theorem 5.2 to hold.

Without going into details, we mention that it is possible to define an algebraic query language equivalent to FO + linear + $\mathcal{O}$ by extending the linear algebra [40] with the operators represented by $\mathcal{O}$. This equivalence result forms a theoretical justification for the approach Güting has taken with the development of the ROSE-algebra [22, 23, 25, 24, 26], which is extending the relational algebra with a class of spatial operators.

We conclude this section by giving an example of an FO + linear + $\mathcal{O}$ query language in which we can express the *colinearity* and *convex hull* queries described in Theorem 4.6. Thereto, let $\mathcal{O}$ be an infinite set of operator names segment$_n$ of signature $[0, n] \to [0, n]$, $n \geq 0$, and associate with each operator name segment$_n$ the operator op(segment$_n$) defined by

$$\text{op}(\text{segment}_n)(S) = \{\vec{x} \in \mathbf{R}^n \mid (\exists \vec{y})(\exists \vec{z})(S(\vec{y}) \wedge S(\vec{z}) \wedge \vec{x} \in [\vec{y}, \vec{z}]\})$$

for each semi-linear set S of $\mathbf{R}^n$. Now let R be a predicate representing a semi-linear set of $\mathbf{R}^n$. The FO + linear + $\mathcal{O}$ formula

$$\underbrace{\text{segment}_n(\text{segment}_n(\ldots \text{segment}_n}_{n \text{ times}}(R) \cdots))(\vec{x})$$

computes the the convex closure of the set represented by R. The convex hull is then easily obtained as the boundary of the convex closure with respect to the topology of its affine support. Using the convex-closure query as a macro, we can express the colinearity query by the following FO + linear + $\mathcal{O}$ formula:

$$(\exists d)(\dim(\{\vec{x} \mid \text{convex-closure}(S)(\vec{x})\}, d) \wedge d \leq 1).$$

6 Conclusions

In this paper we studied languages that define FO + poly^{lin} queries. Amongst these languages, the most natural one is FO + linear. For this language, we showed that, on the one hand, several useful complex linear queries, such as the dimension query, can be expressed in it, but, on the other hand, that equally important linear queries, such as deciding colinearity or computing the convex hull, are not expressible. These

latter results led us to the introduction of extensions of FO + linear with FO + poly^{lin}-definable operators. The crucial part of this construction was requiring that operators can only be applied to set terms *without* free real variables. From our exposition, it follows that lifting this restriction can destroy the soundness of the query language.

We conclude by mentioning the two most prominent open problems raised by this paper:

1. Does there exist a syntactic restriction on FO + poly formulae that yields a sublanguage of FO + poly which is sound and complete for the FO + poly^{lin}-definable queries?

2. Does there exist an extension of FO + linear (or other sublanguages of FO + poly) with operators that yields soundness and completeness with respect to the FO + poly^{lin}-definable queries?

References

[1] F. Afrati, S. Cosmadakis, S. Grumbach, and G. Kuper, "Linear versus polynomial constraints in database query languages," in Proceedings *2nd International Workshop on Principles and Practice of Constraint Programming* (Rosario, WA), A. Borning, ed., in *Lecture Notes in Computer Science*, vol. 874, Springer-Verlag, Berlin, 1994, pp. 181–192.

[2] F. Afrati, T. Andronikos, and T. Kavalieros, "On the expressiveness of first-order constraint languages," in Proceedings *Workshop on Constraint Databases and their Applications*, 1995.

[3] 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., in *Lecture Notes in Computer Science*, vol. 525, Springer-Verlag, Berlin, 1991, pp. 299–319.

[4] M. Benedikt, G. Dong, L. Libkin, and L. Wong, "Relational expressive power of constraint query languages," in Proceedings *15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Montreal, Canada), 1996, pp. 5–16.

[5] 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.

[6] A. Brodsky, J. Jaffar, and M.J. Maher, "Toward practical constraint databases," in Proceedings *19th International Conference on Very Large Databases* (Dublin, Ireland), 1993, pp. 567–580.

[7] A. Brodsky and Y. Kornatzky, "The LyriC language: querying constraint objects," in Proceedings *Post-ILPS'94 Workshop on Constraints and Databases* (Ithaca, NY), 1994.

[8] J.-H. Byon and P. Revesz, "DISCO: a constraint database system with sets," in Proceedings *Workshop on Constraint Databases and Applications* (Friedrichshafen, Germany), G. Kuper and M. Wallace, eds., in *Lecture Notes in Computer Science*, vol. 1034, Springer-Verlag, 1996, pp. 68–83.

[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] J. Chomicki, D.Q. Goldin, and G.M. Kuper, "Variable independence and aggregation closure," in Proceedings *15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Montreal, Canada), 1996, pp. 40–48.

[11] E. Clementini, P. Di Felice, and P. van Oosterom, "A small set of formal topological relationships suitable for end-user interaction," in *Advances in Spatial Databases*, Proceedings 3th International Symposium on Large Spatial Databases, D. Abel and B.C. Ooi, eds., in *Lecture Notes in Computer Science*, vol. 692, Springer-Verlag, Berlin, 1993, pp. 277–295.

[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., in *Lecture Notes in Computer Science*, vol. 367, Springer-Verlag, Berlin, 1989, pp. 457–472.

[14] M.J. Egenhofer and J. Herring, "A mathematical framework for the definition of topological relationships," in Proceedings *4th International Symposium on Spatial Data Handling* , K. Brassel and H. Kishimoto, eds., Zurich, Switzerland, 1990, pp. 803–813.

[15] M.J. Egenhofer, "Reasoning about binary topological relations," in *Advances in Spatial Databases*, Proceedings 2nd Symposium on Very Large Spatial Databases, O. Günther and H.-J. Schek, eds., in *Lecture Notes in Computer Science*, vol. 525, Springer-Verlag, Berlin, 1991, pp. 143–160.

[16] M.J. Egenhofer, "Why not SQL!", *International Journal on Geographical Information Systems*, 6:2, 1992, pp. 71–85.

[17] M.J. Egenhofer, "Spatial SQL: a query and presentation language," *IEEE transactions on Knowledge and Data Engineering*, 6:1, 1994, pp. 86–95.

[18] S. Grumbach and J. Su, "Towards practical constraint databases," in Proceedings *15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Montreal, Canada), 1996, pp. 28–39.

[19] S. Grumbach, J. Su, and C. Tollu, "Linear constraint query languages: expressive power and complexity," in Proceedings *Logic and Computational Complexity Workshop* (Indianapolis, IN), D. Leivant, ed., 1994.

[20] O. Günther, ed., "Efficient Structures for Geometric Data Management," in *Lecture Notes in Computer Science*, vol. 337, Springer-Verlag, Berlin, 1988.

[21] O. Günther, "Research issues in spatial databases," *SIGMOD Record*, 19:4, 1990, pp. 61–68.

[22] R.H. Güting, "Geo-Relational Algebra: a model and query language for geometric database systems," in *Advances in Database Technology—EDBT '88*, Proceedings International Conference on Extending Database Technology (Venice, Italy), J.W. Schmidt, S. Ceri, and M. Missikoff, eds., in *Lecture Notes in Computer Science*, vol. 303, Springer-Verlag, Berlin, 1988, pp. 506–527.

[23] R.H. Güting, "GRAL: An extensible relational database system for geometric applications," in Proceedings *15th International Conference on Very Large Databases* (Amsterdam, the Netherlands), 1989, pp. 33–34.

[24] R.H. Güting, and M. Schneider, "Realms: a foundation for spatial data types in database systems," in Proceedings *3rd International Conference on Very Large Databases* (Singapore), 1993, pp. 14–35.

[25] R.H. Güting, "An introduction to spatial database systems," *VLDB-Journal*, 3:4, 1994, pp. 357–399.

[26] R.H. Güting, T. de Ridder, and M. Schneider, "Implementation of the ROSE Algebra: efficient algorithms for realm-based spatial data types," in *Advances in Spatial Databases*, Proceedings 4th International Symposium on Large Spatial Databases, M.J. Egenhofer and J.R. Herring, eds., in *Lecture Notes in Computer Science*, vol. 951, Springer-Verlag, Berlin, 1995, pp. 216–239.

[27] P.C. Kanellakis and D.Q. Goldin, "Constraint programming and database query languages," in Proceedings *2nd Conference on Theoretical Aspects of Computer Software* (Sendai, Japan), M. Hagiya and J.C. Mitchell, eds., in *Lecture Notes in Computer Science*, vol. 789, Springer-Verlag, Berlin, 1994, pp. 96–120.

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

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

[30] G. Kuper, "On the expressive power of the relational calculus with arithmetic constraints," in Proceedings *3rd International Conference on Database Theory* (Paris, France), S. Abiteboul and P.C. Kanellakis, eds., in *Lecture Notes of Computer Science*, vol. 470, Springer-Verlag, Berlin, pp. 202–214.

[31] C. B. Medeiros and F. Pires, "Databases for GIS," *SIGMOD Record*, 23:1, 1994, pp. 107–115.

[32] B.C. Ooi, "Efficient query processing in geographic information systems," G. Goos and J. Hartmanis, eds., in *Lecture Notes in Computer Science*, vol. 471, Springer Verlag, Berlin.

[33] 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.

[34] J. Paredaens, "Spatial databases, the final frontier," in Proceedings *5th International Conference on Database Theory* (Prague, Czech Republic), G. Gottlob, and M.Y. Vardi, eds., in *Lecture Notes of Computer Science*, vol. 893, Springer-Verlag, 1995, pp. 14–32.

[35] J. Paredaens, J. Van den Bussche, and D. Van Gucht, "First-order queries on finite structures over the reals," in Proceedings *10th IEEE Symposium on Logic in Computer Science* (Washington, DC), 1995, pp. 79–87, full version to appear in *SIAM Journal on Computing*.

[36] 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 Record*, 23:1, 1994, pp. 35–51.

[37] P. Svensson and Z. Huang, "GEO-SAL: a query language for spatial data analysis," in *Advances in Spatial Databases*, Proceedings 2nd Symposium on Very Large Spatial Databases, O. Günther and H.-J. Schek, eds., in *Lecture Notes in Computer Science*, vol. 525. Springer-Verlag, Berlin, 1991, pp. 119–140.

[38] A. Tarski, "A Decision Methos for Elementary Algebra and Geometry," University of California Press, 1951.

[39] J. D. Ullman, "Principles of Database Systems," 2nd edition, Pitman Publishing, London, 1982.

[40] L. Vandeurzen, M. Gyssens, and D. Van Gucht, "On the desirability and limitations of linear spatial query languages," in *Advances in Spatial Databases*, Proceedings 4th International Symposium on Large Spatial Databases, M.J. Egenhofer and J.R. Herring, eds., in *Lecture Notes in Computer Science*, vol. 951, Springer Verlag, Berlin, 1995, pp. 14–28.

[41] L. Vandeurzen, M. Gyssens, and D. Van Gucht, "On query languages for linear queries definable with polynomial constraints," in Proceedings *2nd International Conference on Principles and Practice of Constraint Programming*, E.C. Freuder, ed., in *Lecture Notes in Computer Science*, vol. 1118. Springer Verlag, Berlin, 1996, pp. 468–481.