Complete geometrical query languages

(extended abstract)

Marc Gyssens
University of Limburg[*]
gyssens@luc.ac.be

Jan Van den Bussche
University of Limburg[*]
vdbuss@luc.ac.be

Dirk Van Gucht
Indiana University[†]
vgucht@cs.indiana.edu

Abstract

We introduce query languages for spatial databases that are complete, in the sense that they can express precisely all computable queries that are generic with respect to certain classes of transformations of space, corresponding to certain geometric interpretations of spatial data. We thus extend Chandra and Harel's seminal work on computable queries for relational databases to a spatial setting. We use a constraint-based spatial data model which models spatial data as semi-algebraic relations over the real numbers. We also introduce natural point-based geometric query languages that are complete relative to the basic class of queries expressible in the relational calculus with real polynomial constraints.

1 Introduction

In their seminal work on computable queries for relational databases [2], Chandra and Harel introduced the notion of *computable query* as a computable function from relational databases to relations that is invariant under all permutations of the universe of atomic data elements. The latter criterion, now known as genericity, states that queries should preserve database isomorphisms, or, more intuitively, that they should be defined at the logical level of the data in the database. Chandra and Harel then introduced a query language, QL, and proved it complete, in the sense that precisely all computable queries can be expressed in QL.

The purpose of the present paper is to continue Chandra and Harel's work in the setting of spatial databases. Spatial data can be viewed conceptually as possibly infinite sets of points in n-dimensional space.

The question of how the concept of genericity extends to the spatial setting was already considered by Paredaens, Van den Bussche, and Van Gucht [7] who, in short, argued as follows. It makes no sense to require that queries are invariant under *all* permutations of the universe. Indeed, spatial data have a certain geometrical interpretation, depending on the application. It is standard mathematical practice to identify a geometry with a group of transformations of space. Hence, the geometrical interpretation of the spatial data corresponds to a group G of transformations. We can thus define a query to be G-*generic* if it is invariant under all transformations in G, or, more intuitively, if it is defined at the intended geometrical level of the data in the database.

We work in an adaptation of the relational model, where the universe of atomic data elements is n-dimensional space, and where relations can be infinite. To ensure finite representability, the relations must be elementarily definable in terms of polynomial inequalities (in mathematical terminology, they must be "semi-algebraic"). Our model is thus an instance of the framework of *constraint databases* introduced by Kanellakis, Kuper, and Revesz [4].

Our search for complete spatial query languages starts with the simplest case, where n equals 1 and the group G consists only of the identity. So, points are real numbers which are fully interpreted; genericity is trivial, and the databases under consideration are what is known as *real polynomial constraint databases* [4]. We observe that first-order logic augmented with polynomial inequalities, relation variables of fixed arities, and while-loops, yields a complete query language in this case, which we denote by FO[R] + while. It is instructive to contrast this result to Chandra and Harel's, where the language QL necessarily involves "unranked" relation variables which can hold relations of any arity.

We then use this result to find complete geometrical query languages under various geometrical interpretations. These languages are all syntactically identical to FO[R] + while, but in each case the semantics of a program is appropriately defined so as to be guaranteed generic. This is accomplished by working on canonical representations of databases, rather than on the databases themselves.

The approach to finding complete languages just described yields languages with a very artificial semantics. However, we can obtain much more natural results when focusing on queries expressible in the basic query language of first-order logic with polynomial inequalities (i.e., without while-loops), which we denote by FO[R], and which is known as the *real polynomial constraint calculus* [4]. We introduce first-order query languages that do not have access to the specific coordinates of points but only to the points themselves as atomic objects. Rather than augmenting first-order logic with polynomial inequalities on real numbers,

[*]LUC, Department WNI, Universitaire Campus, B-3590 Diepenbeek, Belgium.
[†]Computer Science Department, Bloomington, IN 47405-4101, USA.

Permission to make digital/hard copies of all or part of this material for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee
PODS '97 Tucson Arizona USA
Copyright 1997 ACM 0-89791-910-6/97/05 ..$3.50

these query languages provide certain built-in geometrical predicates on points. For example, we show that providing the predicate between(p, q, r) (with the obvious meaning) yields a query language that expresses exactly all FO[R]-queries that are generic for affine geometry. This result is interesting because G-genericity of FO[R] queries is undecidable for every non-trivial G [7]. Our proof, which exploits the classical geometrical construction of addition and multiplication, is inspired by by the work by Tarski and his collaborators on axiomatizations of elementary geometry [9, 10, 8].

Complete generic query languages relative to FO[R] were first discovered by Kuijpers, Paredaens, and Suciu [5]. Our results improve upon theirs in the sense that our languages are purely point-based, while the languages of [5] involve both variables ranging over points and variables ranging over real numbers. Also Papadimitriou, Suciu, and Vianu [6] obtained relative completeness results for point-based query languages in the context of a different type of genericy than the geometric types of genericity we consider.

This paper is organized as follows. The database model is presented in Section 2. The notion of generic query is reviewed in Section 3. The complete query languages based on FO[R] + while are presented in Section 4. The geometric completeness results are presented in Section 5. Concluding remarks are given in Section 6.

2 Semi-algebraic and geometric databases

In this section, we define semi-algebraic and geometric databases.

Both database models are described using the first-order language of the ordered field of the real numbers $(\mathbf{R}, \leq, +, \times, 0, 1)$, i.e., the language

$$(\leq, +, \times, 0, 1).$$

A first-order formula in this language is called a *real formula*. By Tarski's theorem [9, 11], every real formula can effectively be transformed into an equivalent quantifier-free one (equivalent in $\mathbf{R}$). So we can implicitly assume real formulas to be quantifier-free. A consequence of Tarski's theorem is that truth of real sentences in $\mathbf{R}$ is effectively decidable.

Let $k \geq 0$. A subset A of $\mathbf{R}^k$ is *defined* by a real formula $\varphi(x_1, \ldots, x_k)$ if

$$A = \{(a_1, \ldots, a_k) \in \mathbf{R}^k \mid \varphi(a_1, \ldots, a_k)\}.$$

A subset of $\mathbf{R}^k$ is called *semi-algebraic* if it can be defined by a real formula. Rephrased in other terminology, a semi-algebraic set is a finite union of sets that can be defined by a system of polynomial inequalities with integer coefficients.

A *semi-algebraic database* is essentially a store of semi-algebraic sets. To define this formally, we recall that a *relational schema* is a finite set σ of relation names, where each relation name is assigned an arity. The notion of "structure" used in the following definition is the one of mathematical logic [3].

Definition 1 Let σ be a relational schema. A *semi-algebraic database* over σ is a structure $\mathcal{D}$ over σ with domain $\mathbf{R}$ such that, for each relation name R of σ, $R^{\mathcal{D}}$ is a semi-algebraic subset of $\mathbf{R}^k$, where k is the arity of R in σ.

Semantically, a semi-algebraic database can be seen as a relational database, with the exception that the relations may be infinite, as semi-algebraic sets may be infinite. Syntactically, however, a semi-algebraic database can be described finitarily using a (quantifier-free) real formula for each relation name in the schema of the database. We formally define a *representation* of a semi-algebraic database as follows:

Definition 2 Let σ be a relational schema and let $\mathcal{D}$ be a semi-algebraic database over σ. A function Φ from the relation names of σ to real formulas is a *representation* of $\mathcal{D}$ if, for each relation name R of σ, $\Phi(R)$ defines $R^{\mathcal{D}}$.

An important application of semi-algebraic databases are the so-called spatial or geometric databases. From now on, we reserve n to denote the dimension of a geometric space which we shall identify with $\mathbf{R}^n$. Let $k \geq 0$. We shall call a k-ary relation on $\mathbf{R}^n$ semi-algebraic if its image under the canonical bijection between $(\mathbf{R}^n)^k$ and $\mathbf{R}^{nk}$ is a semi-algebraic subset of $\mathbf{R}^{nk}$.

Definition 3 Let σ be a relational schema. A *geometric database* over σ in $\mathbf{R}^n$ is a structure $\mathcal{D}$ over σ with domain $\mathbf{R}^n$ such that, for each relation name R of σ, $R^{\mathcal{D}}$ is semi-algebraic.

A geometric database $\mathcal{D}$ over σ in $\mathbf{R}^n$ can be viewed naturally as a semi-algebraic database $\overline{\mathcal{D}}$ over the schema $\overline{\sigma}$, which has, for each relation name R of σ, a relation name $\overline{R}$ with $\overline{\sigma}(\overline{R}) = n\sigma(R)$. For each relation name R, of arity k, $\overline{R}^{\overline{\mathcal{D}}}$ is obtained from $R^{\mathcal{D}}$ by applying the canonical bijection between $(\mathbf{R}^n)^k$ and $\mathbf{R}^{nk}$.

3 Semi-algebraic and geometric queries

In this section, we define algebraic and geometric queries and review a classification for geometric queries.

Definition 4 Let σ be a relational schema. A k-ary *semi-algebraic query* Q over σ is a partial function on the set of semi-algebraic databases over σ, satisfying the following conditions:

1. for each semi-algebraic database $\mathcal{D}$ over σ on which Q is defined, $Q(\mathcal{D})$ is a semi-algebraic subset of $\mathbf{R}^k$;
2. there is an algorithm taking representations of semi-algebraic databases as input satisfying the following conditions:
   (a) for each semi-algebraic database $\mathcal{D}$ over σ, and for each representation Φ of $\mathcal{D}$, the algorithm terminates on input Φ if and only if Q is defined on $\mathcal{D}$; and
   (b) for each semi-algebraic database $\mathcal{D}$ on which Q is defined, and for each representation Φ of $\mathcal{D}$, the output of the algorithm on Φ is a real formula defining $Q(D)$.

The second condition in Definition 4 indicates in which sense semi-algebraic queries are computable.

In analogy to Definition 4, we define *geometric queries*.

Definition 5 Let σ be a relational schema. A k-ary *geometric query* Q over σ in $\mathbf{R}^n$ is a partial function on the set of geometric databases over σ, satisfying the following conditions:

1. for each geometric database $\mathcal{D}$ over σ on which Q is defined, $Q(\mathcal{D})$ is a semi-algebraic subset of $(\mathbf{R}^n)^k$;

2. Q is computable in the sense of Definition 4 (where "semi-algebraic" is replaced by "geometric").

Since geometric databases can be seen as special kinds of semi-algebraic databases, a comparison of Definitions 4 and 5 reveals that geometric queries can be seen as special kinds of semi-algebraic queries.

In the geometric database model, the result of many natural queries does not depend on the particular coordinates assigned to points by the canonical coordinate system in the space considered. More precisely, natural geometric queries preserve coordinate system transitions. The coordinate transitions that must be considered, of course, depend on the geometry of the space, which can be described by a group of transformations (permutations) of space. Therefore, we adopt the following general notion of genericity, originally proposed by Paredaens, Van den Bussche, and Van Gucht [7].

Definition 6 Let σ be a relational schema and let Q be a geometric query in $\mathbf{R}^n$, and let G be a group of transformations of $\mathbf{R}^n$. Then Q is called *G-generic* if for any two geometric databases $\mathcal{D}$ and $\mathcal{D}'$, if $\mathcal{D}' = g(\mathcal{D})$ for some transformation $g \in G$, then $Q(\mathcal{D}') = g(Q(\mathcal{D}))$.

In affine geometry, for instance, G is the group of affinities (affine transformations), and the corresponding class of queries is called the affine-generic queries. Under an affine interpretation of certain two-dimensional spatial data, it would make no sense to ask for all points in the database lying in the unit disk, as this is not an affine-generic query (points can be moved out of the unit disk by applying a translation, which is an affine transformation). It would make sense to ask for all straight lines contained in the database, as this is affine-generic; the collinearity of three points cannot be altered by applying an affinity.

Besides affine-genericity, there are several other notions of genericity that correspond to sensible geometry. We summarize some of them below:

- Similarity genericity, with respect to the group of similarities, i.e., compositions of isometries (see below) and scalings. This genericity notion corresponds to Euclidean geometry.

- Isometry genericity, with respect to the isometries, i.e., compositions of translations, rotations, and reflections. This genericity notion corresponds to the fragment of Euclidean geometry where absolute rather than relative measures are important.

- Direct-isometry genericity, with respect to the direct isometries, i.e., compositions of translations and rotations. This genericity notion corresponds to the fragment of the previous geometry where also orientation is important. In this geometry, two objects are considered isomorphic if one can be mapped to the other by a rigid motion.

- Translation genericity, with respect to the translations. This genericity notion corresponds to the fragment of the previous geometry where also the relative position of objects (e.g., above or left of) is important.

4 Complete languages for semi-algebraic queries

In this section, we consider the query languages FO[$\mathbf{R}$] and FO[$\mathbf{R}$] + while and show that the latter language expresses exactly all semi-algebraic queries. We then show how the semantics of programs in this language can be modified so as to be guaranteed generic, yielding query languages expressing exactly all generic geometric queries for a wide variety of geometries.

4.1 Semi-algebraic queries

Let σ be a relational schema. A first-order formula

$$\varphi(x_1, \ldots, x_k)$$

in the language of the real numbers augmented with the relation names of σ defines on each semi-algebraic database $\mathcal{D}$ over σ a subset $\varphi(\mathcal{D})$ of $\mathbf{R}^k$ in the standard manner. Since $\varphi(\mathcal{D})$ is obviously semi-algebraic, φ thus defines a k-ary semi-algebraic query over σ. The basic query language obtained by all such formulas φ is denoted by FO[$\mathbf{R}$].

We can extend FO[$\mathbf{R}$] into a full-fledged programming language FO[$\mathbf{R}$] + while as follows. A *program* over σ is a finite sequence of *statements* and *while-loops*. Each statement has the form

$$R := \{(x_1, \ldots, x_k) \mid \varphi(x_1, \ldots, x_k)\},$$

with R a relation variable and φ a first-order formula in the language of the real numbers augmented with the relation names of σ and the previously introduced relation variables. Each while-loop has the form while φ do P, where P is a program and φ is a first-order sentence in the language of the real numbers augmented with the relation names of σ and the previously introduced relation variables.

Semantically, a program in the query language FO[$\mathbf{R}$] + while expresses a semi-algebraic query in the obvious way as soon as one of its relation variables has been designated as the output variable. Of course, since while-loops need not terminate, this query will in general no longer be totally defined (as is the case with FO[$\mathbf{R}$] queries).

As announced, we can show that FO[$\mathbf{R}$] + while is complete for the semi-algebraic queries.

Theorem 1 *Every semi-algebraic query is expressible in the query language* FO[$\mathbf{R}$] + while.

Proof. We provide a rough sketch of the proof.

Let Q be a k-ary semi-algebraic query over a schema σ. Let K be the maximum of k and the arities of relation names of σ. Then every relation in a semi-algebraic database over σ can be defined by a quantifier-free real formula using only the variables $x_1, \ldots, x_K$. We may assume that these formulas are encoded as natural numbers in such a way that the encoding of a subterm or subformula occurring in another term or formula comes before the encoding of that term or formula. Note that the language FO[$\mathbf{R}$] + while provides full computational power on the natural numbers.

We can write a program that builds up relations T (for term) and F (for formula). The arity of T is $K+2$; each tuple in T is of the form $(t, a_1, \ldots, a_K, \tau)$, where t is the encoding of a term, $a_1, \ldots, a_K$ are real numbers, and τ is the value of t evaluated under the valuation $x_i \mapsto a_i$. The arity of F is $K + 1$; each tuple in F is of the form $(f, a_1, \ldots, a_K)$, where f is the encoding of a formula and $f(a_1, \ldots, a_K)$ is true. We can also write a similar program which, when applied to the encoding f of a real formula ϕ, computes the semi-algebraic subset of $\mathbf{R}^K$ defined by ϕ.

The theorem now follows readily. To compute the result of Q applied to a database we first compute the encodings of formulas defining the relations of the database, which gives us (an encoding of) a representation of the datatabase. We then perform the computation on this representation prescribed by the algorithm for computing Q, leading to a resulting natural number. Finally we decode this natural number into the result relation of Q. ■

4.2 Geometric queries

Let σ be a relational schema and let G be a group of transformations of space (we will work in the plane, i.e, $\mathbf{R}^2$, to simplify the notation). Representations of geometric databases over σ are essentially strings over some finite alphabet and hence can be compared lexicographically. We can thus define:

Definition 7 1. Two geometric databases $\mathcal{D}$ and $\mathcal{D}'$ are called *G-isomorphic* if $\mathcal{D}' = g(\mathcal{D})$ for some $g \in G$.

2. The *G-canonization* of a geometric database $\mathcal{D}$, denoted by $canon_G(\mathcal{D})$, is the geometric database $\mathcal{D}'$ G-isomorphic to $\mathcal{D}$ having a representation that is minimal among all geometric databases G-isomorphic to $\mathcal{D}$.

3. The *G-type* of $\mathcal{D}$, denoted by $Type_G(\mathcal{D})$, equals $\{g \in G \mid g(\mathcal{D}) = canon_G(\mathcal{D})\}$.

Canonization can effectively be carried out for a wide variety of groups G. It suffices for G to be identifiable with a semi-algebraic subset of $\mathbf{R}^m$, for some fixed m, such that the "graph" of G, the set $\{(g, x, y, x', y') \mid g \in G \wedge g(x,y) = (x', y')\}$ is also semi-algebraic. If this is the case we call G *semi-algebraic*. Most naturally occurring groups are semi-algebraic; in particular, all groups considered in Section 3 are. For example, an affinity is given by a regular 2×2-matrix and a 2-vector. We can thus identify the group of affinities with the semi-algebraic set

$$\{(a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2) \mid a_{11}a_{22} - a_{12}a_{21} \neq 0\},$$

and the graph of this group equals

$$\{(a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2, x, y, x', y') \mid \\ x' = a_{11}x + a_{12}y + b_1 \wedge y' = a_{21}x + a_{22}y + b_2\},$$

clearly semi-algebraic.

For semi-algebraic G, we can compute a representation of $canon_G(\mathcal{D})$ from a representation of $\mathcal{D}$ by enumerating all representations of databases $\mathcal{D}'$ until we find one for which $(\exists g \in G)(g(\mathcal{D}) = \mathcal{D}')$ is true. This condition is a real sentence and therefore decidable by Tarski's theorem.

We are now ready to define a modified semantics of programs, in accordance with some (semi-algebraic) group G. If P is a program expressing a geometric query and $\mathcal{D}$ a database, then we define

$$P^G(\mathcal{D}) := \bigcup \{g^{-1}(P(canon_G(\mathcal{D}))) \mid g \in Type_G(\mathcal{D})\}.$$

We can show the following for semi-algebraic G: (proof omitted)

Theorem 2 *The partial function mapping $\mathcal{D}$ to $P^G(\mathcal{D})$ for each geometric database $\mathcal{D}$ is a G-generic geometric query (in particular, it is computable). Moreover, if P already expresses a G-generic query then $P^G(\mathcal{D}) = P(\mathcal{D})$ for each geometric database $\mathcal{D}$.*

In this manner, we can produce complete, generic, geometric query languages for a wide variety of geometries. Note that all these languages are syntactically identical to FO[**R**] + while and are thus very artificial.

5 Complete languages for geometric queries

In this section, we propose a family of query languages FO[Π], parameterized by sets Π of so-called point predicates. We then proceed by identifying several members of this family and showing that each of these is sound and complete for a natural genericity-class of geometric queries expressible in FO[**R**]. We work in the plane for simplicity of exposition, but the results carry over straightforwardly to higher dimensions.

We recall that the domain of a geometric database in the plane is $\mathbf{R}^2$, i.e., the plane itself, the elements of which we naturally call *points*. In the logic-based query languages we will define, the variables stand for points (as opposed to real numbers). Thus the predicates used in these languages are evaluated over the set of points of the plane (as opposed to the set of real numbers) and will therefore be referred to as *point predicates*.

Apart from relation names, we shall in particular consider the following point predicates:

- **between**(p, q, r) denotes that p lies on the closed line segment between q and r;
- **equidistance**(p, q, r, s) denotes that the distance between p and q equals the distance between r and s;
- **unitdistance**(p, q) denotes that the distance between p and q equals 1;
- **positive**(p, q, r) denotes that $(\vec{pq}, \vec{pr})$ is positively oriented (oriented in the same way as the standard coordinate system), i.e., that the smallest angle between the vectors $\vec{pq}$ and $\vec{pr}$ is oriented counterclockwise;
- **left**(p, q) denotes that p is to the left of q (i.e., the first coordinate of p is less than or equal to the first coordinate of q);
- **below**(p, q) denotes that p is below q (i.e., the second coordinate of p is less than or equal to the second coordinate of q).

Now let Π be a finite set of point predicates such as the ones above, and let σ be a relational schema. A first-order formula $\varphi(x_1, \ldots, x_k)$ over the relation names of σ and the predicate names in Π defines on each geometric database $\mathcal{D}$ over σ a subset $\varphi(\mathcal{D})$ of $(\mathbf{R}^2)^k$ in the standard manner. Note that variables now range over $\mathbf{R}^2$ instead of $\mathbf{R}$, i.e., points instead of coordinates.

Let us assume that the predicates in Π can be defined by real formulas in terms of the explicit coordinates of points. This is the case for all point predicates considered in this section. For example, **unitdistance**(p, q) can be defined by $(p_x - q_x)^2 + (p_y - q_y)^2 = 1$. Under this assumption, φ is equivalent to an FO[**R**]-formula ψ over the schema $\overline{\sigma}$

Table 1: Sound and complete point-predicate languages for geometric genericity notions.

Genericity notion	Point predicate set Π
Affine genericity	{**between**}
Similarity genericity	{**between**, **equidistance**}
Isometry genericity	{**between**, **equidistance**, **unitdistance**}
Direct-isometry genericity	{**between**, **equidistance**, **unitdistance**, **positive**}
Translation genericity	{**between**, **equidistance**, **unitdistance**, **left**, **below**}

corresponding to σ (cf. Section 2). Hence, $\varphi(\mathcal{D})$ will be semi-algebraic and thus φ defines a k-ary geometric query over σ. The query language obtained is denoted by FO[Π].

We now show the following completeness theorem:

Theorem 3 *The query language* FO[Π] *expresses exactly all G-generic queries expressible in* FO[**R**], *with G and* Π *as listed in Table 1.*

Proof. We provide a sketch of the proof for the query language FO[**between**].

First, we observe that queries expressed in FO[**between**] are indeed affine-generic, since FO[$\emptyset$] preserves arbitrary permutations of the plane and the ternary betweenness relation on $\mathbf{R}^2$ is invariant under all affine transformations of the plane.

Now, consider a k-ary affine-generic geometric query in the plane over the schema σ, expressed by an FO[**R**]-formula $\psi(x_1, y_1, \ldots, x_k, y_k)$ over the schema $\overline{\sigma}$.

We are going to simulate formulas in FO[**R**] by formulas in FO[**between**] that are parameterized by a basis, i.e., a three-tuple of points (o, e_1, e_2) such that the vectors $\overrightarrow{oe_1}$ and $\overrightarrow{oe_2}$ are linearly independent. It is easy to construct a first-order formula *basis* in the language (**between**) such that $basis(o, e_1, e_2)$ if and only if (o, e_1, e_2) is a basis.

In a basis (o, e_1, e_2) we associate to the real number ρ the point p on the line oe_1 for which $\overrightarrow{op} = \rho\overrightarrow{oe_1}$ (i.e., the point on the first coordinate axis with coordinate ρ). Conversely, each point p on the line oe_1 is associated to the real number ρ for which $\overrightarrow{op} = \rho\overrightarrow{oe_1}$. We shall denote this real number ρ as $\overrightarrow{op}/\overrightarrow{oe_1}$.

It is known (e.g., [8]) that the arithmetic operations on these numbers are first-order expressible in the language (**between**). So, we may assume the existence of the following formulas, where p, q, and r must be points on the line oe_1:

- $less(o, e_1, e_2, p, q)$ holds if $\frac{\overrightarrow{op}}{\overrightarrow{oe_1}} \leq \frac{\overrightarrow{oq}}{\overrightarrow{oe_1}}$;
- $plus(o, e_1, e_2, p, q, r)$ holds if $\frac{\overrightarrow{op}}{\overrightarrow{oe_1}} + \frac{\overrightarrow{oq}}{\overrightarrow{oe_1}} = \frac{\overrightarrow{or}}{\overrightarrow{oe_1}}$; and
- $times(o, e_1, e_2, p, q, r)$ holds if $\frac{\overrightarrow{op}}{\overrightarrow{oe_1}} \times \frac{\overrightarrow{oq}}{\overrightarrow{oe_1}} = \frac{\overrightarrow{or}}{\overrightarrow{oe_1}}$.

We also introduce the predicate

$$coordinates(o, e_1, e_2, p, p_1, p_2),$$

with p an arbitrary point in the plane and p_1 and p_2 points on the line oe_1, which holds if

$$\overrightarrow{op} = \frac{\overrightarrow{op_1}}{\overrightarrow{oe_1}}\,\overrightarrow{oe_1} + \frac{\overrightarrow{op_2}}{\overrightarrow{oe_1}}\,\overrightarrow{oe_2}.$$

Again, this predicate can be expressed as a first-order formula in the language (**between**).

Finally, we observe that the predicate

$$collinear(p, q, r),$$

with p, q, and r arbitrary points, which holds if p, q, and r are collinear, can be expressed as

$$\mathbf{between}(p, r, q) \vee \mathbf{between}(r, p, q) \vee \mathbf{between}(q, p, r).$$

Using the predicates introduced above, one can show by structural induction that, for each FO[**R**] formula

$$\xi(u_1, \ldots, u_\ell)$$

over $\overline{\sigma}$, there exists an FO[**between**] formula

$$\widehat{\xi}(\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{u}_1, \ldots, \widehat{u}_\ell)$$

over σ, with $\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{u}_1, \ldots, \widehat{u}_\ell$ free point variables, such that for each basis (o, e_1, e_2) and for all points $p_1, \ldots, p_\ell$ on the line oe_1, $\widehat{\xi}(o, e_1, e_2, p_1, \ldots, p_\ell)$ if and only if

$$\xi(\overrightarrow{op_1}/\overrightarrow{oe_1}, \ldots, \overrightarrow{op_\ell}/\overrightarrow{oe_1}).$$

In particular, the translation is done as follows for relation names. If R is an ℓ-ary relation name in σ, then the 2ℓ-ary relation predicate $\overline{R}(x_1, y_1, \ldots, x_\ell, y_\ell)$ is transformed into

$$(\exists\widehat{v}_1)\ldots(\exists\widehat{v}_\ell)(R(\widehat{v}_1, \ldots, \widehat{v}_\ell) \wedge \bigwedge_{i=1}^{\ell} coordinates(\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{v}_i, \widehat{x}_i, \widehat{y}_i)).$$

Now consider the following formulas φ' and φ in the language FO[**between**]:

$$\varphi'(\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{x}_1, \widehat{y}_1, \ldots, \widehat{x}_k, \widehat{y}_k) \equiv basis(\widehat{z}_0, \widehat{z}_1, \widehat{z}_2) \wedge \bigwedge_{i=1}^{k} collinear(\widehat{z}_0, \widehat{z}_1, \widehat{x}_i) \wedge collinear(\widehat{z}_0, \widehat{z}_1, \widehat{y}_i) \wedge \widehat{\psi}(\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{x}_1, \widehat{y}_1, \ldots, \widehat{x}_k, \widehat{y}_k);$$

$$\varphi(\widehat{v}_1, \ldots, \widehat{v}_k) \equiv (\exists\widehat{z}_0)(\exists\widehat{z}_1)(\exists\widehat{z}_2)(\exists\widehat{x}_1)(\exists\widehat{y}_1)\ldots(\exists\widehat{x}_k)(\exists\widehat{y}_k)(\varphi'(\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{x}_1, \widehat{y}_1, \ldots, \widehat{x}_k, \widehat{y}_k) \wedge \bigwedge_{i=1}^{k} coordinates(\widehat{z}_0, \widehat{z}_1, \widehat{z}_2, \widehat{v}_i, \widehat{x}_i, \widehat{y}_i)).$$

Formula φ expresses a k-ary geometric query over σ.

To see what φ does, let $\mathcal{D}$ be a geometric database over σ. Let (o, e_1, e_2) be an arbitrary basis of the plane $\mathbf{R}^2$ and let α be the affine transformation mapping the standard basis of $\mathbf{R}^2$ to (o, e_1, e_2). Consider the partial output of φ obtained by substituting o, e_1, and e_2 for $\widehat{z}_0$, $\widehat{z}_1$, and $\widehat{z}_2$, respectively. Clearly, φ simulates ψ (with points on the line oe_1 being used to represent real numbers), with the exception that the two components of a point in $\mathbf{R}^2$ are its coordinates with respect to the standard basis whereas in $\widehat{P}$, their representations refer to the basis (o, e_1, e_2). Thus the partial output of φ considered equals $\alpha(\psi(\alpha^{-1}(\mathcal{D})))$. Since there is a one-to-one correspondence between the affine transformations and the bases of $\mathbf{R}^2$, it follows that $\varphi(\mathcal{D}) = \bigcup_\alpha \alpha(\psi(\alpha^{-1}(\mathcal{D})))$, where α ranges over all affine transformations of $\mathbf{R}^2$. Since ψ is affine-generic, $\alpha(\psi(\alpha^{-1}(\mathcal{D}))) = \psi(\alpha(\alpha^{-1}\mathcal{D})) = \psi(\mathcal{D})$ for each α, whence $\varphi(\mathcal{D}) = \psi(\mathcal{D})$.

It must be pointed out that the above sketch of argument is not completely rigorous. Indeed, $\alpha^{-1}(\mathcal{D})$ need not be semi-algebraic for arbitrary α (under our definition of semi-algebraic, which requires integer coefficients) so that the definition of genericity (which considers only semi-algebraic databases) cannot be applied immediately. However, it can be shown that if $\psi(\alpha(\mathcal{D}) = \alpha(\psi(\mathcal{D}))$ for all semi-algebraic $\alpha(\mathcal{D})$, then the equality holds for *all* $\alpha(\mathcal{D})$. ■

If one is not interested in effective computability but only in the purely logical aspects of the above theorem, one may consider arbitrary (i.e., not necessarily semi-algebraic) structures as geometric databases and consider arbitrary (i.e., not necessarily computable) functions as geometric queries. Since the proof of Theorem 3 is purely logical, it will remain valid in such a setting.

6 Discussion

6.1 Extension to a model with non-spatial data values

In the model we have been using so far, a database can contain semi-algebraic sets only. Practical spatial database models support, in addition to purely spatial data, also data values without a geometrical interpretation, such as the data stored in classical relational databases. For example, for a road one typically not only wants to store its appearance on a map as a curve (a semi-algebraic set), but also its name or number.

It is not difficult to extend our model to incorporate non-spatial data; this was presented in [7]. This extended model fits nicely in the model for the query language EQL described by Chandra and Harel [2]. EQL is an extension of the well-known QL, a complete query language for classical relational databases. The extension supports the occurrence of fully interpreted data values in relations. In our application of this model these interpreted data values are real formulas.

The key construct of EQL is an operator for going from an i-ary relation to the i-th interpreted data value. In a direct combination of the query languages QL and FO[**R**] + while, this construct can be expressed. The QL component of the combined language deals with the projection of the relations on the non-spatial data columns, and the FO[**R**] + while component deals with the spatial projection. Based on this observation, it can be verified that the combined query language "QL⊕(FO[**R**]+while)" is complete for semi-algebraic databases extended with non-spatial data.

6.2 Open problem

A problem that remains open concerns the expressive power of the query language FO[between] + while, the extension of FO[between] to a full-fledge programming language with assignment statements and while-loops. Is it complete for the affine-generic queries? Extension of our proof of Theorem 3 to deal with while-loops presents serious difficulties. If one wants to simulate the stopping condition of a while-loop in an FO[**R**] + while program by an FO[between]-formula one obtains a formula with spurious free parameters $\widehat{z}_0$, $\widehat{z}_1$ and $\widehat{z}_2$. Applying an existential quantifier to these variables does not work, since this often leads to a simulating while-loop that never terminates. A possible way to circumvent this problem may be to extend the language by allowing while-loops with stopping conditions that have free parameters, as in query languages proposed by Abiteboul and Vianu. Such while-loops can be given a non-deterministic semantics [1].

References

[1] S. Abiteboul and V. Vianu. Procedural languages for database queries and updates. *Journal of Computer and System Sciences*, 41(2):181–229, 1990.

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

[3] H.B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[4] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, August 1995.

[5] B. Kuijpers, J. Paredaens, and D. Suciu. Unpublished results. University of Antwerp, 1995.

[6] C.H. Papadimitriou, D. Suciu, and V. Vianu. Topological queries in spatial databases. In *Proceedings 15th ACM Symposium on Principles of Database Systems*, pages 81–92. ACM Press, 1996.

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

[8] W. Schwabhäuser, W. Szmielew, and A. Tarski. *Metamathematische Methoden in der Geometrie.* Springer-Verlag, 1983.

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

[10] A. Tarski. What is elementary geometry? In J. Hintikka, editor, *The Philosophy of Mathematics*, pages 164–175. Oxford University Press, 1969.

[11] L. Van Den Dries. Alfred Tarski's elimination theory for real closed fields. *Journal of Symbolic Logic*, 53:7–19, 1988.