ELSEVIER

Theoretical Computer Science 254 (2001) 423–463

Theoretical Computer Science

www.elsevier.com/locate/tcs

On the expressiveness of linear-constraint query languages for spatial databases ☆

Luc Vandeurzen, Marc Gyssens[a,*,1], Dirk Van Gucht[b]

[a] Department WNI, University of Limburg (LUC), Universitaire Campus, B-3590 Diepenbeek, Belgium
[b] Computer Science Department, Indiana University, Bloomington, IN 47405-4101, USA

Received March 1999; revised July 1999
Communicated by M. Nivat

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 in FO + linear of a whole class of linear queries that are related to sets not definable by linear formulae, a result which demonstrates the need for more expressive linear query languages. We present a method to extend FO + linear with operators in a sound way with respect to the linear queries expressible in FO + poly, and argue its validity by comparing it to another paradigm for enriching FO + linear. Whether any of the proposed extensions is complete for the linear queries definable in FO + poly remains open. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Spatial databases; Constraint databases; Linear constraints; Query languages; Expressiveness

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

☆ 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, MA, USA, August 1996).

* Corresponding author.

E-mail addresses: luc.vandeurzen@luc.ac.be (L. Vandeurzen), marc.gyssens@luc.ac.be (M. Gyssens), vgucht@cs.indiana.edu (D. Van Gucht).

[1] Part of the work of this author was conducted at and supported by the University of Antwerp.

data. Typical examples of such applications are geographic information systems (GIS), geometric modeling systems (CAD), and temporal databases (see, e.g., [38, 43, 39] 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, 24, 37]. 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: datatype-based models [3, 17, 25–29, 45] and constraint-based models [32, 33].

Datatype-based models extend the relational database model with a fixed set of spatial data types, typically points, lines, and polygons. As a consequence, geometric 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 a Euclidean space of some fixed, generally low, dimension. Query languages for datatype-based languages are essentially relational algebra extended with a fixed set of geometric 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 geometric operators can be computed as efficient as possible using techniques from computational geometry. While this approach guarantees very good performance for several applications, the major drawback of datatype-based models is that there is no single set of data types and geometric operators known to serve all purposes well.

The constraint-based approach was first proposed in 1990 in the seminal work of Kuper et al. [33]. 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 geometric 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 and for linear constraints, because the corresponding formulae can be replaced by equivalent, effectively computable quantifier-free formulae. 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 datatype-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. A lot of effort has gone into the study of the expressive power of database query languages based on linear and polynomial constraints (e.g., [1, 2, 4, 5, 11, 13, 19, 22, 30, 35, 42, 44, 48–50]).

Looking at the constraint-based models 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 are on-going (e.g., [7–9, 20, 21]).

Since they are both interesting from a theoretical and an implementational point of view, linear-constraint databases are the focus of this paper. More in particular, we study the strengths and weaknesses of FO + linear, first-order logic extended with linear constraints, the basic language of linear-constraint databases. Our contribution is threefold:

1. We identify a collection of general queries expressible in FO + linear, among which several queries of a topological nature. In particular, we show that the dimension query is expressible in FO + linear, which in turn yields the expressibility of several other important linear queries. To further illustrate the expressive power of FO + linear, we express convexity, parallelism, and orthogonality, as well as several intricate linear queries on one-dimensional spatial databases.
2. We present a general theorem stating that the non-definability in FO + linear of certain sets of points can be lifted to the non-expressibility of closely related linear queries. This theorem provides us with a technique to prove the inexpressibility of linear queries in FO + linear. Using this technique, we show that several important concrete 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 linear geometric operators in a sound way. (This is a non-trivial matter, as naive extensions of FO + linear easily yield non-linear languages.) The new query languages thus obtained can be seen as a bridge between constraint-based query languages and datatype-based query languages. We discuss the viability of the paradigm we propose and compare it to another paradigm for enriching FO + linear.

The paper is organized as follows. In Section 2, we review the polynomial and the linear database model. In Section 3, we illustrate the expressiveness of FO + linear by exhibiting several practical, general-purpose queries which can be stated in FO + linear. 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 by establishing a theorem that lifts the non-definability of certain point sets to the non-expressibility of closely related linear queries. We exhibit several examples of important linear queries which are proven to be inexpressible in FO + linear using the above-mentioned theorem. In Section 5, we remedy the shortcomings of FO + linear by providing a method to extend FO + linear with linear geometric operators in a sound way. We exhibit an example of such an extended language, and link it with related work on methods to extend FO + linear. 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 [33]. In particular, we explain the notion of query in the context of these database 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 submodel 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 with real algebraic [2] coefficients over the *real variables* $x_1, x_2, x_3, \ldots$, i.e.,

- if p and q are polynomials with real algebraic coefficients over the variables $x_1, \ldots, x_n$, then $p\,\theta\,q$, with θ in $\{=, <, >, \leqslant, \geqslant, \neq\}$, is a real formula with free variables $x_1, \ldots, x_n$;
- if φ and ψ are real formulae with free variables free(φ) and free(ψ), respectively, then $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\neg\varphi$ are real formulae with free variables free(φ) $\cup$ free(ψ); and
- if x is a real variable and φ is a real formula with free variables free(φ), then $(\exists x)\varphi$ is a real formula with free variables free(φ) $- \{x\}$.

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

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

in n-dimensional Euclidean space $\mathbb{R}^n$ 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. Consequently, $\neg(\boldsymbol{x} = \boldsymbol{0})$ indicates that $\boldsymbol{x}$ is not the origin of the coordinate system, whereas $\boldsymbol{x} \neq \boldsymbol{0}$ denotes that *none* of the coordinates of $\boldsymbol{x}$ equals 0. For $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$, the product $\boldsymbol{x}.\boldsymbol{y}$ equals $x_1 y_1 + \cdots + x_n y_n$. As usual, $\varphi \Rightarrow \psi$, $\varphi \Leftrightarrow \psi$, and $(\forall x)\varphi$ will be used as abbreviations for $\neg\varphi \vee \psi$, $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$, and $\neg(\exists x)\neg\varphi$, respectively. We use the customary precedence rules for omitting parentheses.

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] Real formula can also be defined by requiring the coefficients to be integers or rationals, without changing their defining power.

Example 2.1. The two real formulae
- $(\exists x_3)(\exists x_4)(x_3^2 + x_4^2 = 100 \wedge (x_3 - x_1)^2 + (x_4 - x_2)^2 < 1)$ and
- $x_1^2 + x_2^2 > 81 \wedge x_1^2 + x_2^2 < 121$

define the same area in the plane, namely an open annulus with inner radius 9 and outer radius 11.

By the quantifier-elimination theorem of Tarski [46], 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 and decidability of membership of a semi-algebraic set.

In essence, the polynomial database model is an extension of the relational database model in which a relation, besides columns that store values of non-spatial data types, can have one extra column of spatial type. In the spatial column, (finitely representable) real formulae are stored, which represent (possibly infinite) point sets, which may be unbounded. In the next two paragraphs, we give the formal definitions.

A *polynomial database scheme*, $\mathscr{S}$, 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. A *syntactic database instance* is a mapping, $\mathscr{I}$, assigning to each relation name, R, of a scheme, $\mathscr{S}$, a syntactic relation $\mathscr{I}(R)$ of the same type.

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

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

This subset of $D^m \times \mathbb{R}^n$ 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, $\mathscr{I}$, over a database scheme, $\mathscr{S}$, is the mapping, I, assigning to each relation name, R, in $\mathscr{S}$ the semantic relation $I(\mathscr{I}(R))$.

If in the type of the above database scheme $\mathscr{S}$ (relation name R) $m_1,\ldots,m_k$ (m) equal(s) 0, then the database (relation) concerned is called *purely spatial*. While in practical applications, spatial data will almost always be linked to non-spatial data, the presence of non-spatial does not add anything significant to the theoretical development, but does make the exposition more complicated. For this reason, we shall often restrict ourselves to purely spatial databases in the sequel.

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 [10, 31].

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 et al. [41]. Therefore, we shall only summarize their main conclusions here:

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

If $m_1,\ldots,m_k$ and m all equal 0, the query is called *purely spatial.* As will be the case for databases and relations, we shall often restrict ourselves to purely spatial queries in the sequel.

Queries of type $[m_1,n_1;\ldots;m_k,n_k]\to[0,0]$ can be interpreted as *Boolean queries*, as the only relations of type $[0,0]$ are the empty relation, which can be interpreted as **false**, and the singleton relation containing the 0-ary tuple $()$, which can be interpreted as **true**.

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 $R(v_1,\ldots,v_n;x_1,\ldots,x_m)$, 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
3. atomic formulae of the form $v_1=v_2$, with v_1 and v_2 non-spatial variables;
4. existential (and universal) 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 of type $[1,2]\to[2,0]$ can be expressed as

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

in FO + poly.

Due to the existence of quantifier elimination algorithms for real formula, every FO + poly query is effectively computable, and returns a polynomial output upon a polynomial input [33].

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 polynomials, i.e., polynomials of the form $a_0 + a_1x_1 + \cdots + a_nx_n$, with $a_0, a_1, \ldots, a_n$ real algebraic coefficients and $x_1, \ldots, x_n$ real variables, are called *linear formulae.* [3] Point sets defined by linear formulae are called *semi-linear sets.*

Günther [23] introduced *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 negation, conjunction, and disjunction occurring in linear formulae can be interpreted as the set operations complementation, union, and intersection, and existential quantification can be interpreted as geometric projection. Therefore, it is easy to see that semi-linear sets and polyhedral chains define the same class of geometric figures. Bounded semi-linear sets can be characterized as finite unions of open polytopes. [4] From this perspective, it follows that semi-linear sets cover all popular two- and three-dimensional spatial data types in existing models.

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

Example 2.3. The example in Fig. 1 shows a linear database of type $[1,2;1,2;1,2]$ representing geographic information about Belgium.

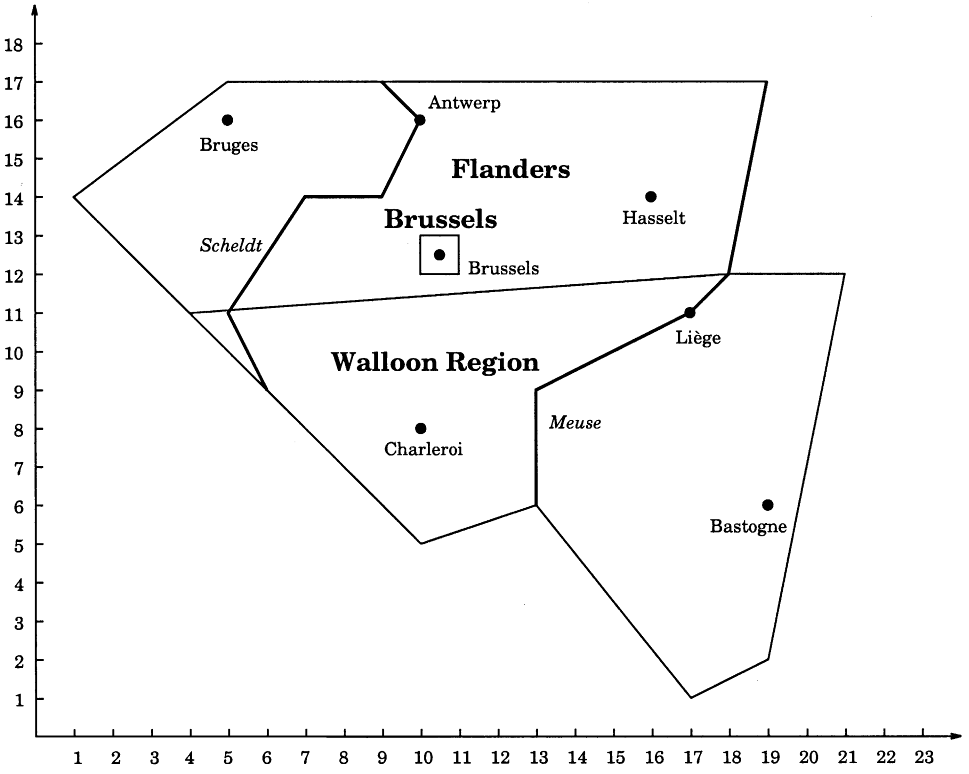
Analogous to polynomial queries, linear queries are defined as mappings between linear databases that are well-defined both at the syntactic and the semantic level.

Notice that all Boolean queries, when restricted to linear inputs, are linear.

A very straightforward linear query language for the linear spatial database model, called FO + linear, is obtained by restricting the real formulae in FO + poly to linear formulae. Using simple algebraic computational techniques for the elimination of variables in sets of linear equations and inequalities [36], it follows that every FO + linear query does indeed return a linear ouput upon a linear input.

[3] Linear formulae can also be defined by requiring the coefficients to be integers or, equivalently, rationals. Linear formula defined in the latter way have stricly less defining power, although the difference is not of practical importance. (This issue was discussed at length by Dumortier and the present authors [13].) Here, we do not elaborate on this issue and make the choice which is most suitable for our purposes.

[4] A *polytope* in a Euclidean space is defined as the convex closure of a non-empty set of points in that space. The *convex closure* of a set of points of a Euclidean space is its smallest convex superset. An *open polytope* is the topological interior of a polytope with respect to the smallest subspace 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	$(y \leqslant 13) \wedge (x \leqslant 11) \wedge (y \geqslant 12) \wedge (x \geqslant 10)$
Flanders	$(y \leqslant 17) \wedge (5x - y \leqslant 78) \wedge (x - 14y \leqslant -150) \wedge (x + y \geqslant 45)$ $\wedge (3x - 4y \geqslant -53) \wedge (\neg((y \leqslant 13) \wedge (x \leqslant 11) \wedge (y \geqslant 12) \wedge (x \geqslant 10)))$
Walloon Region	$((x - 14y \geqslant -150) \wedge (y \leqslant 12) \wedge (19x + 7y \leqslant 375) \wedge (x - 2y \leqslant 15)$ $\wedge (5x + 4y \geqslant 89) \wedge (x \geqslant 13)) \vee ((-x + 3y \geqslant 5) \wedge (x + y \geqslant 45)$ $\wedge (x - 14y \geqslant -150) \wedge (x \geqslant 13))$

Cities

Name	*Geometry*
Antwerp	$(x = 10) \wedge (y = 16)$
Bastogne	$(x = 19) \wedge (y = 6)$
Bruges	$(x = 5) \wedge (y = 16)$
Brussels	$(2x = 20) \wedge (2y = 25)$
Charleroi	$(x = 10) \wedge (y = 8)$
Hasselt	$(x = 16) \wedge (y = 14)$
Liège	$(x = 17) \wedge (y = 11)$

Rivers

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

Fig. 1. Example of a (linear) spatial database.

Example 2.4. An example of a (very simple) linear 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 of type $[1,2;1,2;1,2] \rightarrow [2,0]$ can be expressed as

$$\{(c,r) \mid (\exists x)(\exists y)(\exists x_B)(\exists y_B)(\text{Cities}(c,x,y) \wedge \text{Rivers}(r,x,y)) \\ \wedge \text{Cities}(\mathit{Brussels}, x_B, y_B) \wedge x \geqslant x_B\}$$

in FO + linear.

3. Expressiveness of FO+linear

In this section, we present a list of fundamental queries of topological or geometric nature expressible in FO + linear. To simplify the presentation, we only consider purely spatial queries. In the queries written out explicitly, below, S is assumed to be a relation name in the input database, which can be interpreted as representing of semi-linear set of a Euclidean space $\mathbb{R}^n$.

To start, we observe that set operations such as union, intersection, difference, complement, and projection can be expressed straightforwardly in FO + linear.

Several basic queries of topological nature can also be expressed in FO + linear:

Proposition 3.1. *Let S be a semi-linear set of $\mathbb{R}^n$. The linear queries respectively returning the topological interior, closure, and boundary, can be expressed in* FO + linear.

Proof. The FO + linear formula

$$(\exists \varepsilon)(\varepsilon > \mathbf{0} \wedge (\forall \boldsymbol{y})(-\varepsilon, < \boldsymbol{y} - \boldsymbol{x} < \varepsilon \Rightarrow S(\boldsymbol{y})))$$

expresses the query returning the topological interior of S. Similarly, the FO + linear formula

$$(\forall \varepsilon)(\varepsilon > \mathbf{0} \Rightarrow (\exists \boldsymbol{y})(S(\boldsymbol{y}) \wedge -\varepsilon < \boldsymbol{y} - \boldsymbol{x} < \varepsilon))$$

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

Egenhofer et al. showed in a series of papers [14, 15, 18] 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 geometric objects. Hence all these relationships are also expressible in FO + linear.

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 [24, 34].

Finiteness and boundedness are also decidable in FO + linear.

Proposition 3.2. *Let S be a semi-linear set of $\mathbb{R}^n$. The Boolean query deciding whether S is finite can be expressed in* FO + linear.

Proof. The FO + linear formula

$$(\forall \boldsymbol{x})(S(\boldsymbol{x}) \Rightarrow (\exists \boldsymbol{\varepsilon})(\boldsymbol{\varepsilon} > \mathbf{0} \wedge \neg(\exists \boldsymbol{y})(S(\boldsymbol{y}) \wedge \neg(\boldsymbol{y} = \boldsymbol{x}) \wedge -\boldsymbol{\varepsilon} < \boldsymbol{y} - \boldsymbol{x} < \boldsymbol{\varepsilon})))$$

expresses the query deciding whether S consists of *isolated points only*. For semi-algebraic sets, whence a fortiori for semi-linear sets, this is equivalent to S being finite [6]. □

Observe that, if in the FO + linear formula in the proof of Proposition 3.2 the first two quantifiers are swapped, the resulting FO + linear formula expresses the query deciding whether S is *discrete* in the sense that S has no *adherence points*. For semi-algebraic sets, whence for semi-linear sets, this property is also equivalent to S being finite, however.

Proposition 3.3. *Let S be a semi-linear set of $\mathbb{R}^n$. The Boolean query deciding whether S is bounded can be expressed in* FO + linear.

Proof. The FO + linear formula

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

expresses the query deciding whether S is *bounded.* □

An important property of geometric objects which plays a key role in many spatial database applications is *dimension*. For instance, Clementini et al. [12] use dimension to further refine the class of topological relationships defined by Egenhofer et al., discussed above.

Definition 3.4. The *dimension* of a non-empty semi-linear set S of $\mathbb{R}^n$ is the maximum value of d for which there exists a d-dimensional open cube fully contained in S. The dimension of the empty set equals -1.

We show that it can be decided in FO + linear whether a given semi-linear set has a given number as its dimension.

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

Theorem 3.5. *The predicate* $\dim_n(S,d)$, *in which* S *is a semi-linear set of* $\mathbb{R}^n$ *and* d *a number, and which evaluates to* **true** *if the dimension of* S *equals* d, *can be defined in* FO + linear.

Given a Euclidean space $\mathbb{R}^n$, there are only finitely many values to consider for the dimension of a semi-linear set in that space, namely $-1,0,\ldots,n$. Hence, the dimension of a semi-linear set in a given space can actually be *computed* in FO + linear.

The correctness of Theorem 3.5 follows from five lemmas we present next.

Notation 3.6. Let S be a semi-linear set of $\mathbb{R}^n$, $n\geqslant 1$. Then $\pi_i(S)$ denotes 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 $\mathbb{R}^{n-1}$, i.e., the orthogonal projection of S onto the ith coordinate hyperplane of $\mathbb{R}^n$.

Obviously, the following is true:

Lemma 3.7. *Let* S *be a non-empty* d*-dimensional semi-linear set of* $\mathbb{R}^n$, $n\geqslant 1$. *Then, for* $1\leqslant i\leqslant n$, $\pi_i(S)$ *is a non-empty semi-linear set of* $\mathbb{R}^{n-1}$ *of dimension at most* $\max(d,n-1)$.

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

Lemma 3.8. *Let* S *be a non-empty* d*-dimensional semi-linear set of* $\mathbb{R}^n$, $n\geqslant 1$. *If* $d<n$, *then there exists* i, $1\leqslant i\leqslant n$, *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 $\boldsymbol{p},\boldsymbol{r}_1,\ldots,\boldsymbol{r}_d$ be points in C such that the vectors[6] $\boldsymbol{pr}_1,\ldots,\boldsymbol{pr}_d$ are linearly independent. Let $\boldsymbol{e}_1,\ldots,\boldsymbol{e}_n$ be the unit coordinate vectors of $\mathbb{R}^n$. Since $d<n$, there exists i, $1\leqslant i\leqslant n$, such that $\boldsymbol{e}_i$ is *not* a linear combination of $\boldsymbol{pr}_1,\ldots,\boldsymbol{pr}_d$.

Clearly, $\pi_i(C)$ is convex and open within $\mathbb{R}^{n-1}$, because C is convex and open within $\mathbb{R}^n$. Let $\boldsymbol{q},\boldsymbol{s}_1,\ldots,\boldsymbol{s}_d$ be the orthogonal projections on the ith coordinate hyperplane of $\boldsymbol{p},\boldsymbol{r}_1,\ldots,\boldsymbol{r}_d$, respectively.

We next show that $\boldsymbol{qs}_1,\ldots,\boldsymbol{qs}_d$ are linearly independent. Let $\lambda_1,\ldots,\lambda_d$ be real numbers for which $\lambda_1\boldsymbol{qs}_1+\cdots+\lambda_d\boldsymbol{qs}_d=\boldsymbol{0}$. Let $\boldsymbol{u}$ be the unique point of $\mathbb{R}^n$ for which $\boldsymbol{pu}=\lambda_1\boldsymbol{pr}_1+\cdots+\lambda_d\boldsymbol{pr}_d$. By the linearity of projection, $\pi_i(\boldsymbol{pu})=\boldsymbol{0}$, whence $\boldsymbol{pu}$ is a multiple of $\boldsymbol{e}_i$. By choice of i, this multiple cannot be non-zero. Hence $\boldsymbol{pu}=\boldsymbol{0}$. From the linear independence of $\boldsymbol{pr}_1,\ldots,\boldsymbol{pr}_d$, it then follows that $\lambda_1=\cdots=\lambda_d=0$. Thus, $\boldsymbol{qs}_1,\ldots,\boldsymbol{qs}_d$ are linearly independent.

Clearly, $\pi_i(C)$, an open convex subset of $\mathbb{R}^{n-1}$ containing $d+1$ points, $\boldsymbol{q},\boldsymbol{s}_1,\ldots,\boldsymbol{s}_d$, such that $\boldsymbol{qs}_1,\ldots,\boldsymbol{qs}_d$ are linearly independent, contains a d-dimensional open cube.

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

Since $\pi_i(S)$, which contains $\pi_i(C)$, 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), *in which S is a semi-linear set of* $\mathbb{R}^n$, *and which evaluates to* **true** *if S is the empty relation, can be defined in* FO + linear.

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

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

Proof. The FO + linear formula $(\exists \boldsymbol{x})(\exists \boldsymbol{\varepsilon})(\boldsymbol{\varepsilon} > \mathbf{0} \wedge (\forall \boldsymbol{y})(-\boldsymbol{\varepsilon} < \boldsymbol{y} - \boldsymbol{x} < \boldsymbol{\varepsilon} \Rightarrow S(\boldsymbol{y})))$ expresses that S contains an open cube of maximal dimension. □

Lemma 3.11. *The predicate* $\max(d, d_1, \ldots, d_n)$, $n \geqslant 1$, *which evaluates to* **true** *if d is the maximum value of $d_1, \ldots, d_n$, can be defined in* FO + linear.

Proof. The FO + linear formula $(d = d_1 \vee \cdots \vee d = d_n) \wedge d \geqslant d_1 \wedge \cdots \wedge d \geqslant d_n$ expresses that d is the maximum value of $d_1, \ldots, d_n$. □

We are now ready to give the proof of Theorem 3.5.

Proof of Theorem 3.5. 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

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

clearly expresses $\dim_0(S, d)$ in $\mathbb{R}^0$, the zero-dimensional Euclidean space, which consists of a single point.

Assume now that, for $0 \leqslant k < n$, an FO + linear formula expressing $\dim_k(S, d)$ in $\mathbb{R}^k$ has been obtained. Then, by the induction hypothesis and Lemmas 3.7–3.11, the FO + linear formula

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

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

In many real-world situations, semi-linear sets contain parts of which the “local” dimension is strictly lower than the overall dimension.

Definition 3.12. Let S be a semi-linear set of $\mathbb{R}^n$, and let $0 \leqslant k \leqslant n$. If $\boldsymbol{p}$ is a point of S, then S has *local dimension k* at $\boldsymbol{p}$ if k is the smallest number such that, for each neighborhood V of $\boldsymbol{p}$ in $\mathbb{R}^n$, $S \cap V$ has dimension at least k. The *k-dimensional component* of S is the set of all points of S in which S has local dimension k.

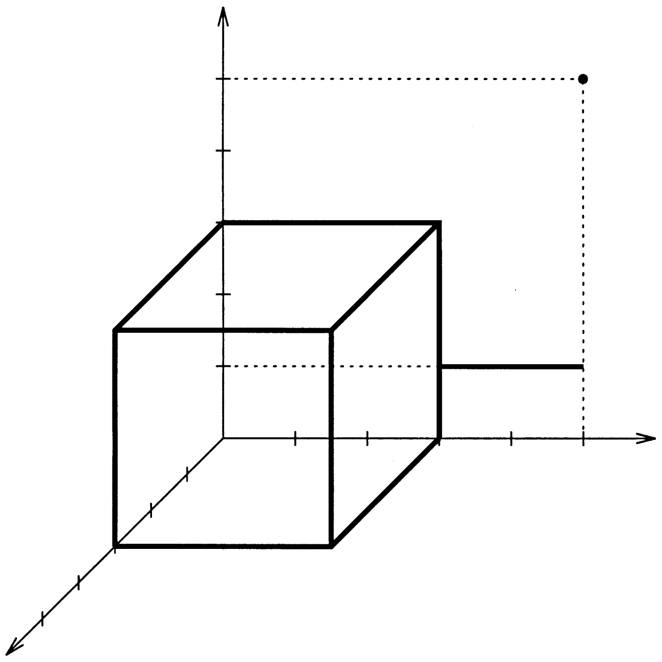


Fig. 2. A semi-linear set with non-empty zero-, one-, and three-dimensional components.

Example 3.13. Consider the three-dimensional semi-linear set displayed in Fig. 2, consisting of a closed, filled cube, to which a closed line segment is attached, and an isolated point. Its three-dimensional component is the closed, filled cube; the two-dimensional component is empty; its one-dimensional component is the attached line segment, open at the point $(3,1,0)$ and closed at the point $(5,1,0)$; and its 0-dimensional component is the singleton set consisting of the point $(5,5,0)$.

Theorem 3.5 now immediately yields the following corollary:

Corollary 3.14. *Let S be a semi-linear set of $\mathbb{R}^n$, let $\boldsymbol{p}$ be a point of $\mathbb{R}^n$, and let $0 \leqslant k \leqslant n$.*

1. *The predicate $\dim_n(S,\boldsymbol{p},k)$, which evaluates to* **true** *if $\boldsymbol{p}$ is in S and the local dimension of S at $\boldsymbol{p}$ equals k, can be defined in* FO + linear.
2. *The query returning the k-dimensional component of S can be expressed in* FO + linear.

From Corollary 3.14, it further follows that the k-dimensional component of a semi-linear set is in turn semi-linear.

In many of the ensuing problems, the expressibility of the dimension and local-dimension predicates will be used to construct an FO + linear formula computing the corresponding linear query.

Clearly, the non-empty k-dimensional components of a semi-linear set S of $\mathbb{R}^n$, $0 \leqslant k \leqslant n$, form a partition of S. In other work [13], Dumortier and the present authors proposed a slightly different decomposition of a semi-linear set in layers of different dimensions, based on the notion of *regular point*. Using this decomposition method, it is possible to compute in FO + linear the corner points and the wire-frame of a polygon in three-dimensional space, both important queries in 3D computer graphics.

We now turn to queries of a geometric nature.

First, we observe that any affine transformation (which includes translations, scalings, rotations, reflections, and compositions thereof) of semi-linear sets can be expressed in FO + linear.

We next show that the Boolean query deciding whether a semi-linear set is convex can be defined in FO + linear.

Theorem 3.15. *The predicate* convex(S), *in which S is a semi-linear set of* $\mathbb{R}^n$, *and which evaluates to* **true** *if S is convex*, *can be defined in* FO + linear.

Proof. We prove that the FO + linear formula

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

defines the predicate convex(S).

In words, the above formula states that, for each pair of points in S, the mid-point is also in S. Clearly, if S is convex, then S satisfies the above FO + linear formula. Thus suppose, conversely, that S satisfies the above FO + linear formula. To prove that S is convex, we must show that, for arbitrary points $\boldsymbol{p}$ and $\boldsymbol{q}$ in S, the (open) line segment I connecting $\boldsymbol{p}$ and $\boldsymbol{q}$ is contained in S. Because of the property expressed by the above FO + linear formula, all points $k\boldsymbol{p} + (1-k)\boldsymbol{q}$, with k a dyadic number[7] between 0 and 1, are in S. Hence, $S \cap I$ is dense in I. Consequently, $I - S$, which is also a semi-linear whence a semi-algebraic set, is zero-dimensional, and, therefore, finite [6]. A point in $I - S$, however, would necessarily be the mid-point of (infinitely many) pairs of points of $S \cap I$, a contradiction. Thus, $I - S$ is empty, or I is contained in S. □

Thus, we have shown that a semi-linear set is convex if it is closed under taking mid-points. A property of the same kind as the latter one is *point symmetry*. Clearly, the property that a semi-linear set S of $\mathbb{R}^n$ is symmetric with respect to the point $\boldsymbol{p}$ of $\mathbb{R}^n$ can also be expressed in FO + linear, more specifically by the sentence

$$(\forall \boldsymbol{x})(S(\boldsymbol{x}) \Rightarrow (\exists \boldsymbol{y})(\boldsymbol{y} = 2\boldsymbol{p} - \boldsymbol{x} \wedge S(\boldsymbol{y}))).$$

The expressibility of point symmetry in FO + linear can be used to prove the following result.

[7] A dyadic number is a finite sum of (positive and negative) integer powers of 2.

Proposition 3.16. *Let S be a semi-linear set of $\mathbb{R}^n$. The Boolean query deciding whether S is an affine subspace*[8] *of $\mathbb{R}^n$ can be expressed in* FO + linear.

Proof. We show that S is an affine subspace of $\mathbb{R}^n$ if and only if S is symmetric with respect to each of its points; since point symmetry is expressible in FO + linear, the result then immediately follows.

Clearly, each affine subspace of $\mathbb{R}^n$ is symmetric with respect to each of its points.

Thus suppose, conversely, that S is symmetric with respect to each of its points. If S consists of a single point, then, clearly, S is an affine subspace of $\mathbb{R}^n$. Otherwise, let $\boldsymbol{p}$ and $\boldsymbol{q}$ be arbitrary but different points of S, and let L be the line through $\boldsymbol{p}$ and $\boldsymbol{q}$. To show that S is an affine subspace of $\mathbb{R}^n$, we must prove that L is fully contained in S.

Consider $S \cap L$, and assume that $S \cap L \neq L$. Since a semi-algebraic set, whence a fortiori a semi-linear set, can only contain a finite number of connected components [6], $S \cap L$ must be the disjoint union of a finite number of isolated points, non-degenerated intervals, and half-lines. Since S (whence $S \cap L$) is symmetric with respect to each of its points, non-degenerated intervals and half-lines cannot occur in this disjoint union, whence $S \cap L$ is a finite set of (at least two) isolated points. Clearly, $S \cap L$ is *not* point-symmetric with respect to any of its two "outermost" points (which, more formally, are the end points of the interval obtained by taking the convex closure of $S \cap L$), a contradiction with our initial assumption. Thus, $S \cap L = L$, whence L is fully contained in S. □

By combining Proposition 3.16 with the expressibility of the dimension predicate (Theorem 3.5), we immediately obtain that the Boolean query deciding whether a semi-linear set is a k-dimensional affine subspace, $0 \leqslant k \leqslant n$, is expressible in FO + linear.

From the proof of Proposition 3.16, we conclude that requiring a semi-linear set to be globally symmetric with respect to each of its points is a very strong condition. It is, however, also possible to require a semi-linear set to be *locally* symmetric (i.e., within some neighborhood) with respect to each of its points, which leads to the next result.

Proposition 3.17. *Let S be a semi-linear set of $\mathbb{R}^n$. The Boolean query deciding whether S is a finite union of affine subspaces of $\mathbb{R}^n$ can be expressed in* FO + linear.

Proof. We show that S is a finite union of affine subspaces of $\mathbb{R}^n$ if and only if S is topologically closed (Proposition 3.1) and locally symmetric with respect to each of its

[8] An *affine subspace* of $\mathbb{R}^n$ is a translation of a linear subspace of $\mathbb{R}^n$. A *linear subspace* of $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ that is closed under vector addition and scalings. For example, the linear subspaces of $\mathbb{R}^3$ are the origin $\mathbf{0}$ of the canonical coordinate system, all lines of $\mathbb{R}^3$ through $\mathbf{0}$, all planes of $\mathbb{R}^3$ through $\mathbf{0}$, and $\mathbb{R}^3$ itself. Hence, the affine subspaces of $\mathbb{R}^3$ are all points of $\mathbb{R}^3$, all lines of $\mathbb{R}^3$, all planes of $\mathbb{R}^3$, and $\mathbb{R}^3$ itself.

points, i.e., satisfies the FO + linear sentence

$$(\forall \boldsymbol{p})(S(\boldsymbol{p}) \Rightarrow (\exists \boldsymbol{\varepsilon})(\varepsilon > \mathbf{0} \wedge (\forall \boldsymbol{x})(S(\boldsymbol{x}) \\ \wedge -\boldsymbol{\varepsilon} < \boldsymbol{x} - \boldsymbol{p} < \boldsymbol{\varepsilon} \Rightarrow (\exists \boldsymbol{y})(\boldsymbol{y} = 2\boldsymbol{p} - \boldsymbol{x} \wedge S(\boldsymbol{y}))))).$$

From this, the result then immediately follows.

Clearly, each finite union of affine subspaces of $\mathbb{R}^n$ is topologically closed and locally symmetric with respect to each of its points.

Thus suppose, conversely, that S is topologically closed and locally symmetric with respect to each of its points. From results by Dumortier and the present authors [13], it follows that S is a finite disjoint union of semi-linear sets, called the *regular strata*, each of which is topologically open within its affine support. [9] Clearly, the set S is a finite union of affine subspaces if, for each regular stratum of S, the affine support of that stratum is fully contained in S.

For the zero-dimensional regular strata of S, the above property is trivially satisfied. To see that the property also holds for higher-dimensional strata, let $\boldsymbol{p}$ and $\boldsymbol{q}$ be arbitrary but different points in such a regular stratum, and let L be the line through $\boldsymbol{p}$ and $\boldsymbol{q}$. We prove that L is fully contained in S.

Consider $S \cap L$, and assume that $S \cap L \neq L$. Since S and L are both topologically closed, so is $S \cap L$. Thus, $S \cap L$ is the disjoint union of a finite number of isolated points, non-degenerated *closed* intervals, and *closed* half-lines. Moreover, $S \cap L$ *must* contain non-degenerated closed intervals and/or closed half-lines, since the stratum under consideration is non-zero-dimensional and open within its affine support. However, S cannot be locally symmetric with respect to the boundary points of these intervals and/or half lines, a contradiction with our initial assumption. Thus, $S \cap L = L$, whence L is fully contained in S.

Now, for any set which is topologically open within its affine support, that affine support is generated by the lines connecting different points of the set. Thus, we have indeed shown that the affine support of each regular stratum of S is fully contained in S. □

Notice that the topological-closedness condition in Proposition 3.17 is essential, as local point symmetry alone is a much weaker property: for instance, each set which is topologically open within its affine support is locally symmetric with respect to each of its points. In the proof, the topological-closedness condition is used to ensure that boundary points with respect to which S would not be locally point-symmetric have to belong to S.

As for Proposition 3.16, combining Proposition 3.17 with the expressibility of the dimension predicate (Theorem 3.5) or the local dimension predicate (Corollary 3.14) yields the expressibility in FO + linear of several other Boolean queries.

We now turn to the issues of parallelism and orthogonality.

[9] The *affine support* of a set of points in a Euclidean space is the smallest affine subspace containing that set.

Proposition 3.18. *Let S be a k-dimensional affine subspace of $\mathbb{R}^n$, $0 \leqslant k \leqslant n$, and let $\boldsymbol{p}$ be a point of $\mathbb{R}^n$. The linear query returning the k-dimensional affine subspace of $\mathbb{R}^n$ through $\boldsymbol{p}$ parallel*[10] *to S can be expressed in* FO + linear.

Proof. The FO + linear formula

$$(\exists \boldsymbol{y_1})(\exists \boldsymbol{y_2})(S(\boldsymbol{y}_1) \wedge S(\boldsymbol{y}_2) \wedge \boldsymbol{x} = \boldsymbol{p} + \boldsymbol{y_1} - \boldsymbol{y_2})$$

computes the query under consideration. □

In order to prove a similar proposition for orthogonality, we need the following technical lemma.

Lemma 3.19. *Let S be a k-dimensional* linear *subspace of $\mathbb{R}^n$, $2 \leqslant k \leqslant n$.*
1. *The linear subspace S is spanned by its intersections with those $(n-1)$-dimensional coordinate hyperplanes*[10] *in which S is not contained.*
2. *The linear subspace S is spanned by the lines that are intersections of S with one of the $(n-k+1)$-dimensional coordinate subspaces.*[11]

Proof.
1. The proof of the first statement of Lemma 3.19 goes by induction on n. If $n = 2$, then $k = 2$, and S equals the entire plane. The coordinate hyperplanes are the coordinate axes, and they equal their intersections with S. Obviously, the coordinate axes span the plane. For higher values of n, there are two cases to consider. If S is contained in one of the coordinate hyperplanes, say H, then the first statement of Lemma 3.19 follows from applying the induction hypothesis to S within H, which is of dimension $n-1$. (Notice that the $(n-2)$-dimensional coordinate hyperplanes of H are obtained by intersecting H with each of the other coordinate hyperplanes of $\mathbb{R}^n$.) If S is contained in none of the coordinate hyperplanes, let $H_1, \dots, H_n$ be the coordinate hyperplanes. Obviously, $S \cap H_1, \dots, S \cap H_n$ are all of dimension $k-1$. Now suppose that $S \cap H_1 = \cdots = S \cap H_n$. Then $\bigcap_{i=1}^{n} S \cap H_i$, which is $(k-1)$-dimensional, would equal $S \cap \bigcap_{i=1}^{n} H_i = \{\mathbf{0}\}$, which is zero-dimensional, a contradiction, since $k \geqslant 2$. Thus, $S \cap H_1, \dots, S \cap H_n$ are not all equal. Consequently, the dimension of the linear subspace of $\mathbb{R}^n$ spanned by $S \cap H_1, \dots, S \cap H_n$ is strictly greater than $k-1$, whence it must be k, whence that linear subspace must equal S.
2. The proof of the second statement of Lemma 3.19 goes by induction on n. If $k = 2$, then the second statement of Lemma 3.19 reduces to the first statement of Lemma 3.19. For higher values of k, we first consider the coordinate hyperplanes with which S has an intersection of dimension $k-1$ (these intersections span S,

[10] Two affine subspaces of $\mathbb{R}^n$ are *parallel* if there is a translation mapping one to a subset of the other.

[11] Let $\boldsymbol{e}_1, \dots, \boldsymbol{e}_n$ be the canonical basis of $\mathbb{R}^n$ (i.e., the vector e_i, $1 \leqslant i \leqslant n$, has ith coordinate 1 and all other coordinates 0). A *d-dimensional coordinate subspace* of $\mathbb{R}^n$, $0 \leqslant d \leqslant n$, is any linear subspace of $\mathbb{R}^n$ generated by exactly d of the canonical basis vectors $\boldsymbol{e}_1, \dots, \boldsymbol{e}_n$. A *coordinate hyperplane* of $\mathbb{R}^n$ is an $(n-1)$-dimensional coordinate subspace of $\mathbb{R}^n$.

by the first statement of Lemma 3.19), and then apply the induction hypothesis to each of these intersections within their respective $(n-1)$-dimensional hyperplanes, yielding the desired result. □

We are now ready to prove the following result.

Proposition 3.20. *Let S be a k-dimensional affine subspace of $\mathbb{R}^n$, $0 \leqslant k \leqslant n$, and let $\boldsymbol{p}$ be a point of $\mathbb{R}^n$. The linear query returning the $(n-k)$-dimensional affine subspace through $\boldsymbol{p}$ orthogonal*[12] *to S can be expressed in* FO + linear.

Proof. By Proposition 3.18, we may assume without loss of generality that S goes through $\mathbf{0}$, the origin of the canonical coordinate system of $\mathbb{R}^n$. By the same token, it suffices to prove that the query returning the $(n-k)$-dimensional affine subspace through $\mathbf{0}$ orthogonal to S can be expressed in FO + linear. In other words, we have to show that, if S is a k-dimensional *linear* subspace of $\mathbb{R}^n$, the linear query returning the $(n-k)$-dimensional *linear* subspace $S^{\perp}$ of $\mathbb{R}^n$ orthogonal to S can be expressed in FO + linear.

By the expressibility of the dimension predicate (Theorem 3.5), we may divide the problem in cases according to the dimension of S.

If S is zero-dimensional (i.e., equals $\{\mathbf{0}\}$), then $S^{\perp}$ equals $\mathbb{R}^n$, which is the evaluation of $\{\boldsymbol{x} \mid \textbf{true}\}$.

If S is one-dimensional, i.e., a line through the origin $\mathbf{0}$ of the canonical coordinate system, we propose an FO + linear formula of the form

$$\varphi_1(\boldsymbol{x}) \vee \cdots \vee \varphi_n(\boldsymbol{x}).$$

We first explain how $\varphi_1(\boldsymbol{x})$ is obtained.

We start by checking whether S is contained in the first coordinate hyperplane, i.e., whether the first coordinate of all points of S equals 0, which can easily be done in FO + linear. If this is the case, we return the empty set as partial output corresponding to $\varphi_1(\boldsymbol{x})$. Otherwise, we compute the unique point $\boldsymbol{p} = (1, p_2, \ldots, p_n)$ of S with 1st coordinate 1, which, again, can easily be done in FO + linear. The real formula

$$x_1 + p_2x_2 + \cdots + p_nx_n = 0$$

defines the set of all vectors $\boldsymbol{x} = (x_1, \ldots, x_n)$ orthogonal to $\boldsymbol{p}$, which constitute $S^{\perp}$. Unfortunately, the above formula is not linear. We can demonstrate, however, that it is possible to compute the above products in FO + linear. Let, for $j = 2, \ldots, n$, the

[12] Two affine subspaces of $\mathbb{R}^n$ are called *orthogonal* if their corresponding linear subspaces (obtained by translating the affine subspaces to the origin of the coordinate system) are orthogonal. Two linear subspaces of $\mathbb{R}^n$ are orthogonal if each vector $\boldsymbol{x}$ of the first subspace is orthogonal to each vector $\boldsymbol{y}$ of the second subspace, i.e., if $\boldsymbol{x}.\boldsymbol{y} = 0$. If S is a linear subspace of $\mathbb{R}^n$ of dimension k, then the set $S^{\perp}$ of *all* vectors of $\mathbb{R}^n$ orthogonal to all vectors of S is a linear subspace of $\mathbb{R}^n$ of dimension $n-k$.

predicate $S_{1j}(x, y)$ be an abbreviation for the FO + linear formula

$$(\exists \boldsymbol{x})(S(\boldsymbol{x}) \wedge x_1 = x \wedge x_j = y).$$

Hence, S_{1j} represents the projection of S on the $(1, j)$th coordinate plane. Hence, S_{1j} is the line through the origin $(0, 0)$ and the point $(1, p_j)$ in $\mathbb{R}^2$, and, therefore, $S_{1j}(x, y)$ is **true** if and only if $y = p_j x$. The real formula $x_1 + p_2x_2 + \cdots + p_nx_n = 0$ is therefore equivalent to the FO + linear formula

$$(\exists y_2)\ldots(\exists y_n)(S_{12}(x_2, y_2) \wedge \cdots \wedge S_{1n}(x_n, y_n) \wedge x_1 + y_2 + \cdots + y_n = 0).$$

Hence, in the case considered, it is possible to return $S^\perp$ as partial output corresponding to $\varphi_1(\boldsymbol{x})$.

The construction of $\varphi_i(\boldsymbol{x})$, $2 \leqslant i \leqslant n$, is analogous, with the role of the 1st coordinate hyperplane taken over by the ith coordinate hyperplane.

Since S is not parallel to at least one of the coordinate hyperplanes, at least one of the partial outputs corresponding to $\varphi_1(\boldsymbol{x}), \ldots, \varphi_n(\boldsymbol{x})$, respectively, is non-empty, whence the union of all the partial outputs is always $S^\perp$, which concludes the case that S is one-dimensional.

If S is of higher dimension, we first compute the intersections of S with the $(n - k + 1)$-dimensional coordinate subspaces. Let E be such a subspace. If $S \cap E$ is one-dimensional (i.e., a line), which can be checked in FO + linear, then we return $(S \cap E)^\perp$ as partial output, computed as in the previous case. Otherwise, we return $\mathbb{R}^n$ as partial output. By Lemma 3.19, the *intersection* of all these partial outputs yields $S^\perp$. □

In the two-dimensional plane $\mathbb{R}^2$, Proposition 3.20 says that it is possible to compute in FO + linear the line through a given point orthogonal to a given line, i.e., making a right angle with it. It is possible to generalize this result to other angles:

Corollary 3.21. *Let S, T_1 and T_2 be lines in the plane $\mathbb{R}^2$, and let $\boldsymbol{p}$ be a point of $\mathbb{R}^2$. The linear query returning the line through $\boldsymbol{p}$ making the same angle with S as T_2 with T_1 can be expressed in* FO + linear.

Proof. As explained at the beginning of the proof of Proposition 3.20, we may assume without loss of generality that all lines concerned go through $\mathbf{0}$, the origin of the canonical coordinate system, and that $\boldsymbol{p} = \mathbf{0}$. The geometric construction of the required line shown in Fig. 3 demonstrates that Corollary 3.21 is indeed an immediate consequence of Proposition 3.20. □

It can be seen that Corollary 3.21 still holds if S, T_1, T_2, and $\boldsymbol{p}$ are contained in a plane of a space $\mathbb{R}^n$ of arbitrary dimension. Indeed, if T_1 and T_2 are parallel (i.e., are equal or have no point in common), then the output of the query in Corollary 3.21 is the line through $\boldsymbol{p}$ parallel to S, which can be computed in FO + linear by Proposition 3.18. If T_1 and T_2 are not parallel, then $\boldsymbol{q}$ is in the plane supported by T_1 and T_2 if and only if $\boldsymbol{q}$ is the mid-point of two points in $T_1 \cup T_2$. Hence, this plane can be computed in

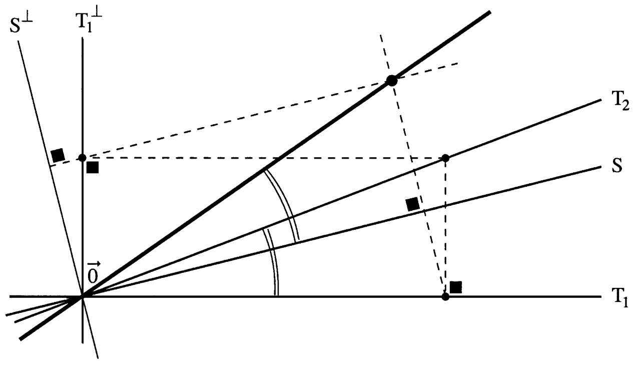


Fig. 3. Geometric construction of the line through **0** making the same angle with S as T_2 with T_1. First, the thin lines $S^{\perp}$ and $T_1{}^{\perp}$, orthogonal to S and T_1, respectively, are drawn. The construction starts with an arbitrary point of T_2 and yields a point of the result. Each point of the result can be reached in this way.

FO + linear. Using this additional information and Proposition 3.20, it is not difficult anymore to see that the construction in the proof of Corollary 3.21 can still be carried out in FO + linear.

To close this section, we restrict ourselves to one-dimensional semi-linear sets, and ascertain the expressibility in FO + linear of several linear queries pertaining to such sets.

We first summarize what we can determine in FO + linear about the composition of such sets.

Proposition 3.22. *Let S be a semi-linear set of $\mathbb{R}^n$. The Boolean queries deciding the following properties can be expressed in* FO + linear:

1. *S consists of a finite number of lines, half-lines, and (non-degenerated) line segments only;*
2. *S consists of a finite number of lines only;*
3. *S is a single line;*
4. *S is a single (open/closed) half-line;*
5. *S is a single (non-degenerated) (open/half-open-half-closed/closed) line segment.*

Proof. For the first query, it suffices to check that S has dimension 1 (Theorem 3.5). Degenerated line segments (i.e., isolated points) can easily be checked for (Proposition 3.2); alternatively, one might check that S equals its one-dimensional component.

For the second query, it suffices to check that, in addition to being one-dimensional, S is a finite union of affine subspaces of $\mathbb{R}^n$ (Proposition 3.17).

For the third query, it suffices to check that, in addition to being one-dimensional, S is an affine subspace of $\mathbb{R}^n$ (Proposition 3.16).

For the fourth and fifth queries, we first check that S consists of lines, half-lines, and (non-degenerated) line segments only (Query 1). If, in addition, S is convex (Theorem 3.15), then S is either a single line, or a single half-line, or a single (non-degenerated) line segment. Thus, if S is not bounded (Proposition 3.3) and S is not a line (Query 3), S is a half-line; if S is bounded, S is a (non-degenerated) interval. The interval is non-degenerated if S is finite (Proposition 3.2). To decide which type of half-line or non-degenerated interval S is, we can in addition compute the set of points of S with respect to which S is *not* locally symmetric. These points are precisely the boundary points of S contained in S. The type of half-line or interval then depends on whether there are 0, 1, or 2 of them. □

We now focus on the case where S consists of a *finite number of lines* of $\mathbb{R}^n$.

Proposition 3.23. *Let S consist of a finite number of lines of $\mathbb{R}^n$, $n \geqslant 1$. The query returning the set of all intersection points between two or more lines in S can be expressed in* FO + linear.

Proof. The intersection points of two or more lines in S are precisely those points $\boldsymbol{p}$ of S in which S is not *locally convex*, i.e., those points $\boldsymbol{p}$ of S such that for no convex neighborhood V of $\boldsymbol{p}$ in $\mathbb{R}^n$, $S \cap V$ is convex. Since the neighborhoods used in the topological queries earlier in this section are convex and suffice to express the above property, Proposition 3.23 now follows from Theorem 3.15. □

Proposition 3.24. *Let S be a semi-linear set of $\mathbb{R}^n$. The Boolean query deciding whether S consists of a finite number of parallel lines can be expressed in* FO + linear.

Proof. By Proposition 3.22 and 3.23, we can decide whether S consists of a finite number of lines *without* intersection points. If this is the case, then every point of S is on a unique line in S.

Now, consider an arbitrary point $\boldsymbol{x}$ of S. Consider a neighborhood V of $\boldsymbol{x}$ in $\mathbb{R}^n$ such that $S \cap V$ is convex. (Such a neighborhood exists, and its existence can be asserted in FO + linear using the techniques employed to express topological queries and Theorem 3.15.) Necessarily, $S \cap V$ is a line segment. Let $\boldsymbol{y}$ be an arbitrary point of $S \cap V$ different from $\boldsymbol{x}$. Let $\boldsymbol{z}$ be an arbitrary point of S. If S consists of parallel lines only, then $\boldsymbol{z} + \boldsymbol{y} - \boldsymbol{x}$ is on the same line as $\boldsymbol{z}$, whence also in S. Conversely, if S does *not* consist of parallel lines only, then the above property fails if $\boldsymbol{x}$ and $\boldsymbol{z}$ are chosen on two non-parallel lines. □

For the following result, we need a technical lemma.

Lemma 3.25. *Let S consist of a finite number of parallel lines of $\mathbb{R}^n$. There exists a linear query expressible in* FO + linear *that selects a single line from these.*

Proof. First, assume that S is *not* parallel to the ith coordinate hyperplane (i.e., the set of ith coordinates of S is *not* a singleton), $1 \leqslant i \leqslant n$, and let, for $j = 1, \ldots, n$, $j \neq i$,

$\min_{ij}(S)$ be the semi-linear set defined by the FO + linear formula

$$S(\boldsymbol{x}) \wedge (\forall \boldsymbol{y})(S(\boldsymbol{y}) \wedge y_i = x_i \Rightarrow x_j \leqslant y_j).$$

Then $\min_{ij}(S)$ consists of those lines of S that are "leftmost" with respect to the jth coordinate axis. Thus,

$$\min_i(S) = \min_{in}\left(\min_{i(n-1)}\left(\ldots \min_{i1}(S)\cdots\right)\right)$$

consists of single line.

We next modify the definition of $\min_i$ such that $\min_i(S) = S$ if S is parallel to the ith coordinate plane, a condition which can easily be checked in FO + linear.

Then, in all cases,

$$\min(S) = \min_n\left(\min_{n-1}\left(\ldots \min_1(S)\cdots\right)\right)$$

consists of a single line, since S is not parallel to at least one of the coordinate hyperplanes. □

Proposition 3.26. *Let S consist of a finite number of lines of* $\mathbb{R}^n$, *and let* $\boldsymbol{p}$ *be a point of S. The linear query returning the semi-linear set consisting of all lines of S through* $\boldsymbol{p}$ *can be expressed in* FO + linear.

Proof. First, assume that $\boldsymbol{p}$ is not an intersection point of two or more lines of S (cf. Proposition 3.23). Then, precisely one line of S goes through $\boldsymbol{p}$. Consider a neighborhood V of $\boldsymbol{p}$ in $\mathbb{R}^n$ such that $S \cap V$ is convex. (Such a neighborhood exists, and its existence can be asserted in FO + linear using the techniques employed to express topological queries and Theorem 3.15.) Necessarily, $S \cap V$ is a line segment. Let $S_{\boldsymbol{p}}$ be the set of all points $\boldsymbol{x}$ of S which have a neighborhood W in $\mathbb{R}^n$ for which $\tau_{\boldsymbol{xp}}(S \cap W) \subseteq S \cap V$, where $\tau_{\boldsymbol{xp}}$ is the translation over vector $\boldsymbol{xp} = \boldsymbol{p} - \boldsymbol{x}$. Clearly, there exists an FO + linear formula defining $S_{\boldsymbol{p}}$. Since translations preserve parallelism, $S_{\boldsymbol{p}}$ consists of the lines of S parallel to the line of S through $\boldsymbol{p}$, from which the intersection points with other lines have been omitted. Hence, the topological closure of $S_{\boldsymbol{p}}$, $\bar{S}_{\boldsymbol{p}}$, which can be computed in FO + linear (Proposition 3.1), consists of all lines of S parallel to the line of S through $\boldsymbol{p}$. From this set, a single line can be selected in FO + linear, by Lemma 3.25. The output of the query is the line through $\boldsymbol{p}$ parallel to this single line, which can be computed in FO + linear, by Proposition 3.18.

Next, assume that $\boldsymbol{p}$ is an intersection point of two or more lines of S. Consider a neighborhood V of $\boldsymbol{p}$ in $\mathbb{R}^n$ such that $S \cap V$ is point-symmetric with respect to $\boldsymbol{p}$. (Such a neighborhood exists, and its existence can be asserted in FO + linear, as was shown earlier in the proof of Proposition 3.17.) For every point $\boldsymbol{q}$ in $S \cap V - \{\boldsymbol{p}\}$, $\boldsymbol{q}$ is *not* an intersection point of lines of S, whence the semi-linear set consisting of all lines of S through $\boldsymbol{q}$ can be computed in FO + linear, as shown in the first part of the proof. The output of the query is the union of all these sets. □

Proposition 3.26 has some remarkable corollaries.

Corollary 3.27. *Let S consist of a finite number of lines of $\mathbb{R}^n$, and let $\boldsymbol{p}$ and $\boldsymbol{q}$ be points on some line of S. The linear queries respectively returning the line through $\boldsymbol{p}$ and $\boldsymbol{q}$ and the closed line segment between $\boldsymbol{p}$ and $\boldsymbol{q}$ can be expressed in* FO + linear.

Proof. The expressibility in FO + linear of the first query is an immediate consequence of Proposition 3.26. To see that the second query is expressible in FO + linear, let $\boldsymbol{r}$ be the mid-point of $\boldsymbol{p}$ and $\boldsymbol{q}$. The points common to the line through $\boldsymbol{p}$ and $\boldsymbol{q}$ and all convex neighborhoods of $\boldsymbol{r}$ containing both $\boldsymbol{p}$ and $\boldsymbol{q}$ constitute the closed line segment between $\boldsymbol{p}$ and $\boldsymbol{q}$. □

Corollary 3.28. *Let S consist of a finite number of lines of $\mathbb{R}^n$. The query returning all pairs of parallel lines of S (which can be seen as a query of type $[0,n] \rightarrow [0,2n]$) can be expressed in* FO + linear.

Proof. Let $\boldsymbol{p}$ and $\boldsymbol{q}$ be points of S which are not intersection points of two or more lines of S. (This can be decided in FO + linear, by Proposition 3.23). By Proposition 3.26, it is possible to compute in FO + linear the unique lines of S going through $\boldsymbol{p}$ and $\boldsymbol{q}$, respectively. This pair of lines is retained in the output of the query if they are parallel (i.e., are equal or have no point in common). □

The above results give some idea of the expressive power of FO + linear. 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. [2] 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 (Afrati et al. [2]). *The Boolean query on semi-linear sets S of $\mathbb{R}$ 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 $\mathrm{FO} + \mathrm{poly}^{\mathrm{lin}}$.

Nevertheless, Proposition 4.1, by the somewhat artificial character of the query exhibited, does not provide us insight into the adequacy of FO + linear as a linear query language. Since the $\mathrm{FO} + \mathrm{poly}^{\mathrm{lin}}$ queries play an important role in practical spatial databases, the most straightforward way to obtain insight in the adequacy of FO + linear as a linear query language is to discover an algorithm to decide whether an FO + poly formula defining an $\mathrm{FO} + \mathrm{poly}^{\mathrm{lin}}$ query is expressible in FO + linear.

However, we must point out that we can prove the following theorem, which can be viewed as an analog of Rice's Theorem for FO + poly-expressible queries:

Theorem 4.2. *Let $\mathscr{C}_1$ and $\mathscr{C}_2$ be subclasses of spatial database relations such that $\mathscr{C}_1$ contains all semantically finite spatial database relations. Let* $\mathrm{FO} + \mathrm{poly}^{\mathscr{C}_1 \to \mathscr{C}_2}$ *be the sublanguage of* FO + poly *consisting of those formulae that return outputs in $\mathscr{C}_2$ upon inputs in $\mathscr{C}_1$. Let $\mathscr{E}$ be a semantic property of* $\mathrm{FO} + \mathrm{poly}^{\mathscr{C}_1 \to \mathscr{C}_2}$ *formulae satisfying the following conditions*:

1. *If $\varphi(R_1,\ldots,R_k; v_1,\ldots,v_m; x_1,\ldots,x_n)$ is an* $\mathrm{FO} + \mathrm{poly}^{\mathscr{C}_1 \to \mathscr{C}_2}$ *formula satisfying property $\mathscr{E}$ and defining a query of type, say, $[m_1,n_1;\ldots;m_k,n_k] \to [m,n]$, and if $\vartheta(v_1, \ldots, v_{m_1}; x_1,\ldots,x_{n_1})$ is a real formula defining a semantically finite spatial database relation of type $[m_1,n_1]$, then the* $\mathrm{FO} + \mathrm{poly}^{\mathscr{C}_1 \to \mathscr{C}_2}$ *formula*

   $$\varphi'(R_2,\ldots,R_k; v_1,\ldots,v_m; x_1,\ldots,x_n),$$

   obtained from φ by substituting each occurrence of R_1 by ϑ, and defining a query of type $[m_2,n_2;\ldots;m_k,n_k] \to [m,n]$, satisfies property $\mathscr{E}$.
2. *If $\varphi(R_2,\ldots,R_k; v_1,\ldots,v_m; x_1,\ldots,x_n)$ is an* $\mathrm{FO} + \mathrm{poly}^{\mathscr{C}_1 \to \mathscr{C}_2}$ *formula satisfying property $\mathscr{E}$ and defining a query of type, say, $[m_2,n_2;\ldots;m_k,n_k] \to [m,n]$, then, for each relation type $[m_1,n_1]$, and for each relation name R_1 of type $[m_1,n_1]$, the* $\mathrm{FO} + \mathrm{poly}^{\mathscr{C}_1 \to \mathscr{C}_2}$ *formula*

   $$\varphi'(R_1,\ldots,R_k; v_1,\ldots,v_m; x_1,\ldots,x_n) \equiv \varphi(R_2,\ldots,R_k; v_1,\ldots,v_m; x_1,\ldots,x_n)$$

   defining a query of type $[m_1,n_1;\ldots;m_k,n_k] \to [m,n]$ satisfies property $\mathscr{E}$.
3. *For some query type $[m_1,n_1;\ldots;m_k,n_k] \to [m,n]$, there exist* $\mathrm{FO} + \mathrm{poly}^{\mathscr{C}_1 \to \mathscr{C}_2}$ *formulae*

   $$\varphi^+(R_1,\ldots,R_k; v_1,\ldots,v_m; x_1,\ldots,x_n) \quad \textit{and} \quad \varphi^-(R_1,\ldots,R_k; v_1,\ldots,v_m; x_1,\ldots,x_n)$$

   both defining a query of type $[m_1,n_1;\ldots;m_k,n_k] \to [m,n]$ such that φ^+ has property $\mathscr{E}$ and φ^- does not have property $\mathscr{E}$.

Then it is undecidable whether an $\mathrm{FO} + \mathrm{poly}^{\mathscr{C}_1 \to \mathscr{C}_2}$ *formula has property $\mathscr{E}$.*

Proof. The proof is a variation of a proof of Paredaens et al. [41] concerning undecidability of genericity in FO + poly (Theorem 1, p. 285).

The $\forall^*$-fragment of number theory is undecidable since Hilbert's 10th problem can be reduced to it. We encode a natural number n by the finite set $\mathit{enc}(n) := \{0,\ldots,n\}$, and we encode a vector of natural numbers $(n_1,\dots,n_k)$ by $enc(n_1)\cup(enc(n_2)+n_1+2)\cup\cdots\cup(enc(n_k)+n_1+2+\cdots+n_{k-1}+2)$.[13] The corresponding decoding is first-order-expressible. Let $\varphi^+(R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$ and $\varphi^-(R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$ be $\mathrm{FO}+\mathrm{poly}^{\mathscr{C}_1\to\mathscr{C}_2}$ formulae defining queries of some common type, say, $[m_1,n_1;\dots;m_k,n_k]\to[m,n]$, such that φ^+ has property $\mathscr{E}$ and φ^- does not have property $\mathscr{E}$. We then reduce a $\forall^*$-sentence $(\forall\boldsymbol{x})\psi(\boldsymbol{x})$ of number theory to the following query of signature $[0,1;m_1,n_1;\dots;m_k,n_k]\to[m,n]$ (S of type $[0,1]$ and $R_1,\dots,R_k$ of types $[m_1,n_1],\dots,[m_k,n_k]$, respectively, are the input relation names of this query):

```
if $S$ encodes a vector $\boldsymbol{x}$
  then
    if $\psi(\boldsymbol{x})$
      then
        return($\{(v_1,\dots,v_m;x_1,\dots,x_n)\mid\varphi^+(R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)\}$)
      else
        return($\{(v_1,\dots,v_m;x_1,\dots,x_n)\mid\varphi^-(R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)\}$)
  else
    return($\{(v_1,\dots,v_m;x_1,\dots,x_n)\mid\varphi^+(R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)\}$).
```

By definition, the above query is $\mathrm{FO}+\mathrm{poly}^{\mathscr{C}_1\to\mathscr{C}_2}$-expressible. Moreover, an $\mathrm{FO}+\mathrm{poly}^{\mathscr{C}_1\to\mathscr{C}_2}$ formula computing this query can effectively be constructed. Let

$$\varphi(S,R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$$

be such a formula. When the $\forall^*$-sentence $(\forall\boldsymbol{x})\psi(\boldsymbol{x})$ is valid, then $\varphi(S,R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)\equiv\varphi^+(R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$ has property $\mathscr{E}$, by conditions 2 and 3 above. When the $\forall^*$ sentence $(\forall\boldsymbol{x})\psi(\boldsymbol{x})$ is not valid, then $\varphi(S,R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$ does not have property $\mathscr{E}$. To see this, let $\boldsymbol{n}$ be a vector of natural numbers for which $\psi(\boldsymbol{n})$ is **false**, and let $\vartheta(x)$ be a real formula defining $enc(\boldsymbol{n})$. Let

$$\varphi'(R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$$

be the $\mathrm{FO}+\mathrm{poly}^{\mathscr{C}_1\to\mathscr{C}_2}$ formula obtained from $\varphi(S,R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$ by substituting each occurrence of S by ϑ and defining a query of type $[m_1,n_1;\dots;m_k,n_k]\to[m,n]$. Now, if $\varphi(S,R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$ would have property $\mathscr{E}$, then $\varphi'(R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$ would have property $\mathscr{E}$, by condition 1, a contradiction with condition 3, since $\varphi'(R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)\equiv\varphi^-(R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$. Thus, $\varphi(S,R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$ does *not* satisfy property $\mathscr{E}$. In summary, $\varphi(S,R_1,\dots,R_k;v_1,\dots,v_m;x_1,\dots,x_n)$ has property $\mathscr{E}$ if and only if the $\forall^*$-sentence $(\forall\boldsymbol{x})\varphi(\boldsymbol{x})$ is valid. □

[13] For N a set of natural numbers and n a natural number, $N+n$ denotes the set $\{x+n\mid x\in N\}$.

If, in Theorem 4.2, we let $\mathscr{C}_1 = \mathscr{C}_2$ be the class of linear spatial database relations, and $\mathscr{E}$ be FO + linear-expressibility of the corresponding FO + poly$^{\text{lin}}$ query, then we immediately obtain the following corollary:

Corollary 4.3. *It is undecidable whether an* FO + poly$^{\text{lin}}$ *query induced by an* FO + poly *formula can be expressed in* FO + linear.

Hence, an algorithm to decide whether an FO + poly expression defining an FO + poly$^{\text{lin}}$ query is expressible in FO + linear cannot be provided, and we must look for other means to assess the adequacy of FO + linear as linear query language.

In the literature (e.g. [2, 48]), researchers have been concerned with the non-definability of certain geometric sets by linear first-order formulae (i.e., with the non-semi-linearity of these sets). In this paper, however, we are concerned with the non-expressibility of *queries* in FO + linear, as opposed to the non-definability of sets by linear formulae. The principal contribution of the remainder of this section is the development of a general tool to lift the non-definability of certain semi-algebraic sets by linear formulae to the non-expressibility in FO + linear of closely related FO + poly queries. Application of this tool allows us to establish the non-expressibility in FO + linear of a wide range of queries in FO + poly$^{\text{lin}}$, which in turn improves our insight in the nature of the FO + poly$^{\text{lin}}$ queries not expressible in FO + linear.

To develop this tool, we first establish a link between certain semi-algebraic sets and certain FO + poly queries.

Definition 4.4. Let P be a semi-algebraic subset of $(\mathbb{R}^n)^m$, $m, n \geqslant 1$. Let k be such that $0 \leqslant k \leqslant m$. Furthermore, assume that P and k are such that, for each sequences $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ in $\mathbb{R}^n$, and for each sequences $i_1, \ldots, i_k$, $1 \leqslant i_1, \ldots, i_k \leqslant k$, such that $\{\boldsymbol{u}_{i_1}, \ldots, \boldsymbol{u}_{i_k}\} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k\}$, the following permutation invariance property holds, for all $\boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_m$ in $\mathbb{R}^n$:

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

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

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

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

1. Let P_1 be the set

   $$\{(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_m)\in(\mathbb{R}^n)^m \mid \boldsymbol{u}_1,\ldots,\boldsymbol{u}_m \text{ are collinear}\},$$

   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

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

   Moreover, it satisfies Definition 4.4 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 collinear points.
2. Let P_2 be the set

   $$\{(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_m)\in(\mathbb{R}^n)^m \mid \boldsymbol{u}_{m-1} \text{ and } \boldsymbol{u}_m \text{ belong to a common closed Voronoi cell with respect to } \boldsymbol{u}_1,\ldots,\boldsymbol{u}_{m-2}\}.$$

   The points $\boldsymbol{u}_{m-1}$ and $\boldsymbol{u}_m$ belongs to a common closed Voronoi cell with respect to $\boldsymbol{u}_1,\ldots,\boldsymbol{u}_{m-2}$ if the condition

   $$(\exists\boldsymbol{u})((\boldsymbol{u}=\boldsymbol{u}_1 \vee \ldots \vee \boldsymbol{u}=\boldsymbol{u}_{m-2})\wedge \\ \wedge_{i=1}^{m-2}(d(\boldsymbol{u}_{m-1},\boldsymbol{u})\leqslant d(\boldsymbol{u}_{m-1},\boldsymbol{u}_i)\wedge d(\boldsymbol{u}_m,\boldsymbol{u})\leqslant d(\boldsymbol{u}_m,\boldsymbol{u}_i)))$$

   is satisfied, where $d(\boldsymbol{r},\boldsymbol{s})$ denotes the Euclidean distance between $\boldsymbol{r}$ and $\boldsymbol{s}$. Hence, P_2 is semi-algebraic. Moreover, it satisfies Definition 4.4 for $k=m-2$. The associated query $Q_{P_2,m-2}$ of type $[0,n]\to[0,2n]$ can be interpreted as the linear query that associates, with each non-empty semi-linear set S consisting of at most $m-2$ points, pairs of points which belong to a common closed Voronoi cell with respect to the points of S, and, with every other semi-linear set S, the empty set.
3. Let S be a semi-algebraic set of $\mathbb{R}^n$. We define the *diameter* of S, denoted $\varnothing(S)$, as the supremum of the (Euclidean) distances between two points of S. Let

   $$P_3=\{(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_{m-1},(d,\underbrace{0,\ldots,0}_{n-1}))\mid \varnothing(\{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_{m-1}\})=d\}.$$

   It is easily seen that P_3 is a semi-algebraic set; e.g., for $m=3$, it computes the distance between two points. Moreover, it satisfies Definition 4.4 for $k=m-1$. The associated query $Q_{P_3,m-1}$ can be interpreted as the aggregate query of type $[0,n]\to[0,n]$ which associates with each non-empty semi-linear set S consisting of at most $m-1$ points the singleton

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

   and, with every other semi-linear set S, the empty set.

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 a linear formula.

Theorem 4.6. *Let P be a semi-algebraic subset of* $(\mathbb{R}^n)^m$, $m,n \geqslant 1$, *let k be such that* $0 \leqslant k \leqslant m$, *and let P and k satisfy the conditions of Definition* 4.4. *If P is not definable by a linear formula* (*which is decidable* [13]), *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* $\mathbb{R}^n$, $Q(S) = Q_{P,k}(S)$ *if* $Q_{P,k}(S)$ *is not empty, then Q is not expressible in* FO + linear.

Proof.
1. Assume, on the contrary, that the query $Q_{P,k}$ is expressible in FO + linear. Then there exists an FO + linear formula $\varphi_{P,k}(R; \boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_m)$, with R an appropriate predicate name, such that, for each semi-linear set S of $\mathbb{R}^n$, $Q_{P,k}(S) = \{(\boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_m) \mid \varphi_{P,k}(S; \boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_m)\}$. 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 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ be variables that do not occur in $\varphi_{P,k}$. Now replace every literal of the form

   $$R(\boldsymbol{z})$$

   in $\varphi_{P,k}$ by the formula

   $$\boldsymbol{z} = \boldsymbol{x}_1 \vee \cdots \vee \boldsymbol{z} = \boldsymbol{x}_k.$$

   Observe that the formula $\hat{\varphi}_{P,k}$ is a linear formula with free variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$. 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 $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \in (\mathbb{R}^n)^m$. From the definition of $Q_{P,k}$ and $\varphi_{P,k}$, we have

   $$(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \in P \quad \Leftrightarrow \quad (\boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_m) \in Q_{P,k}(\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k\}),$$

   whence

   $$(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \in P \quad \Leftrightarrow \quad \varphi_{P,k}(\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k\}; \boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_m).$$

   It follows from the construction of $\hat{\varphi}_{P,k}$ from $\varphi_{P,k}$ that

   $$(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \in P \quad \Leftrightarrow \quad \hat{\varphi}_{P,k}(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m).$$

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

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

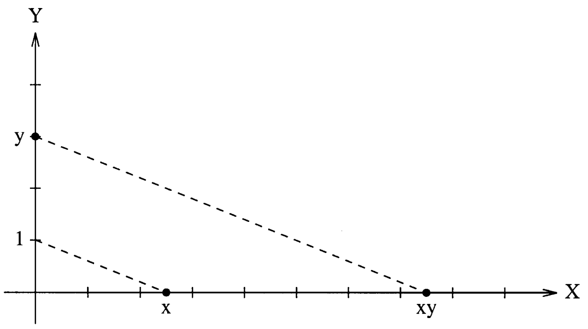


Fig. 4. Geometric construction of the product.

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

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

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 4.6, we first establish that the semi-algebraic sets in Example 4.5 are not definable by linear formulae for most values of m and n.

Proposition 4.7. *The sets P_1 and P_3 are not definable by linear formulae if $n \geqslant 2$ and $m \geqslant 3$. The set P_2 is not definable by a linear formula if $n \geqslant 2$ and $m \geqslant 4$.*

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 $n \geqslant 2$ and some $m \geqslant 3$. Then, clearly, P_1 is also definable by a linear formula for $n = 2$ and $m = 3$. Let $collinear(x_1, x_2, y_1, y_2, z_1, z_2)$ denote this formula. We now show that there exists a linear formula $product(x, y, z)$, 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 Fig. 4, it follows that

$$(x = 0 \wedge z = 0) \vee (y = 0 \wedge z = 0) \vee (y = 1 \wedge z = x)$$
$$\vee \neg(\exists v)(\exists w)(collinear(x, 0, 0, 1, v, w) \wedge collinear(z, 0, 0, y, v, w))$$

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, for $m = 3$, respetively, $m = 4$, we have that

$$(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3) \in P_1 \Leftrightarrow (\exists \boldsymbol{p})(\exists \boldsymbol{q})(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{q}, \boldsymbol{p}_1) \in P_2 \wedge (\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{q}, \boldsymbol{p}_2) \in P_2 \wedge (\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{q}, \boldsymbol{p}_3) \in P_2 \wedge (\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{p}, \boldsymbol{p}_1) \in P_2 \wedge (\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{p}, \boldsymbol{p}_2) \in P_2 \wedge (\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{p}, \boldsymbol{p}_3) \in P_2.$$

3. The semi-algebraic set P_3 is not definable by a linear formula, because, for $m \geqslant 3$, it is possible to obtain a disk by applying appropriate FO + linear-expressible operations to P_3. $\square$

Theorem 4.6 and Proposition 4.7 yield the following corollary, the proof of which is immediate from the former:

Theorem 4.8. (1) *The Boolean query of type* $[0,n] \to [0,0]$ *deciding whether a semi-linear set of* $\mathbb{R}^n$ *is contained in a line is not expressible in* FO + linear.

(2) *The linear query of type* $[0,n] \to [0,2n]$ *returning all pairs of points of* $\mathbb{R}^n$ *that belong to a common closed Voronoi cell with respect to a semi-linear set of* $\mathbb{R}^n$ *if that set is finite and non-empty, and returning the empty set otherwise, is not expressible in* FO + linear.

(3) *The linear aggregate query of type* $[0,n] \to [0,n]$ *returning, upon a semi-linear set* S *of* $\mathbb{R}^n$ *as input, the singleton*

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

if S *is bounded and non-empty, and returning the empty set otherwise, is not expressible in* FO + linear (*whence the corresponding diameter query of type* $[0,n] \to [0,1]$ *is not expressible either*).

Obviously, Theorem 4.6 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.

As a final example, we discuss the non-expressibility in FO + linear of the queries with type $[0,n] \to [0,n]$ which compute the convex closure and the affine support of a semi-linear set. We can show that, for $n \geqslant 2$ and $m \geqslant 3$, the semi-algebraic sets

$$\{(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \in (\mathbb{R}^n)^m \mid \boldsymbol{u}_m \text{ is in the convex closure of } \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{m-1}\}\}$$

and

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

cannot be expressed by linear formulae as the product of two real numbers would become expressible (in the same way as the definability of the set P_1 led to the expressibility of the product of two real numbers). Then, according to Theorem 4.6, we can lift the non-definability of these sets by linear formulae to the non-expressibility in FO + linear of the queries computing the convex closure and the affine support of a semi-linear set. For these last two queries however, the non-expressibility can also be established more directly by reduction to the non-expressibility of the collinearity query.

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 an FO + poly formula induces a linear query. Unfortunately, such an algorithm does not exist, because we can again invoke Theorem 4.2, this time with $\mathscr{C}_1 = \mathscr{C}_2$ the class of all spatial database relations and $\mathscr{E}$ the property of inducing a linear query, yielding

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

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 sublanguage of the relational calculus which precisely expresses the domain-independent relational calculus queries [47]. As a matter of fact, Dumortier and the present authors [13] established the existence of a syntactically definable query language which expresses precisely the FO + poly^{lin} queries. The result is based on the decidability of semi-linearity for semi-algebraic sets which is shown in the same paper. Syntactically, the query language concerned is just FO + poly. Semantically, the standard output of an FO + poly query is replaced by the empty set if it is not a semi-linear set. The language thus obtained is sound (queries return linear outputs on linear inputs) and complete (the standard output of a linear FO + poly query is not modified). Of course, this language is not particularly elegant nor has it practical value, but, at least, its existence justifies the search for more natural sound and complete languages for the FO + poly^{lin} queries. Nevertheless, a top-down approach towards discovering useful linear query languages remains difficult.

In the remainder of 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 linear queries listed in Theorem 4.8 or the collinearity or the convex-closure query.

However, we can*not* achieve our goal by adding the corresponding predicates to FO + linear. Indeed, from the proof of Proposition 4.7, it follows that, e.g., adding a predicate *collinear*$(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, which evaluates to **true** if its arguments are collinear 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 $\text{op}(O)$ of the same signature.[14]

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 $x_1,\dots,x_n$ and m free non-spatial variables $v_1,\dots,v_m$, and if $k \leqslant m$, then

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

  is a *set term* of type $[k,n]$. Observe that, of the non-spatial variables, $v_{k+1},\dots,v_m$ occur *free*, while *all* real variables, $x_1,\dots,x_n$, occur *bounded* in the set term.[15]
- If O is an operator name in $\mathcal{O}$ of type $[m_1,n_1;\dots;m_k,n_k] \to [m,n]$, and $S_1,\dots,S_k$ are set terms of types $[m_1,n_1],\dots,[m_k,n_k]$, respectively, then

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

  is a *set term* of type $[m,n]$ with free variables the free (non-spatial) variables in S_1 through S_k.

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

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

Semantically, when actual values are substituted for the free variables, a set term of type $[m,n]$ represents a subset of $D^m \times \mathbb{R}^n$. Now consider an atomic formula of the form $S(v_1,\dots,v_m;x_1,\dots,x_n)$. When actual values are substituted for the free variables, this atomic formula evaluates to **true** if the evaluation of $(v_1,\dots,v_m;x_1,\dots,x_n)$ 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 easily shown by structural induction:

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

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

[14] To be practically relevant, the set $\mathcal{O}$ must be recursively enumerable.

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

Without going into details, we mention that it is possible to define an algebraic query language equivalent to $\mathrm{FO} + \mathrm{linear} + \mathcal{O}$ by extending the linear algebra [48] with the operators represented by $\mathcal{O}$. This equivalence result forms a theoretical justification for the approach Güting et al. have taken with the development of the ROSE-algebra [25–29], which is extending the relational algebra with a class of spatial operators.

Before we discuss the expressive power of $\mathrm{FO} + \mathrm{linear}$ extended with operators, we give an example of an $\mathrm{FO} + \mathrm{linear} + \mathcal{O}$ query language in which we can express the *collinearity* and *convex-closure* queries described in Section 3, and shown to be non-expressible in $\mathrm{FO} + \mathrm{linear}$.

Example 5.3. Let $\mathcal{O}$ be an infinite set of operator names $\mathrm{segment}_n$ of signature $[0,n] \to [0,n]$, $n \geqslant 0$, and associate with each operator name $\mathrm{segment}_n$ the operator $\mathrm{op}(\mathrm{segment}_n)$ defined by

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

for each semi-linear set S of $\mathbb{R}^n$. Thus, $\mathrm{op}(\mathrm{segment}_n)(S)$ is the union of all closed line segments with both endpoints in S. The $\mathrm{FO} + \mathrm{linear} + \mathcal{O}$ formula

$$\underbrace{\mathrm{segment}_n(\mathrm{segment}_n(\ldots \mathrm{segment}_n}_{n \text{ times}}(S) \cdots))(\boldsymbol{x})$$

computes the convex closure of S. Using the convex-closure query as a macro, the $\mathrm{FO} + \mathrm{linear} + \mathcal{O}$ formula

$$(\exists d)(\dim_n(\{\boldsymbol{x} \mid \text{convex-closure } (S)(\boldsymbol{x})\}, d) \wedge d \leqslant 1)$$

expresses the collinearity query.

To illustrate the potential of the above paradigm for extending $\mathrm{FO} + \mathrm{linear}$, we show that an existing powerful extension of $\mathrm{FO} + \mathrm{linear}$ can alternatively be captured by extending $\mathrm{FO} + \mathrm{linear}$ with operators.

The language concerned is PFOL, recently introduced by the present authors [50]. The language PFOL was obtained by augmenting $\mathrm{FO} + \mathrm{linear}$ with a limited amount of multiplicative power. As a result, this language allows, for instance, to compute convex closure, Voronoi diagrams, and distances, queries which are not expressible in $\mathrm{FO} + \mathrm{linear}$. On finite databases, i.e., databases which describe finite point sets, PFOL captures all the “ruler-and-compass” queries expressible in SafeEuQl [40]. The language PFOL is a powerful linear query language, yet still feasible for implementation.

First, we define the language PFOL. [16]

We assume *three* sorts of variables, called *non-spatial variables*, *real variables*, and *product variables*. We shall use v, possibly subscripted, to denote a non-spatial variable, $x, y, z, \ldots$, possibly subscripted, to denote real variables, and $p, q, r, \ldots$, possibly

[16] The definition given here is more general than the one in [50], as the latter was stated in a purely spatial context, and, consequently, did not take into account the possible presence of non-spatial data.

subscripted, to denote product variables. Finally, we shall use the symbol t, possibly subscripted, to denote a variable that can either be a real variable or a product variable.

Definition 5.4. (i) Let $D_1,\dots,D_k$, $k \geqslant 0$, be symbols we shall refer to as *domain symbols*. An $\mathrm{FO}+\mathrm{linear}+\mathrm{P}D_1,\dots,D_k$ *formula* is built using the connectives $\neg$ and $\wedge$ and the quantifiers $(\exists v)$, with v a non-spatial variable, $(\exists x)$, with x a real variable, and $(\exists p \in D_i)$, with p a product variable and $1 \leqslant i \leqslant k$, from the following atomic formulae:

– $R(v_1,\dots,v_m;t_1,\dots,t_n)$, with R a relation name of type $[m,n]$, $v_1,\dots,v_m$ non-spatial variables, and $t_1,\dots,t_n$ real variables or product variables;
– $v_1 = v_2$, with v_1 and v_2 non-spatial variables;
– $\sum_{i=1}^{n} a_i t_i \,\theta\, a$ with $a_1,\dots,a_n$, and a real algebraic numbers, $t_1,\dots,t_n$ real variables or product variables, and $\theta \in \{=,\neq,<,\leqslant,>,\geqslant\}$;
– $t_1 = pt_2$, with t_1 and t_2 real variables or product variables and p a product variable; and
– $t = \sqrt{|p|}$ with t a real variable or a product variable and p a product variable.

In addition, each product variable must be bound by an appropriate quantifier.

In the context of sets of real numbers as interpretations for $D_1,\dots,D_k$ and an appropriate linear spatial database, the semantics of an $\mathrm{FO}+\mathrm{linear}+\mathrm{P}D_1,\dots,D_k$ formula is the obvious one.

(ii) A PFOL *program* is of the form

$$D_1 \leftarrow \varphi_1(x);\dots;D_k \leftarrow \varphi_k(x);\{(v_1,\dots,v_m;x_1,\dots,x_n) \mid \varphi(v_1,\dots,v_m;x_1,\dots,x_n)\},$$

with $D_1,\dots,D_k$ domain symbols, for $i=1,\dots,k$, φ_i an $\mathrm{FO}+\mathrm{linear}+\mathrm{P}(D_1,\dots,D_{i-1})$ formula, and $\varphi(v_1,\dots,v_m;x_1,\dots,x_n)$ an $\mathrm{FO}+\mathrm{linear}+\mathrm{P}D_1,\dots,D_k$ formula with free non-spatial variables $v_1,\dots,v_m$ and free real variables $x_1,\dots,x_n$.

The semantics of a PFOL program is as follows. First, $D_1,\dots,D_k$ are consecutively interpreted as *finite* sets of real algebraic numbers. In this process, D_i is interpreted as the set $\{x \mid \varphi_i(x)\}$ if this set is finite, and as the empty set otherwise.[17] Then, $\{(v_1,\dots,v_m;x_1,\dots,x_n) \mid \varphi(v_1,\dots,v_m;x_1,\dots,x_n)\}$ is interpreted in the obvious way.

In [50], it has been shown that PFOL is a proper extension of $\mathrm{FO}+\mathrm{linear}$ which remains sound with respect to the $\mathrm{FO}+\mathrm{poly}^{\mathrm{lin}}$ queries.

We now propose an extension of $\mathrm{FO}+\mathrm{linear}$ with operators of which we will prove that it has the same expressive power as PFOL.

Definition 5.5. (1) The linear operator “line”[18] of type $[0,2]\to[0,2]$ is defined by the $\mathrm{FO}+\mathrm{poly}$ formula $(\exists u)(\exists v)(R(u,v) \wedge uy = vx)$. In words, the operator “line” returns the union of all lines through some point of the input and the origin $(0,0)$.[19]

[17] Remember that finiteness of a semi-linear set is decidable (Proposition 3.2).

[18] From their definitions, it is not obvious that “line” and “sqrt” are linear operators. However, since PFOL can only express linear queries [50], the proof of Lemma 5.8 also entails a proof of the linearity of these operators.

[19] Notice that, if $R(0,0)$ is **true**, then the operator “line” returns the entire plane.

(2) The linear operator "sqrt"[17] of type $[0,1] \to [0,1]$ is defined by the FO + poly formula $(\exists u)(R(u) \wedge x^2 = |u| \wedge x \geqslant 0)$. In words, the operator "sqrt" returns those real numbers that are the square root of a nonnegative real number in the input.

We now prove that FO + linear + {line, sqrt} has the same expressive power as PFOL.

For this purpose, we need the following lemmas.

Lemma 5.6. *Let R be of type* $[0,1]$. *The predicate* product$(R;\, p,x,y)$ *which evaluates to* **true** *if R is finite, p is in R, and $y = px$ can be expressed in* FO + linear + {line} (*whence in* FO + linear + {line, sqrt}).

Proof. By Proposition 3.2, we can check in FO + linear, whence in FO + linear + {line}, whether R is finite. If R is finite, $p = 0$, and p is in R, then product$(R;\, p,x,y)$ evaluates to **true** if and only if $y = 0$. We can thus suffice by explaining what to do if R is finite, $p \neq 0$, and p is in R.

Let $L = \text{line}(\{(x,y) \mid x = 1 \wedge R(y) \wedge y \neq 0\})$. Since the operand is finite and does not contain the origin of the plane, L is a finite union of lines. By Proposition 3.18, we can select in FO + linear, whence in FO + linear + {line}, the line L_p going through the point $(1,p)$. Since this line also goes through the point $(0,0)$, it readily follows that product$(R;\, p,x,y)$ is **true** if and only if the point (x,y) is on L_p. □

Lemma 5.7. *Every* PFOL-*expressible query can be expressed in* FO + linear + {line, sqrt}.

Proof. Consider the PFOL program

$$D_1 \leftarrow \varphi_1(x); \ldots ; D_k \leftarrow \varphi_k(x); \{(v_1,\ldots,v_m; x_1,\ldots,x_n) \mid \varphi(v_1,\ldots,v_m; x_1,\ldots,x_n)\},$$

with $D_1,\ldots,D_k$ domain symbols, for $i = 1,\ldots,k$, φ_i an FO + linear + P$(D_1,\ldots,D_{i-1})$ formula, and φ an FO + linear + P$(D_1,\ldots,D_k)$ formula.

First, we show how to obtain FO + linear + {line, sqrt} expressions which compute the domains $D_1,\ldots,D_k$ according to the semantics of PFOL. The FO + linear + {line, sqrt} expressions are obtained by inductively translating the expressions $\varphi_1,\ldots,\varphi_k$ to FO + linear + {line, sqrt} expressions $\tilde{\varphi}_1,\ldots,\tilde{\varphi}_k$.

For the basis of this induction, we observe that φ_1 is an FO + linear expression, whence also an FO + linear + {line, sqrt} expression. Let $\tilde{\varphi}_1$ be the formula finite$(\{(x) \mid \varphi_1(x)\}) \wedge \varphi_1$, with "finite" the FO + linear query which decides whether a semi-linear set is finite. Clearly, $\tilde{\varphi}_1$, evaluated in the standard manner, yields the correct interpretation of D_1, independent of whether $\{(x) \mid \varphi_1(x)\}$ is finite or not.

Now, assume that, for $1 \leqslant i < k$, there are FO + linear + {line, sqrt} expressions $\tilde{\varphi}_1,\ldots,\tilde{\varphi}_i$ computing the domains $D_1,\ldots,D_i$ according to the semantics of PFOL. Consider the FO + linear + P$(D_1,\ldots,D_i)$ expression φ_{i+1}. In φ_{i+1}, we substitute every subformula of the form $(\exists p \in D_j)\psi$, $1 \leqslant j \leqslant i$, by $(\exists p)(\tilde{\varphi}_j(p) \wedge \tilde{\psi})$, where $\tilde{\psi}$ is the formula obtained from ψ as follows:

- every occurrence of an atomic formula of the form $pt_1 = t_2$ is replaced by the FO + linear + {line, sqrt} formula simulating product($\{(x) \mid \tilde{\varphi}_j(x)\}; p, t_1, t_2$)(Lemma 5.6); and
- every occurrence of an atomic formula of the form $t = \sqrt{|p|}$ is replaced by the FO + linear + {line, sqrt} formula simulating $(p \geqslant 0 \wedge \text{product}(\{(x) \mid \text{sqrt}(\{(x) \mid \tilde{\varphi}_j(x)\})(x)\}; t, t, p)) \vee (p < 0 \wedge \text{product}(\{(x) \mid \text{sqrt}(\{(x) \mid \tilde{\varphi}_j(x)\})(x)\}; t, t, -p))$.

We abbreviate the FO + linear + {line, sqrt} formula obtained in this way as $\hat{\varphi}_{i+1}$. By Lemma 5.6, $\hat{\varphi}_{i+1}$ and φ_{i+1} are equivalent when evaluated in the standard manner. Therefore, the FO + linear + {line, sqrt} expression $\tilde{\varphi}_{i+1} = \text{finite}(\{(x) \mid \hat{\varphi}_{i+1}(x)\}) \wedge \hat{\varphi}_{i+1}(x)$, evaluated in the standard manner, yields the correct interpretation of D_{i+1}, independent of whether $\{(x) \mid \varphi_{i+1}(x)\}$ is finite or not.

Thus, all domains can be computed in FO + linear + {line, sqrt}.

The only thing that remains is to show that the FO + linear + P($D_1, \ldots, D_k$) expression φ can be translated into an FO + linear + {line, sqrt} expression, for which, of course, the same technique applies: every subformula $(\exists p \in D_j)\psi$, $1 \leqslant j \leqslant k$, is replaced by an FO + linear + {line, sqrt} formula $(\exists p)(\tilde{\varphi}_j(p) \wedge \tilde{\psi})$ in the way explained above. The resulting FO + linear + {line, sqrt} formula clearly expresses the same query as the original PFOL program. □

Lemma 5.8. *Every* FO + linear + {line, sqrt}*-expressible query can be expressed in* PFOL.

Proof. It suffices to show that both "line" and "sqrt" can be simulated by a PFOL program to establish the result. The main difficulty in doing so is that, in FO + linear + {line sqrt} both "line" and "sqrt" can take an infinite input. However, it turns out to be always possible to select a finite number of points from the input such that the output can be constructed from the result of the operator applied to this finite set of points and the general structure of the input.

Since the operator "line" takes a two-dimensional set as input, whereas the operator "sqrt" takes a one-dimensional set as input, we start with the latter, simpler, case.

Thus, let R be of type [0, 1]. As observed above, the naive "solution" of using the PFOL program $D_1 \leftarrow R(x); \{(x) \mid (\exists p \in D_1)(x = \sqrt{|p|})\}$ fails, since R may be infinite. Since sqrt(R)(x) is equivalent to sqrt $(\{(x) | R(x) \wedge x \geqslant 0\})(x) \vee \text{sqrt}(\{(x) | R(-x) \wedge x \geqslant 0\})(x)$, we assume in the following that R contains only nonnegative real numbers. *A* semi-algebraic set, in particular also a semi-linear set, has only a finite number of connected components, whence R is either empty or a finite union of isolated points, intervals, and half-lines. If we apply the "sqrt" to the isolated points and the end points of intervals and half lines in R, the output can be obtained from these by "filling in" the intervals and half lines appropriately. Thus, let D_1 be the boundary of R, which consists of all isolated points and end points of line segments and half lines of R. Clearly, D_1 is a finite set which can be computed in FO + linear (Proposition 3.1). The following FO + linear + P(D_1) expression then yields sqrt(R):

$$\{(x) | (\exists p \in D_1)(R(p) \wedge x = \sqrt{|p|})$$

$$\vee (\exists p_1 \in D_1)(\exists p_2 \in D_1)(\exists y_1)(\exists y_2)((\forall z)(p_1 < z < p_2 \Rightarrow R(z))$$
$$\wedge\, y_1 = \sqrt{|p_1|} \wedge y_2 = \sqrt{|p_2|} \wedge y_1 < x < y_2)$$
$$\vee (\exists p \in D_1)(\exists y)((\forall z)(p < z \Rightarrow R(z)) \wedge y = \sqrt{|p|} \wedge y < x)$$
$$\vee (\exists p \in D_1)(\exists y)((\forall z)(p > z \Rightarrow R(z)) \wedge y = \sqrt{|p|} \wedge y > x)\}.$$

We now turn to the operator “line”, which takes a (possibly infinite) two-dimensional semi-linear set as input. Let R, of type $[0,2]$, be the input predicate. Rather than providing the PFOL program, we explain how to construct it.

Since line(R) yields the entire plane if R contains the origin $(0,0)$, we can deal with this case separately and assume for the remainder of the proof that R does *not* contain the origin. We furthermore assume that R is *bounded* and explain afterwards which changes must be made to the reasoning below if R is not necessarily bound.

Again since a semi-linear set has only a finite number of connected components, R is a finite union of isolated points, intervals, and polygons. It is possible to compute these isolated points, the end points of the intervals, and the corner points of the polygons, which together are often called the *special points* of R, in [13]. Let S be the finite set of special points of R, augmented with the points $(1,0)$ and $(0,1)$[20]. If present, we remove the origin (0,0) from S.[21] Let D_1 be the finite set consisting of the coordinates of points of S. Let L be the union of all the lines through the origin and a point of S. Obviously, L can be computed by the PFOL expression

$$\{(x,y) \mid (\exists p \in D_1)(\exists q \in D_1)(S(p,q) \wedge py = qx)\}.$$

The lines of L induce a partition of the plane, consisting of the origin (which always belongs to the output of line(R), except when R is empty), the open half-lines in which the origin divides the lines of L, and the open convex angular sectors between successive half-lines. The output of line(R) consists of the union of all the one- and two-dimensional classes of this partition which have a non-empty intersection with R and their mirror images with respect to the origin, augmented with the origin if R is not empty.

The union of the open half-lines which have a non-empty intersection with R can be computed by the PFOL expression

$$\{(x,y) \mid (\exists p \in D_1)(\exists q \in D_2)(\exists \lambda)(\exists \mu)(\exists z_1)(\exists z_2)(S(p,q)$$
$$\wedge\, \lambda > 0 \wedge \mu > 0 \wedge R(z_1,z_2) \wedge z_1 = \lambda p \wedge z_2 = \lambda q \wedge x = \mu p \wedge y = \mu q\}.$$

The open convex angular sector defined by two *non-collinear* open half-lines consists of all mid-points of a point of the first and a point of the second half-line.[22] Such

[20] The reason for the addition of these two points will be explained a little later.

[21] Notice that S may contain the origin $(0,0)$, even though R does not.

[22] Notice that if we would have worked with lines rather than half-lines, this part of the proof would have failed; the same construction applied to full lines does not yield the union of the relevant angular sector and its mirror image with respect to the origin, but the entire plane.

an angular sector is a class of the partition if the defining half-lines are classes of the partition, and if no other half-line which is a class of the partition has a point in common with the angular sector. Notice that, by the addition of the points $(1,0)$ and $(0,1)$ to S, the partition is guaranteed not to contain angular sectors of 180°, whence the above observations can be used to construct a PFOL expression computing the union of the open convex angular sectors which have a non-empty intersection with R.

Finally, to conclude the case where R is bounded, let U be the union of all one- and two-dimensional classes of the partition which have a non-empty intersection with R. Then the output of line(R) is computed by the PFOL expression

$$\{(x,y) \mid (x=0 \wedge y=0 \wedge (\exists u)(\exists v)R(u,v)) \vee U(x,y)$$
$$\vee(\exists z_1)(\exists z_2)(U(z_1,z_2) \wedge x+z_1=0 \wedge y+z_2=0)\}.$$

In recent work [50], the present authors showed that the above reasoning can be generalized to unbounded semi-linear sets by considering "special points at infinity", which are represented by pairs of directional numbers. (Conceptually, all semi-linear sets can then be treated as if they were bounded.) It was shown that all the required constructions can be simulated in PFOL. This completes the proof. □

Lemmas 5.7 and 5.8 together yield the following conclusion.

Theorem 5.9. *The linear spatial query languages* PFOL *and* FO+linear+{line, sqrt} *have the same expressive power.*

By examining the proofs of the previous lemmas, it also follows that the restriction of PFOL in which square-root terms are disallowed is equivalent to FO+linear+{line}. Alternatively, one may extend PFOL by adding cube-root terms, etc. One can easily see that the above proofs generalize provided FO+linear+{line, sqrt} is extended by operators corresponding to the added terms.

Of course, the above result does not settle the expressiveness of extensions of FO+linear with operators in general. In particular, the question whether there exists an extension of FO+linear with operators which captures precisely the FO+poly$^{\text{lin}}$ queries remains open.

6. Conclusions

In this paper, we studied languages that define FO+poly$^{\text{lin}}$ queries. Amongst these languages, the most natural one is FO+linear.

For this language, we showed that, on the one hand, a wide range of useful complex linear queries can be expressed in it, but, on the other, that equally important linear queries, such as deciding collinearity or computing the convex closure, are not expressible. We also provided a powerful tool to prove the inexpressibility of linear queries in FO + linear by reducing the inexpressibility of the query to the undefinability of an associated semi-algebraic set by linear constraints.

To overcome the limitations of FO + linear, we considered extensions of FO + linear with FO + poly^{lin}-definable operators. Defining such extensions is non-trivial, however, as some naive extensions have the same expressive power as FO + poly. The crux in defining sound extensions was requiring that the added operators can only be applied to set terms *without* free real variables. The possibilities of sound extensions of FO + linear were discussed, and the viability of the paradigm was illustrated by showing that another linear language recently introduced by the authors using a completely different paradigm can be captured in FO + linear extended with operators.

One of the quests in extending FO + linear is finding a syntactic query language precisely capturing all FO + poly^{lin} queries. Dumortier and the present authors [13] showed that such a language exists, but the language they presented is of no practical value. One might wonder whether there exists a suitable set of operators $\mathcal{O}$ such that FO + linear + $\mathcal{O}$ precisely captures all FO + poly^{lin} queries, as such a language could be more practical. This question, however, remains open.

Acknowledgements

The authors wish to thank Ed Robertson, for having suggested Theorem 3.15 to them, and Alfons Ooms, for useful discussions on the proof of Lemma 3.19.

References

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

[2] F. Afrati, S. Cosmadakis, S. Grumbach, G. Kuper, Linear versus polynomial constraints in database query languages, in: A. Borning (Ed.), Proc. 2nd Internat. Workshop on Principles and Practice of Constraint Programming, Rosario, WA, Lecture Notes in Computer Science, Vol. 874, Springer, Berlin, 1994, pp. 181–192.

[3] W.G. Aref, H. Samet, Extending a database with spatial operations, in: O. Günther, H.-J. Schek (Eds.), Proc. 2nd Symp. on Advances in Spatial Databases, Lecture Notes in Computer Science, Vol. 525, Springer, Berlin, 1991, pp. 299–319.

[4] M. Benedikt, G. Dong, L. Libkin, L. Wong, Relational expressive power of constraint query languages, J. ACM 45 (1998) 1–34.

[5] M. Benedikt, L. Libkin, Safe constraint queries, in Proc. 13th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, Seattle, WA, 1998, pp. 99–108.

[6] J. Bochnak, M. Coste, M.F. Roy, Real Algebraic Geometry, in Ergebnisse der Mathematik und ihrer Grenzgebiete, 3. Folge/A Series of Modern Surveys in Mathematics, vol. 36, Springer-Verlag, Berlin, 1998.

[7] A. Brodsky, J. Jaffar, M.J. Maher, Toward practical constraint databases, Constraint 2 (1997) 279–304.

[8] A. Brodsky, Y. Kornatzky, The LyriC language: querying constraint objects, in Proceedings ACM SIGMOD International Conference on Management of Data (San Jose, CA), M.J. Carey and D.A. Schneider, eds., 1995, pp. 35–46; also SIGMOD Record 24 (1995).

[9] J.-H. Byon, P. Revesz, DISCO: a constraint database system with sets, in: G. Kuper, M. Wallace (Eds.), Proc. Workshop on Constraint Databases and Applications, Friedrichshafen, Germany, Lecture Notes in Computer Science, Vol. 1034, Springer, Berlin, 1996, pp. 68–83.

[10] A. Chandra, D. Harel, Computable queries for relational database systems, J. Comput. System Sci. 21 (1980) 156–178.

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

[12] E. Clementini, P. Di Felice, P. van Oosterom, A small set of formal topological relationships suitable for end-user interaction, in: D. Abel, B.C. Ooi (Eds.), Advances in Spatial Databases, Proc. 3th Internat. Symp. on Large Spatial Databases, Lecture Notes in Computer Science, Vol. 692, Springer, Berlin, 1993, pp. 277–295.

[13] F. Dumortier, M. Gyssens, L. Vandeurzen, D. Van Gucht, On the decidability of semi-linearity for semi-algebraic sets and its implications for spatial databases, J. Comput. System Sci. 58 (1999) 535–571.

[14] M.J. Egenhofer, A formal definition of binary topological relationships, in: W. Litwin, H.-J. Schek (Eds.), Proc. Foundations of Data Organization and Algorithms, Lecture Notes in Computer Science, Vol. 367, Springer, Berlin, 1989, pp. 457–472.

[15] M.J. Egenhofer, Reasoning about binary topological relations, in: O. Günther, H.-J. Schek (Eds.), Advances in Spatial Databases, Proc. 2nd Symp. on Very Large Spatial Databases, Lecture Notes in Computer Science, Vol. 525, Springer, Berlin, 1991, pp. 143–160.

[16] M.J. Egenhofer, Why not SQL! Internat. J. Geogr. Inform. Systems 6 (1992) 71–85.

[17] M.J. Egenhofer, Spatial SQL: a query and presentation language, IEEE Trans. Knowledge Data Eng. 6(1) (1994) 86–95.

[18] M.J. Egenhofer, J. Herring, A mathematical framework for the definition of topological relationships, in: K. Brassel, H. Kishimoto (Eds.), Proc. 4th Internat. Symp. on Spatial Data Handling, Zurich, Switzerland, 1990, pp. 803–813.

[19] S. Grumbach, G. Kuper, Tractable recursion over geometric data, in: G. Smolka (Ed.), Proc. 3rd Internat. Conf. on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science, Vol. 1330. Springer, Berlin, 1997, pp. 450–462.

[20] S. Grumbach, P. Rigaux, L. Segoufin, The DEDALE system for complex spatial queries, in: L. Haas, A. Tiwary (Eds.), Proc. ACM SIGMOD Internat. Conf. on Management of Data, Seattle, WA, 1998, pp. 213–224; also SIGMOD Record 27 (1998).

[21] S. Grumbach, J. Su, Towards practical constraint databases, in Proc. 15th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, Montreal, Canada, 1996, pp. 28–39.

[22] S. Grumbach, J. Su, C. Tollu, Linear constraint query languages: expressive power and complexity, in: D. Leivant (Ed.), Proc. Logic and Computational Complexity Workshop, Indianapolis, IN, 1994.

[23] O. Günther (Ed.), Efficient structures for geometric data management, in Lecture Notes in Computer Science, Vol. 337, Springer, Berlin, 1988.

[24] O. Günther, Research issues in spatial databases, SIGMOD Record 19(4) (1990) 61–68.

[25] R.H. Güting, Geo-Relational Algebra: a model and query language for geometric database systems, in: J.W. Schmidt, S. Ceri, M. Missikoff (Eds.), Advances in Database Technology-EDBT '88, Proc. Internat. Conf. on Extending Database Technology, Venice, Italy, Lecture Notes in Computer Science, Vol. 303, Springer, Berlin, 1988, pp. 506–527.

[26] R.H. Güting, GRAL: an extensible relational database system for geometric applications, in Proc. 15th Internat. Conf. on very Large Databases, Amsterdam, Netherlands, 1989, pp. 33–44.

[27] R.H. Güting, An introduction to spatial database systems VLDB J. 3(4) (1994) 357–399

[28] R.H. Güting, T. de Ridder, M. Schneider, Implementation of the ROSE Algebra: efficient algorithms for realm-based spatial data types, in: M.J. Egenhofer, J.R. Herring (Eds.), Advances in Spatial Databases, Proc. 4th Internat. Symp. on Large Spatial Databases, Lecture Notes in Computer Science, Vol. 951, Springer, Berlin, 1995, pp. 216–239.

[29] R.H. Güting, M. Schneider, Realms: a foundation for spatial data types in database systems, in Proc. 3rd Internat. Conf. on very Large Databases, Singapore, 1993, pp. 14–35.

[30] M. Gyssens, J. Van den Bussche, D. Van Gucht, Complete geometrical query languages, J. Comput. System Sci. 58 (1999) 483–511.

[31] R. Hull, C.K. Yap, The format model, a theory of database organization J. ACM 31(3) (1984) 518–537

[32] P.C. Kanellakis, D.Q. Goldin, Constraint programming and database query languages, in: M. Hagiya, J.C. Mitchell (Eds.), Proc. 2nd Conf. on Theoretical Aspects of Computer Software, Sendai, Japan, Lecture Notes in Computer Science, Vol. 789, Springer, Berlin, 1994, pp. 96–120.

[33] P.C. Kanellakis, G.M. Kuper, P.Z. Revesz, Constraint query languages J. Comput. System Sci. 51 (1995) 26–52.
[34] A. Kemper, M. Wallrath, An analysis of geometric modeling in database systems, Comput. Surv. 19 (1987) 47–91.
[35] G. Kuper, On the expressive power of the relational calculus with arithmetic constraints, in: S. Abiteboul, P.C. Kanellakis (Eds.), Proc. 3rd Internat. Conf. on Database Theory, Paris, France, Lecture Notes of Computer Science, Vol. 470, Springer, Berlin, 1990 pp. 202–211.
[36] J.L. Lassez, Querying constraints, in Proc. 9th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, Nashville, TN, 1990, pp. 288–298.
[37] C.B. Medeiros, F. Pires, Databases for GIS SIGMOD Record 23 (1994) 107–115
[38] J. Nievergelt, M. Freeston (Eds.), Special issue on spatial data, Comput. J. 37 (1994).
[39] J. Paredaens, Spatial databases, the final frontier, in: G. Gottlob, M.Y. Vardi (Eds.), Proc. 5th Internat. Conf. on Database Theory, Prague, Czech Republic, Lecture Notes of Computer Science, Vol. 893, Springer, Berlin, 1995, pp. 14–32.
[40] J. Paredaens, B. Kuijpers, G. Kuper, L. Vandeurzen, Euclid, Tarski, and Engeler encompassed, in: S. Cluet, R. Hull (Eds.), Database Programming Languages, Proc. 6th Internat. Workshop, DBPL-6, Estes Park, CO, 1997, Lecture Notes of Computer Science, Vol. 1369, Springer, Berlin, 1998, pp. 1–24.
[41] J. Paredaens, J. Van den Bussche, D. Van Gucht, Towards a theory of spatial database queries, in Proc. 13th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, Minneapolis, MN, 1994, pp. 279–288.
[42] J. Paredaens, J. Van den Bussche, D. Van Gucht, First-order queries on finite structures over the reals, SIAM J. Comput 27 (1998) 1747–1763.
[43] N. Pissinou, R. Snodgrass, R. Elmasri, I. Mumick, T. Özsu, B. Pernici, A. Segef, B. Theodoulidis, U. Dayal, Towards an infrastructure for temporal databases SIGMOD Record 23 (1994) 35–51.
[44] L. Segoufin, Victor Vianu, Spatial databases via topological invariants, in Proc. 18th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, Seattle, WA, 1998, pp. 89–98.
[45] P. Svensson, Z. Huang, GEO-SAL: a query language for spatial data analysis, in: O. Günther, H.-J. Schek (Eds.), Advances in Spatial Databases, Proc. 2nd Symp. on Very Large Spatial Databases, Lecture Notes in Computer Science, Vol. 525. Springer, Berlin, 1991, pp. 119–140.
[46] A. Tarski, A Decision Method for Elementary Algebra and Geometry, University of California Press, Berkeley, CA, 1951.
[47] J.D. Ullman, Principles of Database Systems, 2nd Edition, Pitman Publishing, London, 1982.
[48] L. Vandeurzen, M. Gyssens, D. Van Gucht, On the desirability and limitations of linear spatial query languages, in: M.J. Egenhofer, J.R. Herring (Eds.), Advances in Spatial Databases, Proc. 4th Internat. Symp. on Large Spatial Databases, Lecture Notes in Computer Science, Vol. 951, Springer, Berlin, 1995, pp. 14–28.
[49] L. Vandeurzen, M. Gyssens, D. Van Gucht, On query languages for linear queries definable with polynomial constraints, in: E.C. Freuder (Ed.), Proc. 2nd Internat. Conf. on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science, Vol. 1118, Springer, Berlin, 1996, pp. 468–481.
[50] L. Vandeurzen, M. Gyssens, D. Van Gucht, An expressive language for linear spatial database queries, in Proc. 18th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, Seattle, WA, 1998, pp. 109–118.